Fundamentos de Geometria Computacional

Pedro J. de Rezende
Jorge Stolfi

Departamento de Ciência da Computação
Universidade Estadual de Campinas
Campinas SP, Brasil
{rezende|stolfi}@dcc.unicamp.br

IX Escola de Computação
Julho de 1994

Aos nossos filhos.

Agradecimentos

À Comissão Organizadora da IX Escola de Computação pela oportunidade e pela paciência;

Ao CNPq e à Fapesp pelo apoio financeiro;

Aos colegas do Departamento de Ciência da Computação da Unicamp pelos incentivos e pela cobrança;

Aos nossos estudantes que suportaram notas preliminares incompletas deste texto;

À Ketty e Rumiko por conduzirem tão bem todo o resto.

Prefácio

Este texto é uma introdução à *geometria computacional*, a disciplina que estuda técnicas, algoritmos, e estruturas de dados para a resolução de problemas geométricos por computador.

Os problemas estudados são geralmente restritos ao plano, com incursões ocasionais pelo espaço tridimensional. Pressupõe-se que o leitor tenha conhecimentos elementares de geometria euclidiana e analítica, ao nível de segundo grau; e de álgebra linear, programação, e estruturas de dados, ao nível de primeiro ano de graduação.

O primeiro capítulo do livro introduz conceitos básicos de projeto e análise de algoritmos — especificamente, análise assintótica do pior caso. O leitor é alertado para a existência de muitos algoritmos para cada problema, que podem diferir bastante em eficiência.

O segundo capítulo descreve as ferramentas elementares da geometria computacional, incluindo a representação de pontos e linhas por meio de coordenadas homogêneas, e as operações geométricas básicas, tais como o cálculo da reta que passa por dois pontos dados, ou o teste da orientação (horária ou anti-horária) de um triângulo.

A fim de motivar e ilustrar o conceito de *algoritmo geométrico*, o capítulo 3 estuda o problema de determinar o par mais próximo dentre uma coleção de pontos dada e também o problema de localização de pontos com relação a um polígono simples, estrelados ou convexos.

O capítulo 4 estuda problemas relacionados com convexidade, em particular a construção do casco convexo de pontos no plano. Diversos algorimos são apresentados, ilustrando diferentes paradigmas de projetos de soluções.

O tema do capítulo 5 é o conceito de *mapa planar*: uma divisão de uma superfície em pontos, curvas, e regiões, disjuntas duas a duas. Em particular, ele define a *superposicão* de dois mapas, uma operação geral que pode ser usada para muitas aplicações, como por exemplo a intersecção e união de polígonos arbitrários. O paradigma da *varredura* do plano é apresentado como solução eficiente para este tipo de problema.

Finalmente, o capítulo 6 estuda vários problemas relacionados com minimização de distâncias entre conjuntos de pontos no plano. As soluções para todos esses problemas estão baseadas na mesma estrutura geométrica, o *diagrama de Voronoi*, e seu dual, a *triangulação de Delaunay*, que são duas das mais importantes e versáteis ferramentas da geometria computacional. Estudamos também como obter, a partir da triangulação de Delaunay, certas redes geométricas importantes, incluindo a *árvore conectora mínima*, o *grafo de Gabriel*, e o *grafo de vizinhança relativa*.

Conteúdo

Capítulo 1

Algoritmos e Complexidade

Antes de estudarmos problemas e algoritmos geométricos é necessário estabelecermos uma terminologia básica para podermos falar de aspectos de complexidade computacional.

Como o leitor já deve ter pelo menos alguns conhecimentos básicos de projeto e análise de algoritmos, o propósito deste capítulo é apenas o de estabelecer com o leitor uma linguagem comum. Diversos textos tratam muito mais extensamente os conceitos aqui descritos e recomendamos especialmente a consulta a [CLR90, Man89, Sed83, AHU74] para aqueles interessados em maiores detalhes.

1.1 Introdução

Há duas facetas no estudo de problemas computacionais do ponto de vista de algoritmos e complexidades. De um lado está a obtenção de soluções eficientes para um dado problema. Do outro está a necessidade de, obtidas soluções eficientes, decidir até que ponto é possível obter outras ainda mais eficientes.

A primeira destas questões consiste, como veremos, da descoberta de algoritmos e do estabelecimento da *eficiência* dos métodos de soluções que estes descrevem. Do outro lado, temos a questão de determinar a dificuldade intrínseca do dado problema — denominada *complexidade* do problema — isto é, encontrar uma expressão para a máxima eficiência atingível por qualquer solução do problema.

1.2 Conceito de algoritmo

Um *algoritmo* é uma receita matemática para resolver algum problema.

O adjetivo “matemática” implica que a receita deve ser tão precisa, correta, e detalhada quanto qualquer teorema, ou construção geométrica.

Assim como em teoremas, o grau de detalhe e formalismo que um algoritmo deve ter depende do “leitor” almejado. No caso extremo em que o “leitor” é um computador, todos os detalhes devem ser especificados, numa notação que possa ser interpretada mecanicamente — ou seja, o algoritmo deve ser apresentado na forma de um programa.

No caso deste livro, suporemos que o “leitor” é uma pessoa: por exemplo, um programador, que pode vir a utilizar os algoritmos aqui apresentados em seus programas; ou um cientista, que pode usar estes algoritmos como ponto de partida para desenvolver algoritmos originais. Portanto, ao descrever nossos algoritmos, usaremos uma linguagem semi-formal, semelhante à usada em demonstrações de teoremas; e omitiremos em geral todos os detalhes que podem ser facilmente supridos por um programador — alocação de memória, declarações de variáveis, entrada e saída de dados, enumeração de listas, etc. Além disso, detalhes que são explicados num algoritmo podem ser omitidos nos algoritmos subseqüentes.

Algoritmo Um algoritmo deve conter os seguintes ingredientes:

1. um *enunciado*,
2. um *modelo computacional*,
3. uma descrição da *seqüência de operações* a efetuar,
4. uma *prova de correção*, e
5. uma *análise de desempenho.*

Examinaremos a seguir cada um destes itens com mais detalhe.

1.2.1 O enunciado

O *enunciado* do algoritmo descreve os dados que devem ser fornecidos ao algoritmo, os resultados calculados pelo mesmo, e a relação entre ambos.

Por exemplo, o enunciado de um algoritmo para inversão de matrizes poderia ser:

Algoritmo 1.1 Inv (A, n)
Dados um inteiro n e uma matriz real A de dimensão $n \times n$, devolve uma matriz real B $n \times n$ tal que AB é a matriz identidade; ou uma mensagem de erro, se tal matriz não existir.

Os resultados previstos no enunciado de um algoritmo podem não fazer nenhuma pré-suposição a respeito dos dados (como neste exemplo) ou podem presumir que o objeto a ser computado é sempre computável a partir dos dados fornecidos. Neste caso, o enunciado deve indicar precisamente que propriedades seus dados de entrada devem satisfazer.

1.2.2 O modelo computacional

Em linhas gerais, estabelecer um modelo computacional consiste em definir quais são as operações permitidas e qual o custo de cada uma delas, de modo a podermos computar a eficiência (ou ineficiência) de soluções.

Este repertório de operações "elementares," que podem ser utilizadas pelos algoritmos que se deseja descrever neste modelo, deve ser tal que o executor do algoritmo — pessoa ou computador — supostamente já as sabe efetuar.

No caso de algoritmos destinados a computadores, as operações elementares são aquelas que o computador pode efetuar com um número fixo de instruções. Estas incluem, em geral: as quatro operações aritméticas (em inteiros ou reais de precisão limitada); operações com valores lógicos; transferência de dados entre posições de memória; indexação de vetores e matrizes; endereçamento indireto; e outras de mesmo nível de complexidade.

Em particular, no caso de algoritmos geométricos, incluímos entre as operações elementares todas as construções geométricas que podem ser efetuadas analiticamente, com um número fixo de operações aritméticas a partir das coordenadas dos dados. Como veremos no capítulo 2, esta categoria inclui determinar a reta que passa por dois pontos, o ponto de intersecção de duas retas, etc.

1.2.3 A seqüência de operações

O item central do algoritmo é uma especificação da *seqüência de operações* que devem ser efetuadas.

Em geral, a seqüência depende dos dados de entrada, e estes podem ser infinitamente variáveis. Portanto, precisamos de uma notação que seja capaz de descrever esta infinidade de seqüências de maneira finita.

Muitas notações foram propostas para esse fim, voltadas tanto para computadores (linguagens de programação) quanto para leitores humanos. Neste livro, usaremos em geral uma descrição semi-formal baseada em comandos que têm similares na maioria das linguagens de programação. Em alguns casos, por clareza de exposição usaremos uma descrição ainda mais informal.

Dada a necessidade de descrevermos operações repetitivas, usaremos tanto recursão quanto laços iterativos. Assumiremos que o leitor tem familiaridade com a descrição de algoritmos em forma recursiva. Se este não é o caso, recomendamos como uma introdução a algoritmos recursivos, o capítulo 5 de [Sed83].

Em particular, usaremos os seguintes comandos, cujos significados podem não ser óbvios:

para *variável* **em** *seqüência* **faça** *comando*

> Repete o *comando* uma vez para cada valor na *seqüência* de valores especificada, com a *variável* representando esse valor.
>
> Note que se a *seqüência* for vazia, o *comando* não é executado nenhuma vez.

enquanto *condição* **repita** *comando*

> Testa a *condição*; se for verdadeira, executa o *comando*, e repete tudo de novo — testa novamente a

condição, etc. A repetição termina somente se e quando a *condição* testada é falsa.

Note que o teste da *condição* é feito *antes* de cada execução do *comando*. Portanto, se a *condição* for falsa na primeira vez, o *comando* não é executado nenhuma vez.

devolva *fórmulas*

Termina a execução do algoritmo, devolvendo os valores das *fórmulas* como resultados do mesmo.

Ex. 1.1: Escreva, usando os comandos **para** ou **enquanto**, um algoritmo que:

(a) Calcule o produto $m \cdot n$ de dois inteiros quaisquer m e n, usando apenas a operação de soma;

(b) Calcule a potência x^y de dois inteiros positivos x e y usando o algoritmo do item (a) ao invés da operação $\times$.

Ex. 1.2: Refaça o exercício anterior (sem usar **para** ou **enquanto**), através do projeto de algoritmos recursivos.

1.2.4 A prova de correção

Para que um algoritmo possa ser usado com confiança, é preciso provar que ele satisfaz as seguintes condições:

1. *As operações que o algoritmo manda executar são sempre válidas e definidas, quaisquer que sejam os dados fornecidos ao mesmo.*

   Por exemplo, se o algoritmo pede para calcular o quociente x/y, é preciso provar que nesse momento y será sempre diferente de zero.

2. *O algoritmo sempre termina após um número finito de operações.*

   Este item precisa ser explicitamente demonstrado quando o algoritmo contém comandos de repetição do tipo **enquanto** *condição*

repita ... . Neste caso, temos que provar que a *condição* sempre será falsa depois de um número finito de repetições.

Este item também precisa ser provado explicitamente quando o algoritmo executa a si próprio recursivamente. Neste caso, temos que provar que os argumentos de cada aplicação recursiva do algoritmo são mais "simples" do que os dados originais, segundo algum critério de simplicidade; e que, depois de um número finito de tais "simplificações," os dados se reduzem a um caso particular (*caso base*) que é resolvido diretamente pelo algoritmo, sem recursão.

3. *O resultado devolvido pelo algoritmo satisfaz as condições prometidas no enunciado do mesmo.*

   Esta é em geral a parte mais difícil da prova, e a que exige um entendimento mais profundo do problema e do algoritmo. No caso dos nossos algoritmos, esta parte de demonstração vai depender crucialmente das propriedades fundamentais do espaço geométrico utilizado, que serão vistas no capítulo 2).

Muitas vezes vamos apresentar apenas as idéias principais da prova de correção de um algoritmo, e deixar a demonstração detalhada e rigorosa a cargo do leitor.

Entretanto, deve ficar claro que um algoritmo só pode ser levado a sério depois que tal demonstração for apresentada. Para que você possa acreditar que um algoritmo funciona é preciso que você tenha na sua mente algo que seja equivalente a uma prova matemática de sua correção. Testes empíricos podem aumentar a confiança, mas nunca podem dar certeza: dada a infinita variedade de dados de entrada, um número finito de experimentos sempre pode deixar de testar os casos particulares em que o algoritmo falha.

O primeiro passo para demonstrar a correção de um algoritmo é determinar um *invariante* apropriado para cada comando do mesmo. O invariante de um comando é uma afirmação lógica sobre as variáveis e dados do programa que supostamente vale sempre que, durante a execução do algoritmo, chegar a vez de executar o comando em questão. Se os invariantes forem corretamente escolhidos, basta então provar que, se um invariante for verdade, então o comando correspondente está bem definido, e sua execução torna válido o invariante do comando imediatamente seguinte. Finalmente, temos que provar que o invariante de todo comando **devolva** *fórmulas* implica que as *fórmulas* devolvidas como resultado satisfazem o enunciado do algoritmo.

Por outro lado, veremos na seção 1.6 que uma maneira alternativa de construir algoritmos corretos é utilizar conversões de provas construtivas da existência de soluções, em procedimentos. Em particular, provas por indução sugerem muito naturalmente algoritmos recursivos, ao mesmo tempo em que se constituem provas da correção destes.

1.2.5 A análise de desempenho

Como veremos, sempre existem infinitos algoritmos diferentes para resolver um mesmo problema, o que torna importante a escolha do algoritmo mais apropriado para uma determinada aplicação. Há muitas maneiras de comparar esses algoritmos: eficiência, flexibilidade, complexidade, robustez, tratamento de dados incorretos, facilidade de uso, etc. Muitas destas propriedades são difíceis de quantificar, e portanto de estudar cientificamente.

Uma propriedade que é realmente fácil de quantificar é o *custo* do algoritmo, que inclui tanto o *tempo* que o algoritmo leva para ser executado, quanto o *espaço* que as variáveis internas ocupam nas unidades de armazenamento de dados do computador (memória, disco, etc.). Como veremos, é comum encontrar dois algoritmos para o mesmo problema que são praticamente equivalentes sob os demais aspectos, mas cujos custos de execução diferem por fatores astronômicos.

Estes custos são quantificados em termos do "tamanho" dos dados do problema — medido, por exemplo, pelo número de elementos ou como o espaço necessário para representá-los. Usualmente, chamamos estes custos de *eficiência* ou *complexidade* do algoritmo.

O leitor familiarizado com algoritmos de ordenação se recordará que duas medidas de custo relevantes para a comparação de eficiência deles é o número de comparações realizadas e o número de atribuições. Por exemplo, para ordenar um vetor de n elementos:

- ordenação por seleção faz $O(n^2)$ comparações e $O(n)$ atribuições;

- ordenação por inserção faz $O(n \log n)$ comparações e $O(n^2)$ atribuições;
- ordenação por intercalação faz $O(n \log n)$ comparações e $O(n)$ atribuições.

Entretanto, o tempo exato de execução T (em segundos) não pode em geral ser determinado a partir da descrição semi-formal do algoritmo, pois esse tempo depende dos detalhes da codificação do algoritmo em forma de programa, e de sua tradução para a linguagem de máquina do computador; e, naturalmente, do modelo de computador utilizado. Assim, temos que nos contentar em contar o número t de operações elementares efetuadas pelo mesmo. Observe que, se cada operação elementar pode ser traduzida num número fixo de instruções da máquina, esta medida será aproximadamente proporcional ao tempo de execução T; ou seja, existem constantes a_1, b_1, a_2, b_2 tais que $a_1 t + b_1 \leq T \leq a_2 t + b_2$.

Mais ainda, uma vez que as constantes a_1 e a_2 dependem da máquina e do programador, não é muito importante determinar o número exato de operações t; basta determinar a maneira como esse número varia com o tamanho do problema. Assim, expressaremos o desempenho de um algoritmo através de uma função cujo parâmetro será o tamanho da entrada deste. Chamaremos de *cota superior* (da complexidade) de um problema qualquer função que expresse a eficiência[1] de algum algoritmo que o resolve. O objetivo do projetista de algoritmos é em geral o de obter a menor cota superior possível.

Cota superior

Ferramentas para comparação entre funções, considerando o comportamento delas à medida que o parâmetro comum a elas cresce (comportamento *assintótico*), serão vistas na seção 1.3.

1.3 Análise assintótica

Tanto a comparação das eficiência de algoritmos para um dado problema quanto a comparação das complexidades de diferentes problemas (como discutiremos na seção 1.4) requerem que possamos comparar o

[1] Estaremos interessados em eficiência de soluções em *pior caso* a menos que seja explicitamente indicado o contrário.

crescimento assintótico de funções. Para isso, convém desenvolvermos uma notação concisa.

Sejam f, $g : \mathbb{N} \to \mathbb{N}$ duas funções positivas. Se o limite do quociente f/g quando n tende a infinito é definido, podemos classificar f e g com respeito a seus crescimentos assintóticos baseando-nos no valor deste limite (zero, positivo finito ou infinito). Há porém situações em que este limite não é definido nas quais ainda é necessário fazer esta classificação.

Assim, definiremos classes de funções baseados nas noções mais fracas de limites inferior ($\underline{\lim}$) e superior ($\overline{\lim}$). Se f e g são funções positivas, então os limites parciais $\underline{\lim}_{n\to\infty} f(n)/g(n)$ e $\overline{\lim}_{n\to\infty} f(n)/g(n)$ sào sempre definidos (números reais ou infinitos).

Dizemos então que:

As classes o, O, Θ, Ω e ω

$$f \in o(g) \text{ se } \quad \overline{\lim_{n\to\infty}} \frac{f(n)}{g(n)} = 0$$

$$f \in O(g) \text{ se } \quad \overline{\lim_{n\to\infty}} \frac{f(n)}{g(n)} < \infty$$

$$f \in \Theta(g) \text{ se } \quad 0 < \underline{\lim_{n\to\infty}} \frac{f(n)}{g(n)} \text{ e } \overline{\lim_{n\to\infty}} \frac{f(n)}{g(n)} < \infty$$

$$f \in \Omega(g) \text{ se } \quad 0 < \underline{\lim_{n\to\infty}} \frac{f(n)}{g(n)}$$

$$f \in \omega(g) \text{ se } \quad \infty = \underline{\lim_{n\to\infty}} \frac{f(n)}{g(n)}$$

$$f \text{ e } g \text{ são incomparáveis se } \quad 0 = \underline{\lim_{n\to\infty}} \frac{f(n)}{g(n)} \text{ e } \overline{\lim_{n\to\infty}} \frac{f(n)}{g(n)} = \infty$$

Como o limite inferior é sempre menor que ou igual ao limite superior, a definição acima cobre todos os casos e portanto, a união das classes $o(g)$, $O(g)$, $\Theta(g)$, $\Omega(g)$ e $\omega(g)$ contém todas as funções comparáveis com a função g.

O leitor pode se sentir ainda mais confortável com esta notação pensando inicialmente em termos de funções cujo limite do quociente é definido, já que nestes casos temos:

$$\varliminf_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \varlimsup_{n\to\infty} \frac{f(n)}{g(n)}.$$

Ex. 1.3: É fácil ver que existem intersecções entre algumas destas classes como ilustra a figura 1.1. Mostre que:

(a) $o(g) \subset O(g)$

(b) $\Theta(g) = O(g) \cap \Omega(g)$

(c) $\Omega(g) \supset \omega(g)$

(d) $o(g) \cap \omega(g) = \emptyset$

(e) $o(g) \cap \Theta(g) = \emptyset$

(f) $\Theta(g) \cap \omega(g) = \emptyset$.

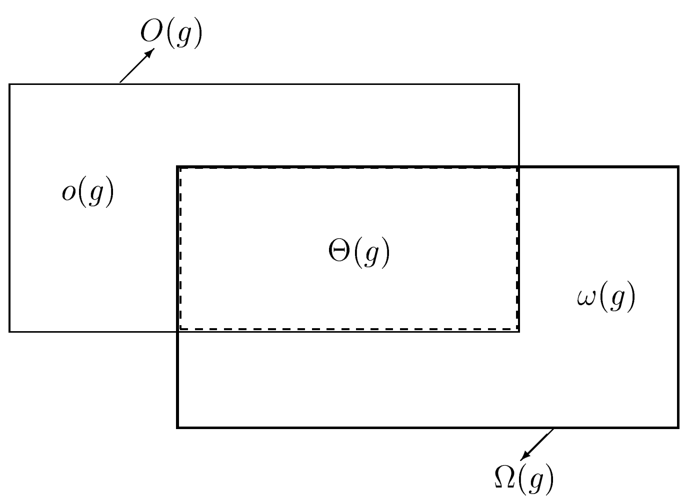


Figura 1.1: Classes de funções

A razão para se estar interessado na complexidade *assintótica* é que uma função que a expressa pode ter um comportamento, para instâncias de tamanhos limitados, que não corresponde à tendência de seu crescimento para entradas de tamanho crescente. Isto é, o comportamento do crescimento da função fica mascarado por constantes multiplicativas que não contribuem para a identificação de seu crescimento. Por exemplo, se o parâmetro n mede o tamanho da entrada de um problema P, tanto faz dizermos que P tem complexidade assintótica $n/2$, n ou $100n$. O crescimento de todas estas funções é linear em n. Se, porém, pudermos mostrar que P também tem cota inferior expressa por $n^{3/2}/100$, esta é mais significante do que qualquer das expressões lineares anteriores. Embora $n^{3/2}/100$ seja menor que $100n$ para $n < 10^8$, o comportamento assintótico de $n^{3/2}/100$ é maior que o de $100n$, isto é:

$$\frac{n^{3/2}}{100} \in \omega(100n).$$

Ex. 1.4: Mostre que:

(a) $\sqrt{n} \in O(n)$	(b) $100n \in \Theta(n)$
(c) $n \log n \in o(n^2)$	(d) $n^k \in o(2^n)\ \forall k \in \mathbb{N}$
(e) $n^{1/4} \in \Omega(\log^4 n)$	(f) $n^\alpha \in \omega(\log^k n)\ \forall k \in \mathbb{N},\ \forall \alpha > \nvdash$

Por outro lado, observe que:

$$\begin{array}{lll} f \in o(g) & \text{se e somente se} & g \in \omega(f) \\ f \in O(g) & \text{se e somente se} & g \in \Omega(f) \\ f \in \Theta(g) & \text{se e somente se} & g \in \Theta(f) \\ f \in \Omega(g) & \text{se e somente se} & g \in O(f) \\ f \in \omega(g) & \text{se e somente se} & g \in o(f) \end{array}$$

Informalmente, referimos às funções das várias classes dizendo que:

$$\begin{array}{rl} f \text{ cresce mais lentamente que } g & \text{se } f \in o(g) \\ f \text{ cresce no máximo tão rapidamente quanto } g & \text{se } f \in O(g) \\ f \text{ cresce tão rapidamente quanto } g & \text{se } f \in \Theta(g) \\ f \text{ cresce no mínimo tão rapidamente quanto } g & \text{se } f \in \Omega(g) \\ f \text{ cresce mais rapidamente que } g & \text{se } f \in \omega(g) \end{array}$$

Veremos já no capítulo 3 alguns exemplos do uso dos conceitos e da terminologia aqui estabelecidos.

1.4 Complexidade de problemas

A escolha de um modelo computacional adequado é importante tanto para se estabelecer as operações levadas em conta na determinação da eficiência de soluções, quanto para o estabelecimento da complexidade intrínseca de problemas neste modelo, isto é, o melhor que se pode esperar para a eficiência de quaisquer soluções que usem as operações do modelo.

Um modelo computacional só é relevante para um problema se as operações nele previstas podem permitir *alguma* solução do problema. Por exemplo, um modelo que permite apenas o uso de operações de comparação ($=, \leq, <$, etc.) entre elementos da entrada é relevante para o problema de ordenação mas não para o problema da multiplicação de matrizes; enquanto um modelo que permite apenas as operações aritméticas ($+, -, \times, \div$) sobre o conjunto dos números reais é relevante para o segundo mas não para o primeiro.

1.4.1 Cotas inferiores

Fixado um problema e um modelo computacional (relevante), limitamos nossa atenção às soluções que sejam descritíveis neste modelo.

A *complexidade assintótica* de um problema é expressa por uma função que indica um número mínimo de operações que tem que ser realizado por qualquer solução do problema, em função do tamanho da entrada. Costuma-se utilizar também o termo *cota inferior* (da complexidade) do problema. A tarefa de quem faz análise de complexidade de problemas é sempre a de obter a maior cota inferior possível.

Cota inferior

1.5 Modelos computacionais

Para acompanhar esta seção, o leitor deve ter alguma familiaridade pelo menos com o modelo de árvore de decisões binárias no qual se pode mostrar que o problema de ordenação de n números reais requer a realização de pelo menos $cn \log n$ comparações entre os elementos a serem ordenados, para alguma constante c positiva — veja, por exemplo, [Baa88]. (Recomendamos a leitura de um texto de projeto e análise de algoritmos [CLR90], [Man89], [Baa88], [Sed83], [AHU74] para quem os parágrafos seguintes soem completa novidade.)

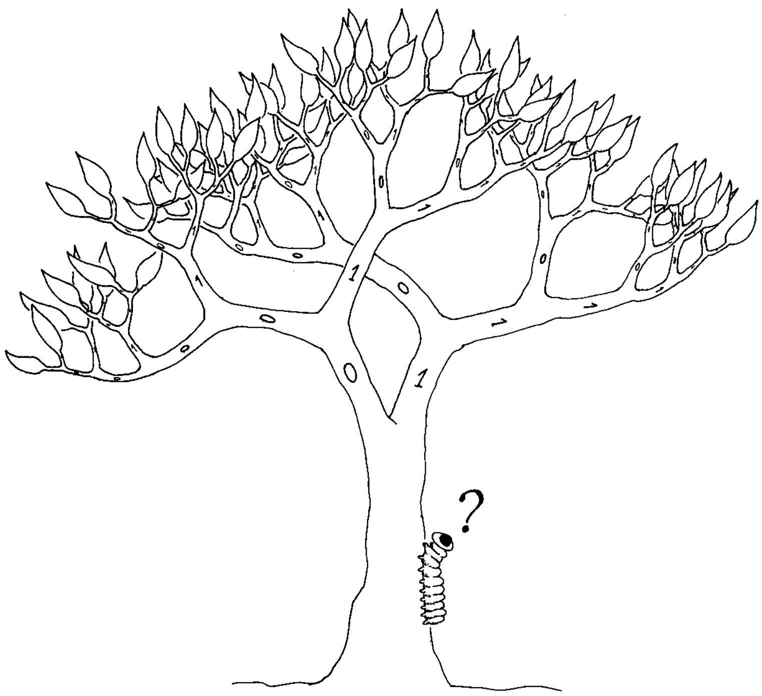


Figura 1.2: Árvore de decisões binária.

O modelo de árvore de decisões binárias é, infelizmente, demasiado fraco para a solução de problemas geométricos em geral. Observe que nem mesmo operações aritméticas entre as coordenadas dos objetos de entrada são permitidas. São portanto necessárias extensões deste modelo, mas ainda atentas à importância da operação de comparação na determinação de complexidades.

Modelos computacionais chamados *modelos de decisões* são aqueles nos quais as operações de comparação desempenham um papel essencial. Algoritmos descritíveis nestes modelos podem ser representados na forma de árvore (em geral binária) onde, em cada nó interno é realizada uma operação de comparação entre dois valores, enquanto nas folhas são armazenados resultados de saída. As funções utilizadas para a geração dos valores usados nas comparações determinam o tipo de modelo, isto é, são nestas funções que aparecem as outras operações permitidas no modelo.

Seja A um algoritmo que se deseja representar num destes modelos e seja $X = \{x_1, x_2, \ldots, x_n\}$ uma entrada para o algoritmo. Em cada nó interno da árvore correspondente a A é realizada uma comparação entre os valores gerados por duas funções em X.

Árvore de decisões binárias

O modelo de *árvore de decisões binárias* é aquele que permite apenas o uso de funções de projeção (da forma $f(X) = x_\ell$ para algum $1 \leq \ell \leq n$), isto é, não são permitidas operações de nenhuma espécie com as componentes da entrada X.

Árvore de decisões lineares

O modelo de *árvore de decisões lineares* permite apenas operações de adição e produto escalar sobre as componentes da entrada, isto é, as funções são necessariamente lineares (da forma $f(X) = \sum_{i=1}^{n} r_i \cdot x_i$ com $r_i \in \mathbb{R}$ constantes).

Árvore de decisões quadráticas

No modelo de *árvore de decisões quadráticas*, *cúbicas*, etc., é permitido que as funções sejam polinômios nas componentes de X, de grau até dois, três, etc.

Árvore de decisões algébricas

Um modelo ainda mais geral é o de *árvore de decisões algébricas* que admite funções polinomiais em geral, sem limitação de grau.

Note que estes modelos são gradativamente mais poderosos no sentido que todo algoritmo representável num deles admite também representação nos modelos seguintes.

Resumimos abaixo estes quatro modelos:

Funções	Modelo
Projeções	Árvore de Decisões Binárias
Lineares	Árvore de Decisões Lineares
Quadráticas	Árvore de Decisões Quadráticas
⋮	⋮
Polinomiais	Árvore de Decisões Algébricas

No capítulo 2, veremos que a determinação de orientação de um triângulo pode ser feita utilizando-se o cálculo de um determinante 3×3, para o qual é necessário calcular produtos entre as coordenadas dos vértices do triângulo. Portanto, o modelo mais fraco no qual se pode utilizar esse método é o modelo de árvore de decisões quadráticas. De fato, o leitor interessado encontrará em [Avi80] uma demonstração de que o modelo de árvore de decisões lineares é demasiado fraco para a construção de cascos convexos (vide capítulo 4). Isso mostra que provas de cota inferior para este problema nesse modelo (como aparece em [vE80]) são irrelevantes.

Por uma questão de conveniência, especialmente para o estabelecimento de cotas inferiores para problemas geométricos, assumiremos doravante que o modelo utilizado é o de árvore de decisões algébricas.

1.6 Paradigmas de projeto de algoritmos

Indução: técnica para prova construtiva de existência de soluções.

Dado um problema cujo tamanho da entrada pode ser expresso por um parâmetro n, suponha que mostramos a existência de uma solução utilizando uma demonstração por indução em n. Transformar esta demonstração em um algoritmo recursivo que constrói uma tal solução é um trabalho usualmente bastante simples já que os ingredientes principais do algoritmo aparecem explicitamente na demonstração: a base da indução é a condição de parada da recursão; a hipótese de indução é a chamada recursiva do algoritmo e o passo da indução é a operação de efetiva construção da solução para a entrada de tamanho n a partir das soluções construídas recursivamente.

Por outro lado, para se estabelecer a correção de um algoritmo assim construído, basta provar que a tradução da demonstração para o procedimento é correta, pois a validade da construção da solução é estabelecida pela prova indutiva.

Assim, convém que recapitulemos algumas formas de provas por indução.

1.6.1 Provas por indução

Dado um problema P que requer que se obtenha uma solução para uma entrada X com n elementos, suponha que queremos demonstrar por indução que é possível obter tal solução. Se denotamos por $\mathcal{P}(X)$ a assertiva:

> *Se X é uma entrada com n elementos para o problema P, então sabemos obter uma solução $S_P(X)$ para a entrada X.*

então queremos demonstrar que $\mathcal{P}(X)$ é verdadeiro, por indução em n onde $|X| = n \geq n_0$ para algum inteiro $n_0 \geq 1$ (em geral, $n_0 = 1$).

Denotemos os elementos da entrada X por $\{x_1, x_2, \ldots, x_n\}$.

Podemos usar diversas formas de indução e cada uma delas leva a um paradigma de projeto de algoritmos que, como veremos a seguir, inclui os paradigmas incremental, de varredura, e de divisão e conquista, entre outros.

Prova por indução simples

Base: Provar que $\mathcal{P}(\{x_1, x_2, \ldots, x_{n_0}\})$ é verdadeiro para algum $n_0 \geq 1$.

Hipótese: Assumir que $\mathcal{P}(X')$ é verdadeiro para $|X'| = n - 1$.

Passo: Provar que $\mathcal{P}(X)$ é verdadeiro para $|X| = n$.

Técnica 1

Remover um elemento arbitrário x_i de X, formando $X' = X - \{x_i\}$. Por hipótese de indução, $\mathcal{P}(X')$ é verdadeiro. Basta mostrar que $\mathcal{P}(X) = \mathcal{P}(X' \cup \{x_i\})$ é verdadeiro, a partir da veracidade de $\mathcal{P}(X')$.

Paradigma Incremental

Paradigma Incremental

O seguinte algoritmo *incremental* (recursivo) pode ser obtido a partir da prova acima:

Algoritmo 1.2 $\mathrm{Inc}_P(X)$

1. Se $|X| = n_0$ então devolva a solução construída diretamente baseada na prova do caso base.
2. Senão, escolha um elemento arbitrário $x_i \in X$.
   Seja $X' = X - \{x_i\}$.
   - 2.1. Obtenha recursivamente $S_P(X') = \mathrm{Inc}_P(X')$.
   - 2.2. Devolva a solução $S_P(X)$ obtida estendendo $S_P(X')$ de modo a incluir x_i conforme o passo da prova indutiva.

Complexidade

Por exemplo, se o passo de extensão da solução parcial para a solução final toma tempo $T_c(n)$, então a seguinte relação de recorrência expressa a complexidade total do algoritmo:

$$\begin{cases} T(n) = T(n-1) + T_c(n) \text{ para } n > n_0 \\ T(n_0) \in O(1). \end{cases} \tag{1.1}$$

Se $T_c(n) \in O(n)$, então $T(n) \in O(n^2)$, mas se $T_c(n) \in O(\log n)$, então $T(n) \in O(n \log n)$.

Técnica 2

Ordenar os n elementos de X segundo algum critério (redenominando-os $\{x_1, x_2, \ldots, x_n\}$ nessa ordem) e remover o mais "à direita," x_n, formando $X' = X - \{x_n\}$. Por hipótese de indução, $\mathcal{P}(X')$ é verdadeiro. Basta mostrar que $\mathcal{P}(X) = \mathcal{P}(X' \cup \{x_n\})$ é verdadeiro, a partir da veracidade de $\mathcal{P}(X')$.

Paradigma de Varredura

O seguinte algoritmo *de varredura* (recursivo) pode ser obtido a partir da prova acima:

Paradigma de Varredura

Algoritmo 1.3 $\mathrm{Var}_P(X)$

1. Ordene os elementos de X segundo algum critério (redenominando-os $\{x_1, x_2, \ldots, x_n\}$ nessa ordem).
2. Devolva o resultado do algoritmo $\text{Var-Aux}_P(X)$ abaixo.

Algoritmo 1.4 $\text{Var-Aux}_P(X)$

1. Se $|X| = n_0$ então devolva a solução construída diretamente baseada na prova do caso base.
2. Senão, seja $X' = X - \{x_n\}$.
   - 2.1. Obtenha recursivamente $S_P(X') = \text{Var-Aux}_P(X')$.
   - 2.2. Devolva a solução $S_P(X)$ obtida estendendo $S_P(X')$ de modo a incluir x_n conforme o passo da prova indutiva.

Complexidade
Como antes, se o passo de extensão da solução parcial para a solução final toma tempo $T_c(n)$, então a seguinte relação de recorrência expressa a complexidade total do algoritmo:

$$\begin{cases} T(n) = T(n-1) + T_c(n) \text{ para } n > n_0 \\ T(n_0) \in O(1). \end{cases} \tag{1.2}$$

Se $T_c(n) \in O(n)$, então $T(n) \in O(n^2)$, mas se $T_c(n) \in O(\log n)$, então $T(n) \in O(n \log n)$.

Prova por indução baseada em divisão equilibrada

Base: Provar que $\mathcal{P}(\{x_1, x_2, \ldots, x_{n_0}\})$ é verdadeiro para algum $n_0 \geq 1$.

Hipótese: Assumir que $\mathcal{P}(X')$ é verdadeiro para $|X'| \leq \lceil n/2 \rceil$.

Passo: Provar que $\mathcal{P}(X)$ é verdadeiro para $|X| = n$.

Técnica
Particionar o conjunto X de n elementos em dois sub-conjuntos X_1 e X_2 de $n/2$ elementos[2]. Por hipótese de indução $\mathcal{P}(X_1)$ e $\mathcal{P}(X_2)$ são verdadeiros. Basta mostrar que $\mathcal{P}(X) = \mathcal{P}(X_1 \dot\cup X_2)$ é verdadeiro a partir da veracidade de $\mathcal{P}(X_1)$ e $\mathcal{P}(X_2)$.

Paradigma de Divisão e Conquista

Paradigma de Divisão e Conquista

O seguinte algoritmo de *divisão e conquista* (recursivo) pode ser obtido a partir da prova acima:

Algoritmo 1.5 D&C$_P(X)$

1. Se$|X| = n_0$ então devolva a solução construída diretamente baseada na prova do caso base.
2. Senão, divida o conjunto X em dois sub-conjuntos X_1 e X_2 de $n/2$ elementos.
   2.1. Obtenha recursivamente $S_P(X_1) = \text{D\&C}_P(X_1)$ e $S_P(X_2) = \text{D\&C}_P(X_2)$.

[2] Assuma, por simplicidade, que n é uma potência de 2.

2.2. Devolva a solução $S_P(X)$ obtida estendendo $S_P(X_1)$ e $S_P(X_2)$ conforme o passo da prova indutiva.

Complexidade
Se o passo de divisão em dois conjuntos toma tempo $T_d(n)$ e o passo de extensão das duas soluções parciais para a solução final toma tempo $T_c(n)$, então a seguinte relação de recorrência expressa a complexidade total do algoritmo:

$$\begin{cases} T(n) = 2 \cdot T(n/2) + (T_d(n) + T_c(n)) \text{ para } n > n_0 \\ T(n_0) \in O(1). \end{cases} \tag{1.3}$$

Como $T_d(n) \in O(n)$ usando-se um algoritmo para cálculo da mediana em tempo linear (vide [AHU74]), se $T_d(n) + T_c(n) \in O(n)$, então $T(n) \in O(n \log n)$. Por outro lado, se despendermos tempo $O(n \log n)$ para pré-ordenar os elementos de X como fizemos no caso do algoritmo de varredura, teremos $T_d(n) \in O(1)$, mas o melhor que poderemos obter neste caso ainda é $O(n \log n)$ devido à ordenação.

Vimos portanto que um método conveniente de projetar algoritmos é simplesmente demonstrar por indução que o problema em questão é solúvel! Não há nada de realmente especial a respeito de provas por indução a não ser o fato de elas serem provas construtivas. Portanto, qualquer outra técnica que possa ser utilizada para demonstrar construtivamente a solubilidade de problemas serve como método de projeto de algoritmos e leva a soluções cujas provas de correção estão essencialmente aí embutidas.

1.7 Reduções entre problemas

Com o que vimos na seção anterior sobre paradigmas de projeto de algoritmos, e como tocamos também na questão de determinação de suas complexidades, começamos a abordar o aspecto de obtenção de cotas superiores.

Resta ver ainda neste capítulo que processos podemos utilizar para demonstrar a complexidade intrínseca de problemas.

São em geral vistas em cursos básicos de projeto de algoritmos as provas de que:

- o problema da ordenação (chamado às vezes de problema de classificação) requer $\Omega(n \log n)$ operações de comparação entre os elementos de entrada;

- o problema de busca de um objeto de dado num vetor de valores reais ordenados requer $\Omega(\log n)$ operações de comparação entre aquele objeto e os elementos do vetor.

Ambas estas provas são feitas utilizando a representação de algoritmos genéricos para cada um desses problemas num modelo de árvore de decisão conveniente. Em seguida computa-se o número de nós-folha desta árvore cujo logaritmo é uma estimativa para a altura da árvore e portanto para a cota mínima do número de comparações que o algoritmo precisa realizar para resolver o dado problema. Estas provas diretas (utilizando representações do modelo computacional e a contagem do número de operações) são em geral trabalhosas.

Em [BO83], Ben-Or apresenta ferramentas poderosas para o estabelecimento de outras cotas inferiores através da realização de provas diretas. Baseando-nos em alguns poucos problemas cujas complexidades intrínsecas foram estabelecidas nesse trabalho pioneiro, (mas que não reproduziremos aqui), vamos demonstrar todas as cotas inferiores para problemas tratadas neste livro.

A técnica que utilizaremos é menos laboriosa que as provas diretas às quais nos referimos, mas não menos rigorosa.

Considere dois problemas P e Q. Queremos definir quando o problema P é *redutível* ao problema Q em tempo $f(n)$, o que será indicado pela notação[3] $P \propto_{f(n)} Q$. De maneira informal, isso significará mostrar que dada uma instância genérica I_P do problema P ela pode ser convertida para uma instância I_Q de Q de modo que, obtida uma solução S_Q para I_Q, é possível obter a partir de S_Q uma solução para a instância I_P dada. Ambas as conversões (entre instâncias e entre soluções) devem ser executadas em no máximo $O(f(n))$ operações.

[3] O tamanho (em geral a cardinalidade) da instância I_P é o que denotamos aqui pelo parâmetro n.

Vejamos como descrever a mesma coisa de um modo um pouco mais formal. Em primeiro lugar, desejamos obter uma transformação τ_I para mapeamento de instâncias de problemas:

$$\tau_I : \mathbb{I}_{\mathbb{P}} \to \mathbb{I}_{\mathbb{Q}}$$

onde $\mathbb{I}_\bullet$ denota o conjunto de todas as instâncias do problema $\bullet$, de modo que o tempo necessário para computar $\tau_I(I_P)$ seja $O(f(n))$. Em segundo lugar, se denotamos por $\mathbb{S}_{\mathbb{P}}$ o conjunto de todas as soluções das instâncias em $\mathbb{I}_{\mathbb{P}}$, e por $\mathbb{S}'_{\mathbb{Q}} \subset \mathbb{S}_{\mathbb{Q}}$ o conjunto de todas as soluções das instâncias em $\tau_I(\mathbb{I}_{\mathbb{P}})$, queremos então obter uma transformação τ_S para mapeamento de soluções de problemas:

$$\tau_S : \mathbb{S}'_{\mathbb{Q}} \to \mathbb{S}_{\mathbb{P}},$$

com a propriedade de que a imagem por τ_S de $\tau_I(I_P)$ é uma solução de I_P, para toda instância $I_P \in \mathbb{I}_{\mathbb{P}}$.

O seguinte diagrama facilita a visualização das transformações que acabamos de descrever:

Transferência de Cotas Inferiores

$$\begin{tikzcd}
P & \propto_{f(n)} & Q \\
\mathbb{I}_{\mathbb{P}} \arrow[rr, "\tau_I"] & & \mathbb{I}_{\mathbb{Q}} \arrow[d, "A_Q"] \\
\mathbb{S}_{\mathbb{P}} & & \mathbb{S}'_{\mathbb{Q}} \arrow[ll, "\tau_S"]
\end{tikzcd}$$

Observe que podemos construir a partir de cada algoritmo A_Q que resolve instâncias arbitrárias do problema Q, um algoritmo que resolve instâncias do problema P do seguinte modo:

$$A_P \doteq \tau_S \circ A_Q \circ \tau_I,$$

isto é, dado $I_P \in \mathbb{I}_{\mathbb{P}}$, $A_P(I_P) = \tau_S(A_Q(\tau_I(I_P))) \in \mathbb{S}_{\mathbb{P}}$. Perceba que a complexidade do algoritmo A_P assim construído é a soma das complexidades das três operações que o compõem. Portanto:

$$T_{A_P} = T_{\tau_I} + T_{A_Q} + T_{\tau_S}.$$

Transferência de Cotas Superiores

Portanto, a primeira conseqüência importante da obtenção de uma redução de um problema P para outro problema Q é que podemos obter um algoritmo para P a partir de *cada* algoritmo para Q. Desta forma, temos um método simples de estabelecer cotas superiores para o problema P.

Em especial, se $P \propto_{f(n)} Q$ e se Q é solúvel em tempo $g(n) \in \Omega(f(n))$, então podemos afirmar que P também é solúvel em tempo $O(g(n))$ pois, como vimos, a complexidade do algoritmo composto $A_P = \tau_S \circ A_Q \circ \tau_I$ é, neste caso, limitada pela soma: $f(n)+g(n)+f(n) \in O(g(n))$.

Transferência de Cotas Inferiores

A segunda não menos importante conseqüência de se mostrar que $P \propto_{f(n)} Q$ é, num certo sentido, simétrica à que descrevemos acima. Suponha que sabemos que $\Omega(h(n))$ é uma cota inferiorpara o problema P, isto é, que qualquer algoritmo que resolve instâncias (arbitrárias) de P de tamanho n deve realizar pelo menos $\Omega(h(n))$ operações. Suponha ainda que a redução se dá em tempo $f(n) \in o(h(n))$. Podemos concluir que $\Omega(h(n))$ é uma cota inferior para o problema Q. De fato, se pudesse haver um algoritmo A_Q que resolvesse instâncias (arbitrárias) de Q em tempo $g(n) \in o(h(n))$, construindo o algoritmo $A_P = \tau_S \circ A_Q \circ \tau_I$ para solução do problema P, teríamos que sua complexidade seria, como vimos antes, $O(f(n) + g(n) + f(n)) \subset o(h(n))$ o que contradiz a cota inferior $\Omega(h(n))$ para P.

Moral da história: a técnica de redução entre problemas serve para a transferência de cotas superiores da direita para a esquerda e de cotas inferiores da esquerda para a direita.

Ex. 1.5: Sejam P_1 e P_2 dois problemas tais que $P_1 \propto_n P_2$ e suponha que P_1 tem cota inferior $\Omega(n \log n)$. Quais das seguintes afirmações são verdadeiras?

(a) $\Omega(n \log n)$ é também uma cota inferior para P_2.

(b) Todo algoritmo que resolve P_1 pode ser usado para resolver P_2.

(c) Todo algoritmo que resolve P_2 pode ser usado para resolver P_1.

(d) O problema P_2 pode ser resolvido no pior caso em tempo $O(n \log n)$.

Ex. 1.6: Sejam P_1 e P_2 dois problemas tais que um deles tem uma cota inferior $\Omega(n^k)$, para algum $k > 1$, num modelo computacional

$\mathcal{M}$ e o outro deles é solúvel em tempo $O(n \log n)$, no mesmo modelo computacional $\mathcal{M}$. Se P_1 é redutível a P_2 em tempo linear ($P_1 \propto_n P_2$), decida qual é qual.

Problemas de enumeração, contagem e decisão

É comum trabalharmos com três tipos de problemas em geometria computacional:

- problemas de enumeração;
- problemas de contagem;
- problemas de decisão.

Um problema é chamado de *enumeração* se ele consiste em determinar *quais* os elementos de sua entrada satisfazem a uma certa propriedade; enquanto que ele é de *contagem* se objetiva obter *quantos* elementos da entrada satisfazem àquela propriedade; e é dito ser de *decisão* se requer que se determine apenas *se existe algum* elemento da entrada que a satisfaz.

Não é difícil ver (deixamos para o leitor formalizar as reduções) que é possível reduzir em tempo linear tanto a versão de decisão de um problema para sua versão de contagem quanto esta para a versão de enumeração.

Perceba, portanto, que se demonstramos uma cota inferior super-linear ($\omega(n)$) para um problema de decisão, a mesma cota vale para as duas outras versões. Similarmente, transferências de cotas superiores se dão em tempo linear na outra direção.

Capítulo 2

Conceitos fundamentais

Neste capítulo estudaremos a representação e manipulação no computador de objetos geométricos elementares, tais como pontos, linhas, segmentos, e triângulos. Estes elementos são a base de todos as estruturas e algoritmos que estudaremos nos capítulos seguintes.

2.1 Coordenadas cartesianas

Coordenadas cartesianas

O leitor certamente sabe de longa data que um ponto do plano pode ser biunivocamente representado por um par de números reais (X, Y), suas *coordenadas cartesianas*, medidas a partir de alguma origem arbitrária ao longo de dois eixos ortogonais. A totalidade desses pares constitui o *plano cartesiano* $\mathbb{R}^{\nvDash}$

Coordenadas cartesianas, apesar de bem conhecidas e bastante usadas, não são a única maneira de representar pontos do plano no computador. Na verdade, em geometria computacional é em geral mais conveniente trabalhar com uma representação mais sofisticada, as chamadas *coordenadas homogêneas*, cujas propriedades e manipulação são o tema principal deste capítulo.

2.1.1 Desvantagens da geometria cartesiana

Não há pontos no infinito

A principal desvantagem das coordenadas cartesianas é que elas não suportam o conceito de pontos no infinito. Esta limitação obriga os algoritmos geométricos a tratar separadamente muitos casos particulares.

Por exemplo, suponha que um algoritmo precisa calcular a intersecção de duas retas. Na geometria cartesiana, temos que considerar três casos diferentes: as retas coincidem (têm infinitos pontos em comum), são paralelas (não têm nenhum ponto em comum), ou estão em posição genérica (têm exatamente um ponto em comum).

Ou, então, suponha que queremos calcular a intersecção de dois ou mais semi-planos dados. O resultado "ordinário" dessa operação é um polígono. Na geometria cartesiana, entretanto, temos que considerar vários casos "excepcionais": a intersecção pode ser uma faixa limitada por duas retas paralelas, ou uma região infinita limitada por duas semi-retas e alguns segmentos, etc. Sem o conceito de pontos no infinito, precisamos inventar representações especiais, e portanto algoritmos especiais, para tratar cada um desses casos.

Outras desvantagens

Poderíamos enumerar outras desvantagens da geometria cartesiana, como por exemplo a falta de uma correspondência perfeita (dualidade) entre pontos e retas, ou a inconveniência da representação de transformações geométricas. Entretanto, estas limitações só poderão ser devidamente apreciadas depois que tivermos estudado a representação por coordenadas homogêneas, que não é afetada por elas.

2.2 Coordenadas homogêneas

Coordenadas homogêneas

Por definição, se (X, Y) são as coordenadas cartesianas de um ponto de $\mathbb{R}^2$, as *coordenadas homogêneas* desse ponto são uma tripla de números reais $[w, x, y]$, tais que $X = x/w$ e $Y = y/w$.

Peso

A coordenada w é chamada de *peso*, e por enquanto vamos supor que ela é estritamente positiva. Repare na notação: usaremos sempre parênteses $(*, *)$ para coordenadas cartesianas, e colchetes $[*, *, *]$ para coordenadas homogêneas.

A definição acima implica que um mesmo ponto de $\mathbb{R}^2$ pode ser representado por muitas triplas de coordenadas homogêneas. Assim, por exemplo, $[1, 2, 5]$, $[2, 4, 10]$, e $[0.03, 0.06, 0.15]$ são todos o mesmo ponto, cujas coordenadas cartesianas são $(2, 5)$.

Em geral, o par cartesiano (X, Y) corresponde a todas as triplas homogêneas $[w, wX, wY]$ com $w > 0$; em particular, a $[1, X, Y]$.

Observe que as coordenadas homogêneas w, x, y de um ponto não tem significado individual; apenas as razões x/w e y/w tem sentido.

Ex. 2.1: Traduza os seguintes pontos de coordenadas cartesianas para homogêneas:

(a) $(0, 0)$ (b) $(1, 0)$

(c) $(0, 1)$ (d) $(5, 6)$

Ex. 2.2: Traduza os seguintes pontos de coordenadas homogêneas para cartesianas:

(a) $[1, 0, 0]$ (b) $[1, 1, 0]$ (c) $[1, 0, 1]$

(d) $[1, 2, 3]$ (e) $[2, 5, 6]$ (f) $[2, 0, 0]$

Ex. 2.3: Escreva o ponto $(1/2, 3/5)$ em coordenadas homogêneas *inteiras*.

Ex. 2.4: O que acontece com o ponto $[w, x, y]$ quando x e y permanecem constantes, e o peso w tende para 0? E quando w tende para $+\infty$?

Ex. 2.5: Descreva a trajetória do ponto $[1 + t^2, 1 - t^2, 2t]$ quando o parâmetro t varia de $-\infty$ a $+\infty$.

2.2.1 Pontos infinitos

Observe que quando o peso w tende para zero, com x e y fixos, o ponto $[w, x, y]$ tende a se afastar infinitamente da origem, na direção do vetor (x, y).

Ponto infinito

É natural portanto considerar uma tripla homogênea $[0, x, y]$, com peso nulo, como sendo um ponto infinitamente distante da origem — um *ponto infinito* — na direção do vetor (x, y). Denotaremos esse ponto por $\infty(x, y)$.

O comprimento do vetor (x, y) é irrelevante, desde que não seja zero; isto é, as coordenadas $[0, \alpha x, \alpha y]$ representam o mesmo ponto infinito, para todo $\alpha > 0$. (Note que esta é a mesma equivalência de coordenadas homogêneas que vale para pontos ordinários.) Por outro lado, o sentido do vetor é importante; isto é, as triplas $[0, x, y]$ e $[0, -x, -y]$ representam pontos infinitos distintos.[1] Diremos que estes dois pontos infinitos são *antipodais*, e que um é o *antípoda* do outro.

Coordenadas inválidas

A tripla $[0, 0, 0]$ é um caso especial. A experiência mostra que não vale a pena tentar interpretá-la como um ponto; é melhor decretar que essa tripla é inválida.

> **Ex. 2.6:** Escreva as coordenadas homogêneas do ponto infinito cuja direção faz um ângulo de θ radianos com o eixo das abscissas.

2.2.2 O outro lado do plano

Se as triplas homogêneas $[w, x, y]$ com peso w positivo são pontos de $\mathbb{R}^{\nvDash}$, e as com peso nulo são pontos infinito, que significado podemos dar para as triplas com peso negativo?

Por estranho que pareça, é melhor interpretar essas triplas como pontos que, tendo passado "além do infinito", foram parar no "outro lado" do plano.

Aquém e além

Informalmente vamos imaginar que o plano é uma folha infinita de papel translúcido; e que, para cada posição (X, Y), existem dois pontos do plano, sobrepostos mas distintos: um na frente da folha, e outro no verso. Com já foi dito, uma tripla homogênea $[w, x, y]$ com $w \neq 0$ descreve um ponto de coordenadas cartesianas $(x/w, y/w)$; sendo que o ponto está na frente da folha se $w > 0$, e no verso se $w < 0$. Diremos que o primeiro está no *aquém* e segundo no *além*.

Pontos coincidentes e antipodais

Diremos também que esses dois pontos são *coincidentes* mas não iguais, e que um é o *antípoda* do outro. Em geral, denotaremos o antípoda de um ponto p por $\neg p$: ou seja, $\neg[w, x, y] = [-w, -x, -y]$. Esta definição vale tanto para pontos no aquém, no além, ou no infinito; em particular, $\neg[0, x, y] = [0, -x, -y]$.

Note que continua válida a regra que $[w, x, y] = [\alpha w, \alpha x, \alpha y]$, para todo $\alpha > 0$.

> **Ex. 2.7:** Para cada um dos pontos seguintes, dê as coordenadas cartesianas, e diga se o ponto está no aquém ou no além:

[1] Esta distinção pode parecer estranha aos leitores que já estão familiarizados com geometria projetiva. Este assunto será discutido na seção 2.10.

(a) $[1, -2, 3]$ (b) $[1, 2, -3]$ (c) $[-1, 2, 3]$

(d) $[-1, -2, -3]$ (e) $[1, 0, 0]$ (f) $[-1, 0, 0]$

Ex. 2.8: Dê coordenadas homogêneas para os pontos do aquém e do além cujas coordenadas cartesianas são:

(a) $(0, 0)$ (b) $(2, 3)$

(c) $(-2, -3)$ (d) $(-2, 3)$

Ex. 2.9: Qual a relação geométrica entre os pontos $[w, x, y]$ e $[-w, x, y]$? (Suponha $w > 0$.)

Ex. 2.10: Sejam w x, e y números reais não nulos. Considere as oito triplas homogêneas $[\alpha w, \beta x, \gamma y]$, onde $\alpha, \beta, \gamma \in \{+1, -1\}$. Quantos pontos distintos estão representados por essas triplas? Quais pares de pontos são antipodais?

2.3 O plano projetivo orientado $\mathbb{T}^{\nvDash}$

Plano projetivo orientado

Ao usarmos coordenadas homogêneas em vez de coordenadas cartesianas, estamos trocando o plano cartesiano $\mathbb{R}^{\nvDash}$ por um espaço geométrico estritamente maior, que que chamaremos de *plano projetivo orientado*, e que denotaremos por $\mathbb{T}^{\nvDash}$. (O índice 2 aqui indica a dimensão do espaço.)

Algebricamente, o conjunto de pontos do espaço $\mathbb{T}^{\nvDash}$ consiste de todas as triplas de números reais $[w, x, y]$, exceto a tripla $[0, 0, 0]$; sendo que duas triplas são consideradas equivalentes se e somente se uma for um múltiplo positivo da outra.

2.3.1 O modelo plano de $\mathbb{T}^{\nvDash}$

Modelo plano

Geometricamente, o espaço $\mathbb{T}^{\nvDash}$ pode ser definido pelo seu *modelo plano*, que consiste de duas cópias do plano cartesiano $\mathbb{R}^{\nvDash}$ (o aquém e o além), mais uma cópia do círculo unitário $\mathbb{S}^{\nVdash}$ (os pontos infinitos). Veja a figura 2.1.

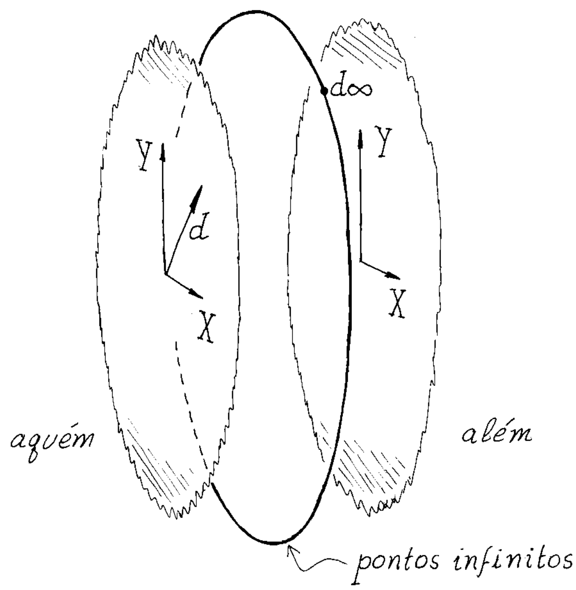


Figura 2.1: O plano projetivo de dois lados.

Lembremos que o ponto $[w, x, y]$ está no aquém se $w > 0$, e no além se $w < 0$; em ambos os casos, suas coordenadas cartesianas são $(x/w, y/w)$. O ponto é infinito se $w = 0$; nesse caso, sua direção é a do vetor (x, y).

Convenções gráficas

Em geral, as ilustrações de figuras geométricas de $\mathbb{T}^{\nvDash}$ que se seguem serão baseadas neste modelo plano. Pontos com mesmas coordenadas cartesianas serão desenhados sobrepostos; para distinguir os dois lados do plano, usaremos pontos cheios ($\bullet$), linhas cheias, e áreas hachuradas para o aquém, e pontos vazios ($\circ$), linhas tracejadas, e áreas pontilhadas para o além. Veja a figura 2.2.

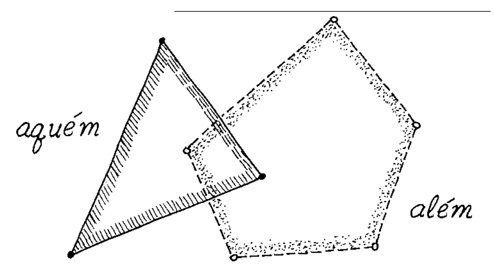


Figura 2.2: Convenções gráficas para os dois lados do plano.

Ex. 2.11: Desenhe os dois triângulos cujos vértices são dados a seguir, segundo as convenções da figura 2.2. Em cada caso, suponha que o triângulo está inteiramente contido num dos lados do plano (aquém ou além).

(a) $[1, 0, 0]$ $[1, 2, 0]$ $[2, 3, 5]$

(b) $[-1, 1, 0]$ $[-1, 2, 2]$ $[-1, 2, -3]$

2.3.2 O modelo esférico de $\mathbb{T}^{\nvDash}$

Modelo esférico

Outra maneira de visualizar o espaço $\mathbb{T}^{\nvDash}$ é através de seu *modelo esférico*, que consiste na superfície $\mathbb{S}^{\nvDash}$ da esfera unitária de $\mathbb{R}^{\nvDash}$ com centro na origem.

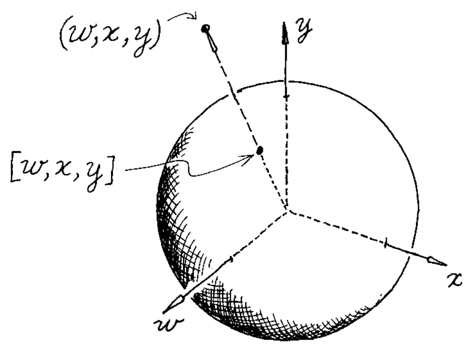


Figura 2.3: O modelo esférico de $\mathbb{T}^{\nvDash}$.

Especificamente, no modelo esférico o ponto de $\mathbb{T}^{\nvDash}$ com coordenadas homogêneas $[w, x, y]$ é representado pelo ponto de $\mathbb{S}^{\nvDash}$ com coordenadas cartesianas

$$\frac{(w, x, y)}{\sqrt{w^2 + x^2 + y^2}}$$

Ou seja, $[w, x, y]$ é representado pela projeção do ponto (w, x, y) de $\mathbb{R}^3$ sobre a esfera, na direção do centro da mesma. Veja a figura 2.3.

Aquém e além na esfera

O aquém e o além de $\mathbb{T}^2$ correspondem portanto aos hemisférios de $\mathbb{S}^2$ com $w > 0$ e $w < 0$, respectivamente. A origem $(0, 0)$ do aquém é o ponto $(1, 0, 0)$ da esfera; e a do além é $(-1, 0, 0)$. Os pontos no infinito de $\mathbb{T}^2$ estão no círculo onde a esfera é cortada pelo plano $w = 0$ de $\mathbb{R}^3$.

Observe que, neste modelo, um ponto p e seu antípoda $\neg p$ estão sempre diametralmente opostos na esfera.

Ex. 2.12: Indique graficamente a posição dos seguintes pontos no modelo esférico de $\mathbb{T}^2$..

(a)	$[1, 0, 0]$	(b)	$[-1, 0, 0]$	(c)	$[1, 1, 0]$
(d)	$[2, 3, 5]$	(e)	$[2, -3, -5]$	(f)	$[-2, 3, 5]$
(g)	$[0, 3, 5]$	(h)	$[0, -3, -5]$	(i)	$[0, 3, -5]$

2.3.3 Correspondência entre os modelos

.

A correspondência entre o modelo esférico e o modelo plano, definida implicitamente pelas coordenadas homogêneas, equivale à projeção de cada hemisfério de $\mathbb{S}^2$ ($w > 0$ e $w < 0$) na cópia correspondente de $\mathbb{R}^2$ (aquém ou além).

Em ambos os casos, devemos imaginar a cópia de $\mathbb{R}^2$ como um plano tangente à esfera $\mathbb{S}^2$, com sua origem no ponto $(1, 0, 0)$ da mesma, e com seus eixos paralelos aos eixos x e y de $\mathbb{R}^3$.

No caso do aquém, projetamos o hemisfério $w > 0$ de $\mathbb{S}^2$ no plano, a partir do centro da esfera. Veja a figura 2.4(a).

No caso do além, projetamos o hemisfério $w < 0$ sobre o plano *através* do centro da esfera. Veja a figura 2.4(b). Observe como esta projeção faz com que o além do modelo esférico fique rodado de 180° em relação ao além do modelo plano.

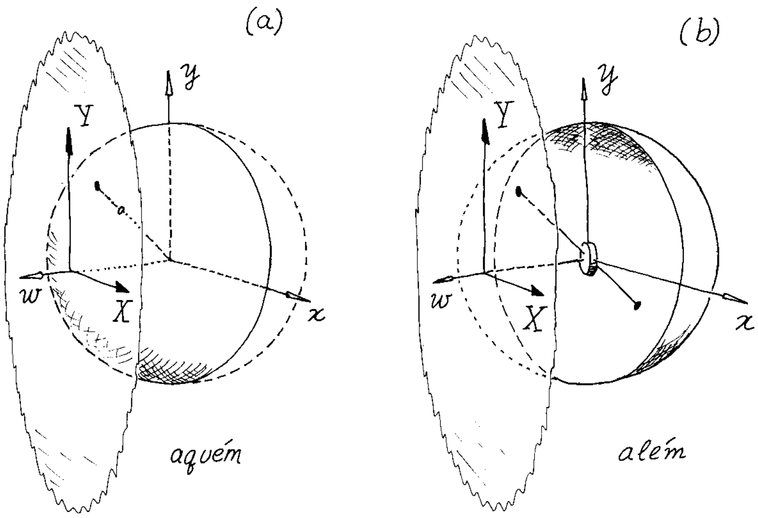


Figura 2.4: Correspondência entre os modelos plano e esférico de $\mathbb{T}^{\not=}$.

Finalmente, lembremos que os pontos infinitos (com $w = 0$) são representados em ambos os modelos por uma cópia do círculo unitário:

$$\Big\{ (x, y) \ : \ x^2 + y^2 = 1 \Big\} \qquad \text{(modelo plano)}$$
$$\Big\{ (0, x, y) \ : \ x^2 + y^2 = 1 \Big\} \qquad \text{(modelo esférico)}$$

A correspondência entre estes dois círculos é a óbvia.

2.4 Retas

2.4.1 Equação homogênea da reta

Como sabemos, uma linha reta do plano é definida por três coeficientes A, B, C, tais que um ponto genérico p de coordenadas cartesianas (X, Y) está nessa linha se e somente $AX+BY+C = 0$. Traduzindo essa equação para coordenadas homogêneas, conclui-se que um ponto finito $p = [w, x, y]$ está nessa linha se e somente se $A(x/w)+B(y/w)+C = 0$, isto é, $Ax + By + Cw = 0$.

Coeficientes homogêneos da reta

A notação fica mais elegante se rebatizarmos os coeficientes A, B, e C de $\mathcal{X}$, $\mathcal{Y}$, e $\mathcal{W}$, e colocarmos $\mathcal{W}$ em primeiro lugar. Desta forma, podemos dizer que uma linha é definida por três *coeficientes homogêneos*

$\langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$, sendo que o ponto genérico $p = [w, x, y]$ está nessa linha se e somente $\mathcal{W}w + \mathcal{X}x + \mathcal{Y}y = 0$.

Por exemplo, a linha $\langle 1, 2, 3 \rangle$ passa por todos os pontos $[w, x, y]$ tais que $w + 2x + 3y = 0$; ou, em termos cartesianos, todos os pontos (X, Y) tais que $2X + 3Y + 1 = 0$.

Ex. 2.13: Qual é a equação cartesiana da reta com coeficientes homogêneos $\langle 2, 3, 5 \rangle$?

Ex. 2.14: Quais são os coeficientes homogêneos da reta cuja equação cartesiana é $3X - 2Y = 6$?

Ex. 2.15: Quais são os coeficientes homogêneos dos eixos cartesianos X e Y?

Ex. 2.16: Determine as condições algébricas sobre os coeficientes $\langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$ que caracterizam:

(a) retas horizontais;
(b) retas verticais;
(c) retas que passam pela origem.

Reta inválida

Observe que a "reta" com coeficientes $\langle 0, 0, 0 \rangle$ é bastante peculiar, pois ela passa por todos os pontos do plano! Para evitar maiores problemas, é melhor decretar que essa tripla de coeficientes é inválida, e não representa nenhuma reta.

2.4.2 Os pontos no infinito de uma reta

Pontos infinitos de uma reta

Note que a equação homogênea da reta $\mathcal{W}w + \mathcal{X}x + \mathcal{Y}y = 0$ não é satisfeita apenas por pontos finitos, mas também pelos pontos infinitos $[0, \mathcal{Y}, -\mathcal{X}]$ e $[0, -\mathcal{Y}, \mathcal{X}]$. Estes são justamente os pontos infinitos nas duas direções paralelas à reta. Ou seja, toda reta paralela a um vetor $d = (x, y)$ contém os pontos infinitos $\infty d = [0, x, y]$ e $\infty(-d) = [0, -x, -y]$.

Ex. 2.17: Determine os dois pontos infinitos da reta $\langle 2, 3, 5 \rangle$.

Retas paralelas

Note que duas retas de $\mathbb{R}^{\nvDash}$ são paralelas entre si se e somente as retas correspondentes de $\mathbb{T}^{\nvDash}$ passam pelos mesmos pontos no infinito.

2.4.3 Retas no além

Retas no modelo plano

Observe que a equação $\mathcal{W}w+\mathcal{X}x+\mathcal{Y}y = 0$, que define quando um ponto pertence a uma reta, continua válida se negarmos as três coordenadas w, x, y do ponto simultaneamente. Portanto, se uma reta passa por um ponto p do aquém, ela também passa pelo seu antípoda $\neg p$ no além.

Ou seja, uma reta de $\mathbb{T}^{\nvDash}$ é representada no modelo plano por *duas* retas euclidianas superpostas, uma no aquém e uma no além, com os mesmos coeficientes cartesianos; mais dois pontos infinitos, nas duas direções paralelas às retas. Veja a figura 2.5.

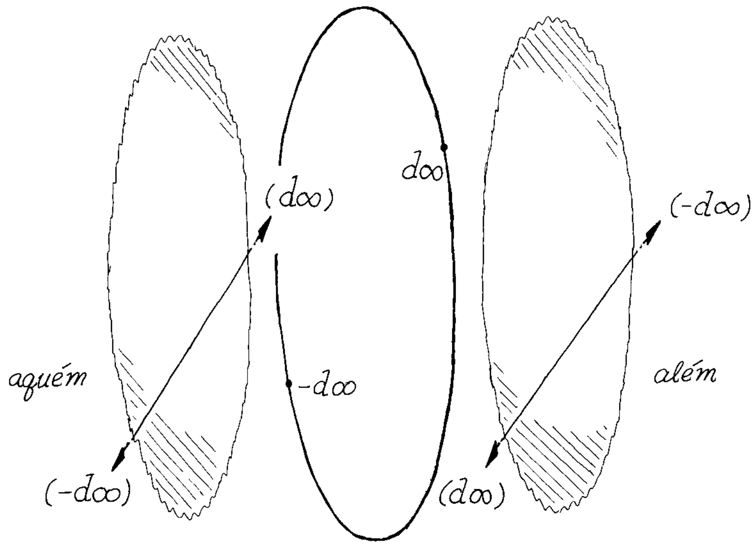


Figura 2.5: Uma reta, no aquém e no além.

2.4.4 Retas no modelo esférico

Lembremos que o vetor unitário (w, x, y) do modelo esférico representa o ponto $[w, x, y]$ de $\mathbb{T}^{\nvDash}$. Portanto, os pontos de $\mathbb{T}^{\nvDash}$ que estão na reta $\langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$ correspondem a pontos de $\mathbb{S}^{\nvDash}$ que satisfazem a satisfazem a equação $\mathcal{W}w + \mathcal{X}x + \mathcal{Y}y = 0$.

Retas no modelo esférico

Esta equação define um plano de $\mathbb{R}^{\nVdash}$ que passa pela origem, e portanto corta a esfera num círculo de raio máximo. Ou seja, uma reta de $\mathbb{T}^{\nvDash}$ corresponde a um círculo máximo de $\mathbb{S}^{\nvDash}$; e é fácil ver que a recíproca também vale. Veja a figura 2.6.

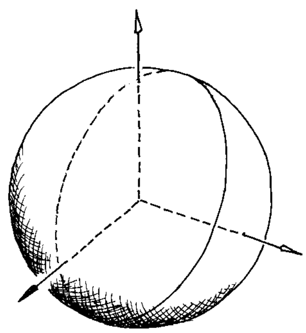


Figura 2.6: Uma reta no modelo esférico.

Ex. 2.18: Indique graficamente a posição das seguintes retas, no modelo esférico:

(a)	$\langle 0, 1, 0 \rangle$	(b)	$\langle 0, 0, 1 \rangle$	(c)	$\langle 0, 1, 1 \rangle$
(d)	$\langle 1, 2, 3 \rangle$	(e)	$\langle 1, 3, 2 \rangle$	(f)	$\langle -1, 2, 3 \rangle$
(g)	$\langle 1, -1, -1 \rangle$	(h)	$\langle 2, -1, -1 \rangle$	(i)	$\langle -10, 1, 1 \rangle$

2.4.5 Os dois lados de uma reta

Lado positivo e negativo

Os pontos $[w, x, y]$ que não estão sobre uma reta $r = \langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$ podem ser divididos em dois conjuntos, os *lados* da reta, conforme o sinal da expressão $\mathcal{W}w + \mathcal{X}x + \mathcal{Y}y$. Este teste define o lado *positivo* e o lado *negativo* da reta r.

No modelo esférico, os dois lados da reta são os dois hemisférios em que a esfera $\mathbb{S}^2$ fica dividida pelo plano de equação $\mathcal{W}w + \mathcal{X}x + \mathcal{Y}y = 0$. Veja a figura 2.7.

Ex. 2.19: No modelo esférico, indique o lado positivo de cada uma das retas do exercício 2.18.

Observe que o sinal de $\mathcal{W}w + \mathcal{X}x + \mathcal{Y}y$ se inverte se negarmos as três coordenadas homogêneas w, x, y; isto é, um ponto está no lado positivo de uma reta se e somente se seu antípoda está no lado negativo.

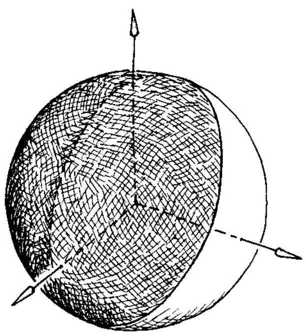


Figura 2.7: Os dois lados de uma reta, no modelo esférico.

Lados no modelo plano

Pode-se concluir daí que, no modelo plano de $\mathbb{T}^{\neq}$, o lado positivo de uma reta r consiste de um semi-plano do aquém, limitado por r, e do *outro* semi-plano do além, que não é antípoda do primeiro. Veja a figura 2.8.

O lado positivo da reta r também inclui todos os pontos no infinito num arco de 180° limitado pelas duas direções paralelas a r. Os dois outros semi-planos, e o o arco complementar no infinito, formam o lado negativo de r.

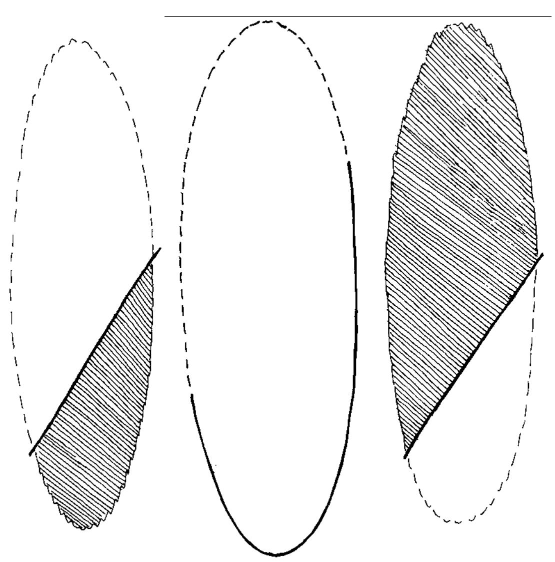


Figura 2.8: Os dois lados de uma reta, no modelo plano.

Ex. 2.20: Descreva, no modelo plano, o lado positivo de cada uma destas retas:

(a) $\langle 0, 1, 0 \rangle$ (b) $\langle 0, 0, 2 \rangle$

(c) $\langle 0, -1, 0 \rangle$ (d) $\langle 2, 3, -4 \rangle$

Ex. 2.21: Qual a condição algébrica para que a origem do aquém esteja no lado positivo da reta $\langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$?

2.4.6 O teste de ponto contra reta

Notação $r \diamond p$

A operação de determinar o lado de uma reta dada que contém um ponto dado é fundamental para muitos algoritmos geométricos. Portanto, vale a pena introduzir a notação

$$r \diamond p = \operatorname{sgn}(\mathcal{W}w + \mathcal{X}x + \mathcal{Y}y) \tag{2.1}$$

onde $p = [w, x, y]$, $r = \langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$, e $\operatorname{sgn} t$ é o sinal de t — isto é, -1 se $t < 0$, 0 se $t = 0$, e $+1$ se $t > 0$. Note que $r \diamond (\neg p) = -(r \diamond p)$.

Ex. 2.22: Determine $r \diamond p$ para cada um dos casos abaixo:

(a) $r = \langle 2, 3, 5 \rangle$, $p = [1, 1, -1]$

(b) $r = \langle 2, 3, 5 \rangle$, $p = [1, 1, 0]$

(c) $r = \langle 2, 3, 5 \rangle$, $p = [1, 1, 1]$

(d) $r = \langle 1, 0, 0 \rangle$, $p = [1, 0, 0]$

(e) $r = \langle 1, 0, 0 \rangle$, $p = [-1, 0, 0]$

(f) $r = \langle a, b, c \rangle$, $p = [a, b, c]$

2.4.7 Retas opostas

Note que a função $r \diamond p$ não se altera se multiplicarmos os coeficientes de r por um número α positivo, pois isto equivale a multiplicar a expressão $\mathcal{W}w + \mathcal{X}x + \mathcal{Y}y$ por α. Entretanto, se multiplicarmos os coeficientes de r por um número negativo, o valor de $r \diamond p$ fica negado; isto é, os lados positivo e negativo da reta se invertem.

Portanto, para que a função $\diamond$ tenha significado bem definido, precisamos distinguir as retas $r = \langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$ e $r' = \langle -\mathcal{W}, -\mathcal{X}, -\mathcal{Y} \rangle$. Apesar de ambas serem *coincidentes* (passarem exatamente pelos mesmos pontos) elas não são *iguais*, pois diferem na sua *orientação* (a rotulação dos seus dois lados). Diremos que essas retas são *opostas* uma da outra, e denotaremos essa relação por $r' = \neg r$.

Retas opostas e coincidentes

Ou seja, a tripla de coeficientes $\langle \alpha\mathcal{W}, \alpha\mathcal{X}, \alpha\mathcal{Y} \rangle$ denota a mesma reta que $\langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$ para todo α positivo, e a reta oposta para todo $\alpha < 0$.

2.4.8 A reta no infinito

Na geometria cartesiana, a equação $AX + BY + C = 0$ só define uma reta se pelo menos um dos coeficientes A e B for diferente de zero. Quando $A = B = 0$, a equação não é satisfeita por nenhum ponto de $\mathbb{R}^{\not\models}$. (A menos que C também seja zero, caso em que todo ponto do plano satisfaz a equação.)

Portanto, os coeficientes homogêneos $\langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$ denotam uma reta do plano euclidiano só se $\mathcal{X} \neq 0$ ou $\mathcal{Y} \neq 0$. O que fazemos então com as triplas da forma $\langle \mathcal{W}, 0, 0 \rangle$? Será que, com a introdução de pontos no infinito, é possível atribuir algum significado geométrico a essas triplas?

Observe primeiro que, se adotarmos para estas triplas a mesma regra de equivalência que vale para as triplas normais, concluímos que $\langle \mathcal{W}, 0, 0 \rangle$ é equivalente à tripla $\langle 1, 0, 0 \rangle$, ou a $\langle -1, 0, 0 \rangle$, dependendo do sinal de $\mathcal{W}$. Ou seja, existem apenas duas "retas" da forma $\langle \mathcal{W}, 0, 0 \rangle$, opostas entre si.

Se $\langle 1, 0, 0 \rangle$ é uma reta, quais são seus pontos? Segundo a definição, um ponto $[w, x, y]$ está em $\langle 1, 0, 0 \rangle$ se e somente se $1 \cdot w + 0 \cdot x + 0 \cdot y = 0$; isto é, $w = 0$. Ou seja, a reta $\langle 1, 0, 0 \rangle$ contém todos os pontos infinitos, e apenas esses pontos. Portanto, diremos que $\langle 1, 0, 0 \rangle$ e $\langle -1, 0, 0 \rangle$ são as duas *retas no infinito*, que denotaremos por Ω e $\neg\Omega$; e chamaremos as outras retas de *ordinárias.*

Retas no infinito: Ω e $\neg\Omega$

Retas ordinárias

Ex. 2.23: Qual é o lado positivo da reta Ω? E o de $\neg\Omega$?

Em resumo: algebricamente, o conjunto de retas de $\mathbb{T}^{\not\models}$ é o conjunto de todas as triplas reais $\langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$, exceto a tripla $\langle 0, 0, 0 \rangle$; sendo que duas triplas são consideradas equivalentes se e somente se uma for um múltiplo positivo da outra. Um ponto $[w, x, y]$ está numa reta $\langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$ se e somente se $\mathcal{W}w + \mathcal{X}x + \mathcal{Y}y$ é zero. Caso contrário, o ponto está no lado positivo ou negativo da reta, conforme o sinal desta fórmula.

2.4.9 Incidência de retas e pontos

Note que, apesar dos pontos e retas no infinito parecerem especiais no modelo plano, no modelo esférico eles são perfeitamente equivalentes aos pontos finitos e retas ordinárias. Como veremos, esta uniformidade — *homogeneidade* — dos elementos de $\mathbb{T}^2$ se reflete na manipulação algébrica de suas coordenadas homogêneas.

Intersecção de duas retas

Reta por dois pontos

Uma manifestação dessa homogeneidade é o fato que, no plano $\mathbb{T}^2$, *duas retas não-coincidentes se interceptam em dois pontos antipodais.* Simetricamente, em $\mathbb{T}^2$ *dois pontos não-coincidentes determinam exatamente duas retas, opostas entre si.* Estas propriedades valem para todos os tipos de pontos — finitos ou infinitos, no aquém ou no além — e para todos os tipos de retas — ordinárias ou no infinito, paralelas ou não.

Note a semelhança entre estas propriedades e os dois principais axiomas da geometria euclidiana; "por dois pontos distintos passa uma única reta", e "duas retas distintas e não paralelas se encontram num único ponto". As diferenças mais óbvias entre as duas formulações são quase que uma questão de nomenclatura — cada cada ponto da geometria euclidiana é desdobrado em dois pontos de $\mathbb{T}^2$, o mesmo acontecendo com as retas. Uma diferença mais significativa é que, graças aos pontos no infinito, o caso das retas paralelas não é mais excepcional.

2.4.10 Topologia de $\mathbb{T}^2$

Informalmente, definir a topologia de um espaço matemático S consiste em definir os limites de seqüências infinitas de pontos de S; ou, o que dá no mesmo, em definir a noção de continuidade para curvas de S (funções de $\mathbb{R}$ para S).

Seqüência convergente de pontos

Por definição, a topologia de $\mathbb{T}^{\not\models}$ é a determinada implicitamente pelo seu modelo esférico. Ou seja, uma seqüência de pontos $p_i = [w_i, x_i, y_i]$ converge para um ponto $p = [w, x, y]$ se e somente se os vetores unitários

$$(w_i, x_i, y_i)/\sqrt{w_i^2 + x_i^2 + y_i^2}$$

de $\mathbb{R}^{\not\models}$ convergirem para o vetor $(w, x, y)/\sqrt{w^2 + x^2 + y^2}$.

Ex. 2.24: Para cada uma das seqüências a seguir, determine os pontos limite, quando i tende para infinito:

(a) $p_i = [1,\, i,\, i^2]$ (b) $p_i = [1,\, -i,\, i^2]$

(c) $p_i = [1,\, i,\, \sin i]$ (d) $p_i = [1/i,\, 1,\, 2]$

(e) $p_i = [1/i,\, 1/(i+1),\, 1/(i+3)]$ (f) $p_i = [i^3,\, i^4,\, i^2]$

Ex. 2.25: Quais das seqüências abaixo converge, quando i tende para infinito:

(a) $p_i = [(-1)^i,\, 1,\, i]$

(b) $p_i = [1,\, 1,\, i(-1)^i]$

(c) $p_i = [(-1)^i,\, 1,\, i(-1)^i]$

Funções contínuas

Este conceito de convergência nos permite definir a noção de continuidade para funções cujo domínio e/ou contra-domínio são subconjuntos de $\mathbb{T}^{\not\models}$. Por exemplo, uma função $c(t) = [w(t), x(t), z(t)]$ de algum intervalo $I \subseteq \mathbb{R}$ para $\mathbb{T}^{\not\models}$ é contínua se e somente se a função

$$\bar{c}(t) = (w(t), x(t), y(t))/\sqrt{w(t)^2 + x(t)^2 + y(t)^2}$$

de I para $\mathbb{S}^{\not\models}$ for contínua.

Ex. 2.26: Prove que uma função $c(t)$ de algum intervalo $I \subseteq \mathbb{R}$ para $\mathbb{T}^{\not\models}$ é contínua se e somente se existirem três funções contínuas $w(t)$, $x(t)$, e $y(t)$, que não se anulam todas para um mesmo $t \in I$, tais que $c(t) = [w(t), x(t), z(t)]$.

Ex. 2.27: Quais das fórmulas abaixo definem funções contínuas do intervalo aberto $(0, 1)$ para $\mathbb{T}^{\not\models}$? Quais delas podem ser estendidas para o intervalo $[0, 1]$, mantendo a continuidade?

(a) $c(t) = [1,\, t,\, t^2]$
(b) $c(t) = [t,\, t^2,\, t^3]$
(c) $c(t) = [t,\, t^2,\, \sin t]$
(d) $c(t) = [t - 1/2,\, 1,\, 2]$
(e) $c(t) = [t,\, 1,\, \sin(1/t)]$

Topologia do modelo plano

No modelo plano, a topologia de $\mathbb{T}^2$ é relativamente complicada. Pode-se verificar sem muita dificuldade que cada um dos lados do plano (aquém ou além) tem de fato a mesma topologia do plano cartesiano $\mathbb{R}^2$; e os pontos no infinito têm a topologia do círculo $\mathbb{S}^1$.

A complexidade toda está na a maneira como estas três partes estão ligadas entre si. Considere uma seqüência de pontos p_i com coordenadas cartesianas (X_i, Y_i), tais que as distâncias $|p_i| = \sqrt{X_i^2 + Y_i^2}$ dos pontos à origem tende para o infinito, enquanto que os vetores unitários $d_i = (X_i, Y_i)/|p_i|$ convergem para algum vetor limite $d = (X, Y)$. Nesse caso, por definição, a seqüência p_i converge para o ponto no infinito $[0, X, Y]$, se os pontos p_i estiverem todos no aquém; e para $[0, -X, -Y]$, se os pontos p_i estiverem todos no além.

Ex. 2.28: Considere a curva $c(t) = [1 - 2t, 2t, 4t^2 - 1]$, onde t varia entre 0 e 1. Determine o ponto onde essa curva está no infinito, e o valor de t correspondente. Desenhe a trajetória dessa curva, no modelo esférico e no modelo plano.

2.5 Fórmulas geométricas

De modo geral, qualquer fórmula da geometria analítica plana pode ser adaptada para coordenadas homogêneas, bastando substituir na mesma as coordenadas cartesianas X e Y por x/w e y/w, respectivamente. (Se a fórmula descreve as coordenadas de um ponto, precisamos também trocar os parênteses da mesma por colchetes, e acrescentar uma coordenada 1 inicial (o peso).)

2.5.1 Ponto médio

Ponto médio de um segmento

Por exemplo, em geometria cartesiana, sabemos que o ponto médio m do segmento com extremos $p_0 = (X_0, Y_0)$ e $p_2 = (X_2, Y_2)$ é dado por

$$m = \left(\frac{X_0 + X_2}{2}, \ \frac{Y_0 + Y_2}{2} \right)$$

Portanto, em termos de coordenadas homogêneas, se os extremos forem $p_0 = [w_0, x_0, y_0]$ e $p_2 = [w_2, x_2, y_2]$, esta fórmula equivale a

$$\begin{aligned} m &= \left[\ 1, \ \frac{\frac{x_0}{w_0} + \frac{x_2}{w_2}}{2}, \ \frac{\frac{y_0}{w_0} + \frac{y_2}{w_2}}{2} \ \right] \\ &= \left[\ 1, \ \frac{x_0 w_2 + x_2 w_0}{2 w_0 w_2}, \ \frac{y_0 w_2 + y_2 w_0}{2 w_0 w_2} \ \right] \end{aligned}$$

Podemos eliminar as divisões desta fórmula, multiplicando as três coordenadas homogêneas por $w_0 w_2$, o que não altera o ponto m. Obtemos assim a fórmula

$$m = [\ 2 w_0 w_2, \ \ w_2 x_0 + w_0 x_2, \ \ w_2 y_0 + w_0 y_2 \ ] \tag{2.2}$$

Ex. 2.29: Dê uma fórmula homogênea, sem divisões, para o centro de gravidade de um triângulo com vértices $p_i = [w_i, x_i, y_i]$, $i = 0, 1, 2$.

Ex. 2.30: O que acontece com o ponto médio de p_0 e p_2, conforme definido pela fórmula (2.2), se *um* dos pontos estiver no infinito? E se *ambos* estiverem no infinito?

Ex. 2.31: Qual a posição do ponto definido pela fórmula (2.2), se um dos pontos estiver no aquém, e o outro no além? E se ambos estiverem no além?

Ex. 2.32: Uma fórmula para o ponto médio que dá resultados mais consistentes para pontos do além é

$$m = [\ |w_2| w_0 + |w_0| w_2, \ \ |w_2| x_0 + |w_0| x_2, \ \ |w_2| y_0 + |w_0| y_2 \ ]$$

Compare o resultado desta fórmula com o de (2.2) para os 6 casos: p_0 e p_2 ambos no aquém, ambos no além, um no aquém e um no além, um no aquém e outro infinito, um no além e outro infinito, e ambos no infinito.

Ex. 2.33: Calcule as coordenadas homogêneas do ponto do aquém que está a $1/3$ do caminho entre os pontos $[1, 2, 3]$ e $[2, 3, 5]$.

Ex. 2.34: Determine uma fórmula homogênea, sem divisões, para o ponto m que divide o segmento $p_0\,p_1$ em duas partes cujos comprimentos estão na razão $\lambda_0 : \lambda_1$. Suponha que p_0, p_1, e m estão no aquém, e $\lambda_0 + \lambda_1 > 0$.

2.5.2 Colinearidade de três pontos

Pontos colineares

Dizemos que três pontos são *colineares* se eles pertencem a uma mesma reta. O que isto significa em termos de coordenadas homogêneas?

Para começar, vamos supor que os três pontos estão no aquém. Em geometria cartesiana, prova-se que três pontos $p_0 = (X_0, Y_0)$, $p_1 = (X_1, Y_1)$, e $p_2 = (X_2, Y_2)$ são colineares se e somente se

$$\begin{vmatrix} 1 & X_0 & Y_0 \\ 1 & X_1 & Y_1 \\ 1 & X_2 & Y_2 \end{vmatrix} = 0$$

Em termos das coordenadas homogêneas $[w_i, x_i, y_i]$ dos três pontos, esta fórmula equivale a

$$\begin{vmatrix} 1 & x_0/w_0 & y_0/w_0 \\ 1 & x_1/w_1 & y_1/w_1 \\ 1 & x_2/w_2 & y_2/w_2 \end{vmatrix} = 0 \tag{2.3}$$

Podemos multiplicar as três linhas desta matriz por w_0, w_1, e w_2, respectivamente, pois isto apenas multiplica o determinante pelo número positivo $w_0 w_1 w_2$, o que não afeta a equação. Concluímos que os três pontos do aquém são colineares se e somente se

$$\begin{vmatrix} w_0 & x_0 & y_0 \\ w_1 & x_1 & y_1 \\ w_2 & x_2 & y_2 \end{vmatrix} = 0 \tag{2.4}$$

Na verdade, a fórmula (2.4) vale para *quaisquer* tres pontos de $\mathbb{T}^{\nvDash}$, finitos ou infinitos, no aquém ou no além. Este resultado pode ser demonstrado sem muita dificuldade a partir do modelo esférico de $\mathbb{T}^{\nvDash}$, ou da definição algébrica de reta.

Ex. 2.35: Usando a fórmula (2.4), determine quais destas triplas de pontos são colineares:

(a) $[1, 0, 0]$, $[1, 1, 0]$, $[1, 0, 1]$.

(b) $[1, 0, 0]$, $[1, 1, 0]$, $[1, 2, 0]$.

(c) $[1, 0, 1]$, $[1, 2, 6]$, $[1, 3, 8]$.

(d) $[1, 2, 3]$, $[2, 2, 3]$, $[5, 2, 3]$.

Ex. 2.36: Demonstre *algebricamente*, usando a fórmula 2.4, que os pontos p_0, p_1, e p_2 são colineares em cada um dos seguintes casos:

(a) $p_0 = p_1$.

(b) p_2 é o ponto médio de p_0 e p_1.

(c) p_1, p_2 e p_0 são colineares.

Ex. 2.37: Considere três pontos móveis p_0, p_1, p_2 no plano, cada qual se deslocando em linha reta com velocidade uniforme. Ou seja, as coordenadas cartesianas de p_i, num instante t quaisquer, são $(X_i, Y_i) + t(X_i', Y_i')$, onde X_i, Y_i, X_i' e Y_i' são constantes.

(a) Mostre como calcular o instante t^* em que esses três pontos estarão alinhados.

(b) Analise o número de soluções t^* distintas admitidas pelo problema do item (a), discutindo todos os casos que podem ocorrer.

2.5.3 Reta determinada por dois pontos

Como já observamos, dois pontos não coincidentes de $\mathbb{T}^{\not\preceq}$ determinam duas retas que passam por eles, coincidentes mas com orientações opostas.

Fórmula da reta por dois pontos

Podemos deduzir a fórmula para os coeficientes destas retas a partir da equação (2.4). De acordo com esta última, a condição para que um ponto genérico $[w, x, y]$ seja colinear com $p_0 = [w_0, x_0, y_0]$ e $p_1 = [w_1, x_1, y_1]$ é

$$\begin{vmatrix} w_0 & x_0 & y_0 \\ w_1 & x_1 & y_1 \\ w & x & y \end{vmatrix} = 0$$

Expandindo este determinante pela última linha, obtemos

$$+\begin{vmatrix} x_0 & y_0 \\ x_1 & y_1 \end{vmatrix} w - \begin{vmatrix} w_0 & y_0 \\ w_1 & y_1 \end{vmatrix} x + \begin{vmatrix} w_0 & x_0 \\ w_1 & x_1 \end{vmatrix} y = 0$$

Daqui se deduz que uma das duas retas que passam por p_0 e p_1 é dada pela fórmula

$$\begin{aligned} p_0 \vee p_1 &= \left\langle\ +\begin{vmatrix} x_0 & y_0 \\ x_1 & y_1 \end{vmatrix},\ -\begin{vmatrix} w_0 & y_0 \\ w_1 & y_1 \end{vmatrix},\ +\begin{vmatrix} w_0 & x_0 \\ w_1 & x_1 \end{vmatrix}\ \right\rangle \qquad (2.5) \\ &= \langle\ +x_0 y_1 - x_1 y_0,\ -w_0 y_1 + w_1 y_0,\ +w_0 x_1 - w_1 x_0\ \rangle \end{aligned}$$

Obviamente, a outra reta que passa por p_0 e p_1 é

$$\neg(p_0 \vee p_1) = \langle\ -x_0 y_1 + x_1 y_0,\ +w_0 y_1 - w_1 y_0,\ -w_0 x_1 + w_1 x_0\ \rangle$$

A orientação da reta $p_0 \vee p_1$, implícita na fórmula (2.5), tem um significado geométrico bastante simples, que será visto na seção 2.7.5.

Ex. 2.38: Determine as duas retas que passam pelos pontos $[1, 2, 3]$ e $[4, 5, 6]$.

Ex. 2.39: Determine a reta que passa pelo ponto do eixo X com abscissa X_0, e pelo ponto do eixo Y com ordenada Y_0.

Ex. 2.40: Determine a reta que passa pela origem e pelo ponto (X, Y).

Ex. 2.41: O que acontece com a fórmula (2.5), quando os pontos p_0 e p_1 coincidem?

2.5.4 Reta por pontos no infinito

Pode-se verificar que a fórmula (2.5) vale para quaisquer dois pontos — no aquém, no além, ou infinitos — que não sejam iguais ou antipodais.

Reta dado ponto e direção

Esta flexibilidade é útil, por exemplo, quando queremos calcular os coeficientes da reta r que passa por um ponto finito $p = [w_p, x_p, y_p]$ e é paralela a um certo vetor $d = (x_d, y_d)$. Esta reta contém o ponto infinito $\infty d = [0, x_d, y_d]$; ou seja, $r = p \vee (\infty d)$. Portanto, não precisamos desenvolver (e programar) uma fórmula especial para este problema; basta usar a fórmula (2.5), da reta $p_0 \vee p_1$, com $p_0 = p$ e $p_1 = \infty d$.

Ex. 2.42: Determine os coeficientes da reta que passa por $[2, 3, 4]$ e é paralela ao vetor $(-2, 3)$.

Ex. 2.43: Determine a fórmula geral explícita para os coeficientes da reta que passa pelo ponto cartesiano (X, Y) e faz um ângulo anti-horário de θ radianos com o eixo X.

Ex. 2.44: Diga como calcular os coeficientes da reta que passa por um ponto finito $p = [w_p, x_p, y_p]$ e é *perpendicular* a um vetor cartesiano $d = (x_d, y_d)$.

Ex. 2.45: O que acontece com a reta $p_0 \vee p_1$ (fórmula (2.5)), quando os pontos p_0 e p_1 são ambos infinitos?

2.5.5 Ponto determinado por duas retas

Na geometria euclidiana, aprendemos que duas retas $r_0 = \langle \mathcal{W}_0, \mathcal{X}_0, \mathcal{Y}_0 \rangle$ e $r_1 = \langle \mathcal{W}_1, \mathcal{X}_1, \mathcal{Y}_1 \rangle$, distintas e não paralelas, se cruzam num único ponto, que vamos denotar por $r_0 \wedge r_1$.

As coordenadas cartesianas (X, Y) desse ponto devem satisfazer as duas equações $\mathcal{W}_i + \mathcal{X}_i X + \mathcal{Y}_i Y = 0$; e portanto podem ser calculadas resolvendo-se o sistema linear

$$\begin{cases} \mathcal{X}_0 X + \mathcal{Y}_0 Y &= -\mathcal{W}_0 \\ \mathcal{X}_1 X + \mathcal{Y}_1 Y &= -\mathcal{W}_1 \end{cases}$$

Segundo a regra de Cramer, a solução deste sistema é

$$X = -\frac{\begin{vmatrix} \mathcal{W}_0 & \mathcal{Y}_0 \\ \mathcal{W}_1 & \mathcal{Y}_1 \end{vmatrix}}{\begin{vmatrix} \mathcal{X}_0 & \mathcal{Y}_0 \\ \mathcal{X}_1 & \mathcal{Y}_1 \end{vmatrix}}, \qquad Y = -\frac{\begin{vmatrix} \mathcal{X}_0 & \mathcal{W}_0 \\ \mathcal{X}_1 & \mathcal{W}_1 \end{vmatrix}}{\begin{vmatrix} \mathcal{X}_0 & \mathcal{Y}_0 \\ \mathcal{X}_1 & \mathcal{Y}_1 \end{vmatrix}} \tag{2.6}$$

Convertendo o par (X, Y) para coordenadas homogêneas, e multiplicando as mesmas pelo denominador comum, concluímos que o ponto

Fórmula para o ponto de encontro de duas retas de intersecção das duas retas é dado pela fórmula

$$r_0 \wedge r_1 = \left[+\begin{vmatrix} \mathcal{X}_0 & \mathcal{Y}_0 \\ \mathcal{X}_1 & \mathcal{Y}_1 \end{vmatrix}, -\begin{vmatrix} \mathcal{W}_0 & \mathcal{Y}_0 \\ \mathcal{W}_1 & \mathcal{Y}_1 \end{vmatrix}, +\begin{vmatrix} \mathcal{W}_0 & \mathcal{X}_0 \\ \mathcal{W}_1 & \mathcal{X}_1 \end{vmatrix} \right] \tag{2.7}$$

$$= \begin{bmatrix} +\mathcal{X}_0\mathcal{Y}_1 - \mathcal{X}_1\mathcal{Y}_0, \\ -\mathcal{W}_0\mathcal{Y}_1 + \mathcal{W}_1\mathcal{Y}_0, \\ +\mathcal{W}_0\mathcal{X}_1 - \mathcal{W}_1\mathcal{X}_0 \end{bmatrix} \tag{2.8}$$

Ex. 2.46: Usando a fórmula 2.8, determine o ponto de encontro $r \wedge s$ para os seguintes pares de retas:

(a) $r = \langle 0, 1, 0\rangle$ $s = \langle 0, 0, 1\rangle$

(b) $r = \langle 0, 0, 1\rangle$ $s = \langle 0, 1, 0\rangle$

(c) $r = \langle 1, 3, 5\rangle$ $s = \langle 2, 4, 6\rangle$

(d) $r = \langle 1, 3, 5\rangle$ $s = \langle 1, 0, 0\rangle$

(d) $r = \langle 1, 3, 5\rangle$ $s = \langle 2, 3, 5\rangle$

Apesar da fórmula (2.8) ter sido desenvolvida sob a hipótese do ponto de encontro ser finito, pode-se verificar que ela vale para quaisquer duas retas não coincidentes, mesmo as que se encontram no infinito. Veja os exercícios 2.47, e 2.48.

Ex. 2.47: Verifique algebricamente que o ponto $r \wedge s$, calculado pela fórmula (2.8), sempre pertence às retas r e s.

Ex. 2.48: Prove que duas retas r e s, ordinárias e não coincidentes, são paralelas se e somente se $r \wedge s$ é um ponto infinito.

Ex. 2.49: Determine o ponto de encontro das retas que passam pelos pontos de abscissa $+1$ e -1 do eixo X, e formam ângulos α e β com o mesmo, respectivamente.

Ex. 2.50: O que acontece com a fórmula (2.8), quando as retas r_0 e r_1 coincidem (são iguais ou opostas)?

Já observamos antes que, no plano $\mathbb{T}^{\not\leq}$, duas retas não coincidentes se encontram em *dois* pontos antipodais. A fórmula (2.8) dá apenas um desses pontos. Qual deles? Os exercícios 2.51 e 2.52 respondem em parte a essa pergunta; a resposta completa será vista na seção 2.7.7.

Ex. 2.51: A partir da fórmula (2.8), mostre que

$$(\neg r_0) \wedge r_1 = r_0 \wedge (\neg r_1) = \neg(r_0 \wedge r_1)$$

Ex. 2.52: A partir da fórmula (2.8), mostre que $r_1 \wedge r_0 = \neg(r_0 \wedge r_1)$

2.5.6 Intersecção de retas paralelas

Na geometria euclidiana (ou cartesiana) clássica, duas retas paralelas nunca se encontram. Com a inclusão dos pontos infinitos, essa propriedade não é mais válida. Pelo contrário, duas retas paralelas se encontram em *dois* pontos infinitos distintos: os pontos $\infty(\pm d)$, onde d é qualquer vetor paralelo às duas retas. Na verdade, podemos usar este fato para definir retas paralelas na geometria projetiva.

Intersecção de retas paralelas

Pode-se verificar que a fórmula (2.8), que dá a intersecção $r_0 \wedge r_1$ de duas retas r_0 e r_1, funciona mesmo quando as duas retas são paralelas, e dá como resultado um dos dois pontos infinitos comuns a ambas. Portanto, quando queremos calcular a intersecção de duas retas, não precisamos nos preocupar com a possibilidade das mesmas serem paralelas.

Ex. 2.53: Usando a fórmula (2.8), determine as coordenadas do ponto de intersecção da reta genérica $\langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$ com a reta Ω. Qual o significado geométrico desse ponto?

2.5.7 Dualidade

Dualidade entre $\vee$ e $\wedge$

Leitores atentos provavelmente notaram a semelhança entre as fórmulas da reta que passa por dois pontos (2.5) e do ponto de intersecção de duas retas (2.8).

Dualidade entre pontos e retas

Esta semelhança é uma manifestação de um princípio muito importante, a *dualidade* entre pontos e retas do plano projetivo. Considere a função que ao ponto $p = [w, x, y]$ associa a reta $p^* = \langle w, x, y \rangle$; e à reta $r = \langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$ associa o ponto $r^* = [\mathcal{W}, \mathcal{X}, \mathcal{Y}]$. É fácil ver que um ponto p está numa reta r se e somente se a reta p^* passa pelo ponto r^*; e, na verdade,

$$r \diamond p = p^* \diamond r^*$$

Com um pouco mais de trabalho, pode-se concluir que

$$(p_0 \vee p_1)^* = (p_0)^* \wedge (p_1)^*$$

para quaisquer dois pontos p_0, p_1. Como $(p^*)^* = p$, temos também

$$(m_0 \wedge m_1)^* = (m_0)^* \vee (m_1)^*$$

para quaisquer duas linhas m_0, m_1.

A função '*' é a *dualidade canônica* de $\mathbb{T}^{\nvDash}$. Ela nos permite traduzir mecanicamente muitas fórmulas geométricas que envolvem pontos em outras fórmulas que envolvem linhas, e vice-versa. O mesmo vale para teoremas, algoritmos, e estruturas de dados. Com isto, o custo de desenvolvimento e programação de algoritmos geométricos fica substancialmente reduzido, quase que pela metade.

Ex. 2.54: Qual é o dual canônico dos seguintes objetos e conceitos:

(a) Um ponto do aquém.

(b) Um ponto no infinito.

(c) O antípoda de um ponto.

(d) A origem do aquém.

(e) O eixo X.

2.6 Segmentos e triângulos

2.6.1 Segmento

Combinação linear e convexa

Lembremos que uma *combinação linear* de vetores $v_1, v_2, .. v_n$ é uma soma da forma $\alpha_0 v_0 + \alpha_1 v_1 + \cdot\cdot \alpha_n v_n$, onde os α_i são números reais. Uma *combinação convexa* é uma combinação linear cujos coeficientes α_i são todos maiores ou iguais a zero.

Segmento que liga dois pontos

Sejam p_0 e p_1 dois pontos não-antipodais de $\mathbb{T}^{\nvDash}$. Por definição, o *segmento* (*fechado*) $p_0\,p_1$ consiste de todos os pontos cujas coordenadas homogêneas são combinações convexas não nulas das coordenadas homogêneas de p_0 e p_1.

Ou seja, se $p_0 = [w_0, x_0, y_0]$ e $p_1 = [w_1, x_1, y_1]$, o segmento $p_0\,p_1$ consiste de todos os pontos da forma

$$[\ \alpha_0\, w_0 + \alpha_1\, w_1,\quad \alpha_0\, x_0 + \alpha_1\, x_1,\quad \alpha_0\, y_0 + \alpha_1\, y_1\ ] \tag{2.9}$$

onde α_0 e α_1 são números reais não-negativos, e não ambos nulos.

Qual o significado geométrico da fórmula (2.9)? No modelo esférico de $\mathbb{T}^{\nvDash}$, é fácil ver que o segmento $p_0\,p_1$ é simplesmente o caminho mais curto de p_0 a p_1 sobre a esfera. Esse caminho é sempre um arco de círculo máximo, com menos de 180° de extensão. Veja a figura 2.9.

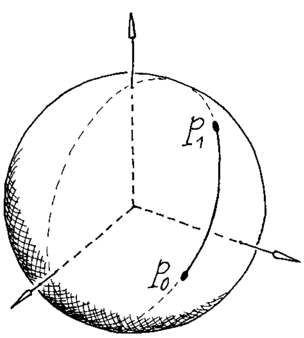


Figura 2.9: Um segmento no modelo esférico.

No modelo plano, como sempre, precisamos distinguir vários casos. Em primeiro lugar, se p_0 e p_1 estão ambos no aquém, pode-se verificar que a fórmula define simplesmente o segmento euclidiano do aquém ligando esses dois pontos. Veja a figura 2.10(a).

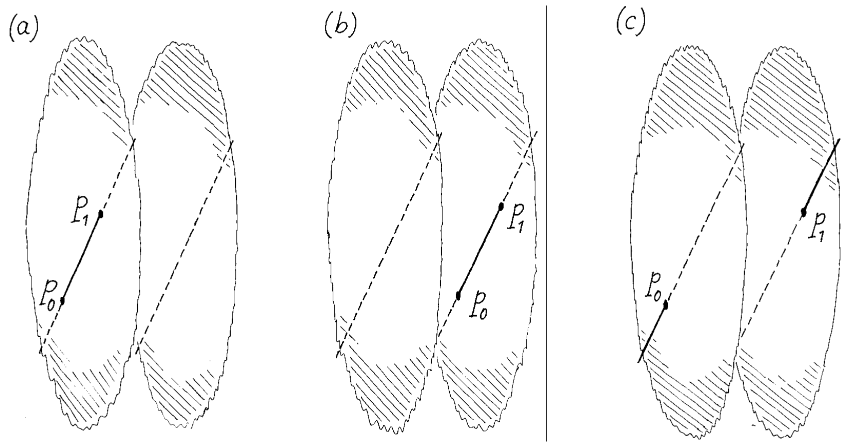


Figura 2.10: Segmentos no modelo plano.

Por outro lado, a fórmula implica que um ponto p está no segmento $p_0\,p_1$ se e somente se o antípoda $\neg p$ está no segmento $(\neg p_0)(\neg p_1)$. Concluímos daí que, se ambos os pontos estão no além, o segmento $p_0\,p_1$ é simplesmente o segmento euclidiano do além que liga os dois pontos. Veja a figura 2.10(b).

Quando p_0 está no aquém, e p_1 no além, a situação fica um pouco mais complicada. Nesse caso, o segmento $p_0\,p_1$ consiste de duas semi-retas: uma no aquém, saindo de p_0 na direção oposta a $\neg p_1$; e uma no além, saindo de p_1 na direção oposta a $\neg p_0$. Veja a figura 2.10(c).

Se um dos pontos é infinito, o segmento $p_0\,p_1$ é uma semi-reta com origem no ponto finito e apontando para o infinito. Veja a figura 2.11. Finalmente, se ambos os pontos são infinitos, o segmento é um conjunto de pontos infinitos, cobrindo um arco de direções menor que 180° entre as direções de p_0 e p_1.

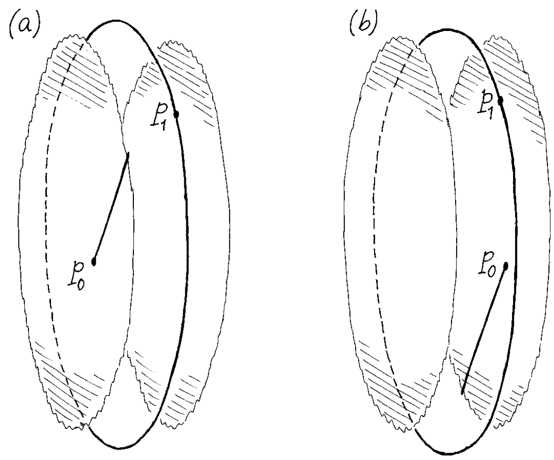


Figura 2.11: Segmentos com um extremo infinito.

Ex. 2.55: Desenhe os seguintes segmentos do plano projetivo orientado, usando as convenções gráficas da figura 2.2:

(a) $[1,1,1]$ – $[1,-1,-1]$.
(b) $[+1,0,0]$ – $[0,1,0]$.
(c) $[-1,0,0]$ – $[0,1,0]$.
(d) $[1,0,0]$ – $[0,-1,0]$.
(e) $[1,1,1]$ – $[-1,1,1]$.
(f) $[0,1,0]$ – $[0,0,1]$.

2.6.2 Consistência

Observe que o ponto descrito pela fórmula (2.9) depende não só de p_0, p_1, α_0, e α_1, mas também da escolha dos pesos de p_0 e p_1. Isto é, se multiplicarmos as coordenadas homogêneas de p_0 por um fator positivo β, o ponto descrito pela fórmula mudará de posição, pois isto equivale a multiplicar α_0 por β.

Portanto, os pontos individuais gerados por (2.9) não tem significado geométrico; apenas o *conjunto* de todos esses pontos — ou seja, o segmento $p_0\,p_1$ — é um conceito bem definido.

É importante notar que o segmento $p_0\,p_1$ não está definido se (e somente se) p_0 e p_1 são antipodais. Nesse caso, existem infinitas retas que passam por p_0 e p_1. Observe que este é o único caso em que a fórmula (2.9) pode dar origem à tripla inválida $[0, 0, 0]$.

Ex. 2.56: Demonstre algebricamente que três pontos de $\mathbb{T}^2$ são colineares se e somente se dois deles são antipodais, ou se um deles está no segmento que liga os outros dois, ou se o antípoda de um deles está no segmento que liga os outros dois.

Ex. 2.57: Qual é o dual canônico do segmento $p_0\,p_1$?

2.6.3 Corte de segmentos

Uma das operações mais comuns em geometria computacional e computação gráfica é determinar ponto onde um segmento $p_0\,p_1$ cruza uma determinada linha m.

Teste se um segmento cruza uma reta

Em primeiro lugar, para saber se o segmento de fato cruza a reta, basta verificar se seus extremos estão do mesmo lado da mesma. Especificamente, se $p_0 = [w_0, x_0, y_0]$, $p_1 = [w_1, x_1, y_1]$, e $m = \langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle$, então basta calcular os números reais

$$\begin{aligned} \alpha_0 &= \mathcal{W}w_0 + \mathcal{X}x_0 + \mathcal{Y}y_0 \\ \alpha_1 &= \mathcal{W}w_1 + \mathcal{X}x_1 + \mathcal{Y}y_1 \end{aligned}$$

O segmento $p_0\,p_1$ cruza (ou toca) a linha m se e somente se s_0 e s_1 tem sinais opostos, ou um deles é nulo; isto é, se $s_0\,s_1 \leq 0$.

Coordenadas da intersecção

Se esta condição é satisfeita, o ponto de intersecção pode ser determinado da seguinte maneira.

Como vimos na seção 2.6.1, um ponto genérico do segmento $p_0\,p_1$ tem coordenadas

$$[\ \alpha_0\,w_0 + \alpha_1\,w_1,\ \alpha_0\,x_0 + \alpha_1\,x_1,\ \alpha_0\,y_0 + \alpha_1\,y_1\ ] \tag{2.10}$$

para $\alpha_0 \geq 0$, $\alpha_1 \geq 0$, $(\alpha_0, \alpha_1) \neq (0,0)$. Este ponto está na reta m se e somente se

$$\mathcal{W}(\alpha_0\,w_0 + \alpha_1\,w_1) + \mathcal{X}(\alpha_0\,x_0 + \alpha_1\,x_1) + \mathcal{Y}(\alpha_0\,y_0 + \alpha_1\,y_1) = 0$$

ou seja

$$\alpha_0(\mathcal{W}w_0 + \mathcal{X}x_0 + \mathcal{Y}y_0) + \alpha_1(\mathcal{W}w_1 + \mathcal{X}x_1 + \mathcal{Y}y_1) = 0$$

ou, ainda,

$$\alpha_0\,s_0 + \alpha_1\,s_1 = 0$$

Portanto, para que o ponto (2.10) esteja na linha m, basta tomar $\alpha_0 = \pm s_1$ e $\alpha_1 = \mp s_0$, escolhendo-se os sinais de modo que α_0 e α_1 sejam não-negativos. Ou seja, o ponto de intersecção é

$$[\ |s_1|\,w_0 + |s_0|\,w_1,\ \ |s_1|\,x_0 + |s_0|\,x_1,\ \ |s_1|\,y_0 + |s_0|\,y_1\ \ ] \tag{2.11}$$

Note que esta fórmula devolve a tripla inválida $[0,0,0]$ se s_0 e s_1 forem ambos nulos; isto é, se os dois extremos do segmento estiverem sobre a reta m.

Ex. 2.58: Em cada um dos casos abaixo, determine a intersecção do segmento $p_0\,p_1$ com a reta r.

(a) $p_0 = [1,0,0]$ $\quad p_1 = [1,2,3]$ $\quad r = \langle 1,-1,-1\rangle$

(b) $p_0 = [1,1,1]$ $\quad p_1 = [-1,1,1]$ $\quad r = \langle 3,-1,-1\rangle$

(c) $p_0 = [1,1,1]$ $\quad p_1 = [-1,1,1]$ $\quad r = \langle 1,0,0\rangle$

Ex. 2.59: Descreva um método para determinar se dois segmentos $p_0\,p_1$ e $q_0\,q_1$ se interceptam, e calcular o ponto de intersecção, a partir das coordenadas homogêneas dos extremos.

2.6.4 Triângulos

Triângulo

Por definição, o *triângulo* determinado por três pontos de $\mathbb{T}^{\not\vDash}$ é o conjunto de todos os pontos cujas coordenadas são combinações convexas não nulas das coordenadas desses pontos.

Ou seja, se $p_i = [w_i, x_i, y_i]$, para $i \in \{0, 1, 2\}$, o triângulo $p_0\, p_1\, p_2$ consiste de todos os pontos da forma

$$\begin{aligned} \Big[\quad & \alpha_0\, w_0 + \alpha_1\, w_1 + \alpha_2\, w_2, \\ & \alpha_0\, x_0 + \alpha_1\, x_1 + \alpha_2\, x_2, \\ & \alpha_0\, y_0 + \alpha_1\, y_1 + \alpha_2\, y_2 \quad \Big] \end{aligned} \tag{2.12}$$

onde α_0, α_1, e α_2 são números reais, não negativos e não todos nulos. a Veja a figura 2.12

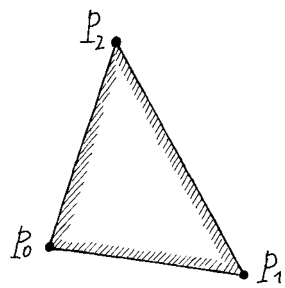


Figura 2.12: Um triângulo no aquém.

Interior, exterior, e fronteira

Note que este conjunto inclui os três pontos p_0, p_1, e p_2 (os *vértices*), bem como os segmentos $p_0\, p_1$, $p_1\, p_2$, e $p_2\, p_0$ (os *lados*). A união destes três segmentos é a *fronteira* do triângulo, uma curva fechada que separa os demais pontos do triângulo (o *interior*) do restante do plano (o *exterior*).

Triângulo próprio e degenerado

Diremos que um triângulo é *degenerado* se seu interior for vazio, e *próprio* caso contrário.

Ex. 2.60: Prove que o triângulo $p_0\, p_1\, p_2$ é a união de todos os segmentos da forma $p_0\, q$, onde q está no segmento $p_1\, p_2$.

Ex. 2.61: Prove que um triângulo é degenerado se e somente se um dos vértices pertence ao segmento que liga os outros dois.

A fórmula (2.12) pode devolver a tripla inválida $[0, 0, 0]$. Isto acontece se e somente se dois vértices são antipodais, ou se o antípoda de um dos vértices está no segmento que liga os outros dois. Nestes casos o triângulo é "indefinido por definição".

Triângulo no modelo esférico

No modelo esférico, o triângulo $p_0\,p_1\,p_2$ é um *triângulo esférico* — uma região limitada por três arcos de círculo máximo, com extremos nesses pontos. Veja a figura 2.13.

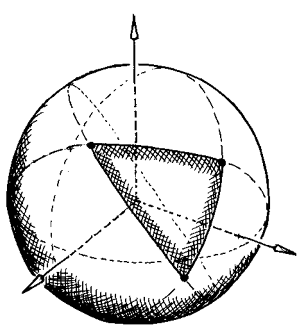


Figura 2.13: Um triângulo no modelo esférico.

Triângulo no modelo plano

No modelo plano, se os vértices estão todos no aquém, ou todos no além, os pontos definidos pela fórmula (2.12) estarão todos no mesmo lado do plano, e constituem simplesmente o triângulo euclidiano com os vértices dados.

Caso contrário, o triângulo tem uma forma mais complicada, que se estende de um lado para outro do plano, através de pontos no infinito. Por exemplo, a figura 2.14 ilustra o triângulo com vértices $a = [1, 1, 0]$, $b = [1, 0, 2]$, e $c = [-1, 1, 1]$.

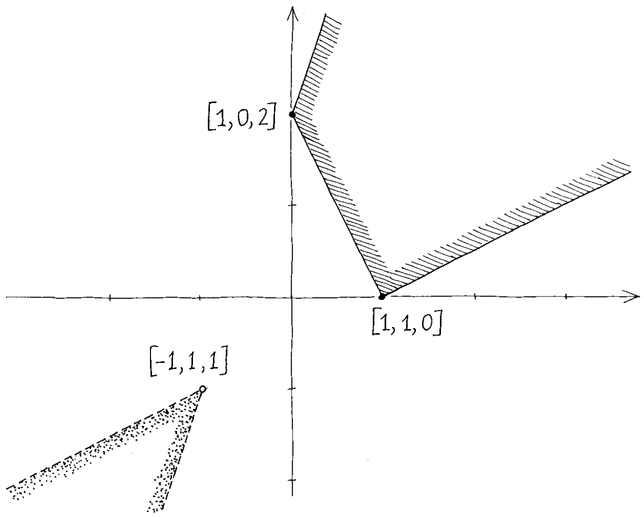


Figura 2.14: Um triângulo no modelo plano.

Ex. 2.62: Em cada um dos casos abaixo, desenhe o triângulo $p_0\,p_1\,p_2$, indicando seu interior com a convenção usual (hachurado no aquém, pontilhado no além):

(a) $p_0 = [1,0,0]$ $p_1 = [1,1,0]$ $p_2 = [1,0,1]$.

(b) $p_0 = [1,0,0]$ $p_1 = [0,1,0]$ $p_2 = [0,0,1]$.

(c) $p_0 = [1,1,0]$ $p_1 = [1,0,1]$ $p_2 = [-1,1,1]$.

(d) $p_0 = [1,1,0]$ $p_1 = [0,0,1]$ $p_2 = [-1,1,0]$.

(d) $p_0 = [1,0,0]$ $p_1 = [1,1,0]$ $p_2 = [0,1,0]$.

2.7 Orientação

2.7.1 Orientação de três pontos

Orientação de três pontos

Sejam p, q, e r três pontos não colineares contido no aquém. Informalmente, a *orientação* da tripla (p, q, r) é o sentido (horário ou anti-horário) em que o segmento segmento $p\,u$ roda em torno de p, quando o ponto u vai de q para r ao longo do segmento $q\,r$.

Observe que a ordem dos pontos é importante. Por exemplo, a tripla $(\,(1,1),\ (3,2),\ (2,4)\,)$ tem orientação anti-horária enquanto que $(\,(1,1),\ (2,4),\ (3,2)\,)$ tem orientação horária.

Sentido positivo e negativo

Os termos *horário* e *anti-horário* pressupõem a convenção usual que o eixo Y do plano cartesiano é desenhado a $90°$ do eixo X, no sentido anti-horário. Para não dependermos dessa convenção puramente gráfica, chamaremos os dois sentidos de rotação de *negativo* e *positivo*; sendo que, por definição, a tripla $(\,(0,0),\ (1,0),\ (0,1)\,)$ tem orientação positiva.

Mais formalmente, definimos a função $\Delta(p, q, r)$, que vale -1 ou $+1$, conforme a tripla (p, q, r) tenha orientação negativa ou positiva, respectivamente; e que vale 0 se os três pontos são colineares.

Ex. 2.63: Determine o valor de $\Delta(p, q, r)$ para cada uma das triplas abaixo:

(a) $p = [1, 0, 0]$ $q = [1, 1, 0]$ $r = [1, 0, 1]$

(b) $p = [1, 0, 0]$ $q = [1, 3, 2]$ $r = [2, 5, 3]$

(c) $p = [1, 2, 3]$ $q = [1, 4, 4]$ $r = [2, 8, 9]$

Na seção 2.6.4 definimos o triângulo $p\,q\,r$ como sendo um conjunto de pontos, incluindo os vértices, lados, e interior. Portanto, estritamente falando, o triângulo não depende da ordem dos seus vértices. Entretanto, abusando um pouco da linguagem, vamos freqüentemente escrever "orientação do triângulo $p\,q\,r$" em vez de "orientação da tripla (p, q, r)"

2.7.2 Orientação algébrica

Vimos na seção 2.5.2 que três pontos são colineares quando o determinante 3×3 de suas coordenadas homogêneas é zero.

Fórmula para a orientação

Verifica-se que, se esses três pontos estão no aquém, e não são colineares, o sinal desse mesmo determinante é precisamente a orientação do triângulo determinado por esses três pontos.

Ou seja, se $p_i = [w_i, x_i, y_i]$, para $i = in\{0, 1, 2\}$, temos

$$\Delta(p_0, p_1, p_2) \;=\; \mathrm{sgn}\begin{vmatrix} w_0 & x_0 & y_0 \\ w_1 & x_1 & y_1 \\ w_2 & x_2 & y_2 \end{vmatrix} \tag{2.13}$$

Em álgebra linear aprendemos que multiplicar uma linha de uma matriz por uma constante α também multiplica seu determinante por α. Decorre daí que o sinal do determinante acima não se altera se substituirmos as triplas homogêneas $[w_i, x_i, y_i]$ por quaisquer outras triplas equivalentes. Ou seja, o valor da função $\Delta(p_0, p_1, p_2)$ depende apenas das posições dos pontos, e não da escolha dos pesos w_i.

Orientação e ordem

Observe também que se trocarmos a ordem de quaisquer duas linhas da matriz, o sinal do determinante se inverte. Portanto, a orientação de um triângulo também se inverte quando trocamos quaisquer dois de seus vértices. Ou seja,

$$\Delta(q, p, r) = \Delta(p, r, q) = \Delta(r, q, p) = -\,\Delta(p, q, r) \tag{2.14}$$

Ex. 2.64: Determine *algebricamente* o valor de $\Delta(p, q, r)$, usando a fórmula (2.13), para cada um dos triângulos

(a) $p = [1, 0, 0]$ $q = [1, 1, 0]$ $r = [1, 0, 1]$

(b) $p = [1, 0, 0]$ $q = [1, 3, 2]$ $r = [2, 5, 3]$

(c) $p = [1, 2, 3]$ $q = [1, 4, 4]$ $r = [2, 8, 9]$

Ex. 2.65: Qual o efeito de uma permutação circular dos argumentos de Δ? Isto é, qual a relação entre $\Delta(p, q, r)$, $\Delta(q, r, p)$, e $\Delta(r, p, q)$?

Ex. 2.66: Demonstre que os triângulos pqr, pmr e mqr têm a mesma orientação, se m for o ponto médio do segmento pq.

2.7.3 Orientação no além

Até agora só definimos a função Δ para pontos do aquém. Como fica esse conceito quando alguns dos pontos estão no infinito, ou no além?

Como o conceito intuitivo de "sentido de rotação" não é muito claro nesses casos, vamos adotar a fórmula (2.13) como sendo a *definição* da função Δ, para quaisquer três pontos de $\mathbb{T}^{\nvDash}$.

Orientação nos antípodas

Observe que o determinante da fórmula (2.13) muda de sinal se multiplicarmos qualquer linha por -1. Portanto, se trocarmos qualquer vértice pelo seu antípoda, a orientação do triângulo se inverte. Ou seja,

$$\Delta(\neg p_0, p_1, p_2) = -\,\Delta(p_0, p_1, p_2) \tag{2.15}$$

Portanto,

$$\Delta(\neg p_0, \neg p_1, \neg p_2) = -\,\Delta(p_0, p_1, p_2) \tag{2.16}$$

Em particular, a orientação de um triângulo no além é oposta à do triângulo coincidente no aquém.

Ex. 2.67: Usando as equações (2.14) e (2.15), determine a orientação dos pontos abaixo, sabendo que $\Delta(p, q, r) = +1$:

(a) $\Delta(p, q, \neg r)$ (b) $\Delta(p, \neg q, \neg r)$

(c) $\Delta(p, \neg r, q)$ (d) $\Delta(q, r, \neg p)$

2.7.4 Orientação no modelo esférico

O significado geométrico da função Δ é relativamente fácil de visualizar no modelo esférico.

Em primeiro lugar, vamos definir informalmente o *sentido positivo de rotação* em torno de um ponto qualquer p de $\mathbb{S}^{\nvDash}$, como sendo o sentido anti-horário se visto de um ponto exterior à esfera mas arbitrariamente próximo de p.

Esta definição pressupõe que os eixos de $\mathbb{R}^{\nVdash}$ estão dispostos de maneira usual; ou seja que os vetores $(1, 0, 0)$, $(0, 1, 0)$, e $(0, 0, 1)$, nessa ordem, rodam em sentido anti-horário quando vistos do ponto $(1, 1, 1)$.

Orientação no modelo esférico

Com esta definição, podemos então verificar que $\Delta(p, q, r)$ indica o sentido em que o segmento $p\,u$ roda em torno de p, quando o ponto u vai de q para r ao longo do segmento $q\,r$. Veja a figura 2.15.

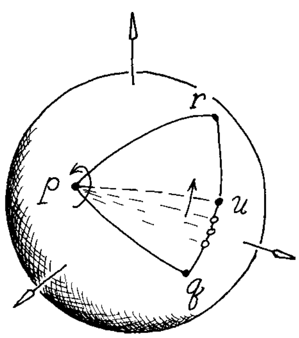


Figura 2.15: Orientação de um triângulo no modelo esférico.

2.7.5 Orientação da reta por dois pontos

Orientação da reta por dois pontos

Estamos agora em condições de esclarecer uma dúvida que restou da seção 2.5.3: qual é a orientação da reta $p_0 \vee p_1$, definida pela fórmula (2.5).

Para esse fim, vamos testar a posição de um ponto genérico $p_2 = [w_2, y_2, x_2]$ em relação à mesma:

$$(p_0 \vee p_1) \diamond p_2 = \operatorname{sgn} \left(+ \begin{vmatrix} x_0 & y_0 \\ x_1 & y_1 \end{vmatrix} w_2 - \begin{vmatrix} w_0 & y_0 \\ w_1 & y_1 \end{vmatrix} x_2 + \begin{vmatrix} w_0 & x_0 \\ w_1 & x_1 \end{vmatrix} y_2 \right)$$

Note que a fórmula dentro dos parênteses é simplesmente a expansão, pela última linha, do determinante da matriz de coordenadas dos três pontos p_0, p_1, p_2. Como vimos, o sinal desse determinante define a orientação do triângulo p_0, p_1, p_2.

Concluímos portanto que *o lado positivo da reta $p_0 \vee p_1$ consiste de todos os pontos p_2 que formam um triângulo positivo com p_0 e p_1.* Formalmente,

$$(p_0 \vee p_1) \diamond p_2 = \Delta(p_0, p_1, p_2) \tag{2.17}$$

Pode-se verificar que, se os eixos X e Y forem desenhados em sua posição convencional, de modo que o sentido positivo é anti-horário, então o lado positivo da reta $p_0 \vee p_1$ é o semi-plano que fica à esquerda da mesma, do ponto de vista de quem vai de p_0 a p_1 ao longo da reta.

Ex. 2.68: Mostre que a reta de p_1 a p_0 tem orientação contrária à reta de p_0 a p_1; isto é, $p_1 \vee p_0 = \neg(p_0 \vee p_1)$.

Ex. 2.69: Qual é a orientação da reta $p \vee (\infty d)$? (Isto é, qual é o lado positivo dessa reta?)

2.7.6 Orientação longitudinal de uma reta

Orientação longitudinal de uma reta

Usando o conceito de rotação positiva em torno de um ponto na esfera, podemos definir a *orientação longitudinal* de uma reta como sendo o sentido de percurso da mesma que circunda os pontos do seu hemisfério positivo no sentido positivo.

Ou seja, se p e q são dois pontos de uma reta r, tais que $p \vee q = r$, a orientação longitudinal de r é o sentido de percurso da mesma que corresponde a ir de p para q ao longo do segmento pq.

2.7.7 Cruzamento de retas e orientação

Podemos agora responder também a outra dúvida que ficou em aberto da seção 2.5.5: se duas retas r e s se cruzam em dois pontos antipodais p e q, qual destes dois é o ponto $r \wedge s$, definido pela fórmula (2.8)?

Lembremos que $r \wedge s$ só está definido se r e s não são coincidentes. Neste caso, existe uma única maneira de rodar a reta r em torno do eixo $p - -q$, por um ângulo menor que $180°$, que a torna igual a s, em posição e orientação.

Escolha de $r \wedge s$

É fácil ver que essa rotação vai ter sentidos opostos em relação a p e q. Pode-se provar que o ponto $r \wedge s$, calculado segundo a fórmula (2.8), é precisamente aquele ponto em que o sentido desta rotação é positivo.

Pode-se também verificar que $r \wedge s$ é o ponto onde r, percorrida no sentido da sua orientação longitudinal, passa do hemisfério positivo de s para o hemisfério negativo.

Ex. 2.70: Mostre que, se $\Delta(p, q, r) = +1$, então $(p \vee q) \wedge (q \vee r) = q$.

Ex. 2.71: Se r é uma reta ordinária, obviamente $r \wedge \Omega$ é um dos dois pontos infinitos de r. Qual deles? (Responda em termos geométricos, e não algébricos.)

2.8 Transformações projetivas

Transformações projetivas

Dentre todas as funções que levam pontos de $\mathbb{T}^{\nvDash}$ para pontos de $\mathbb{T}^{\nvDash}$, existe uma classe importante, as *transformações projetivas*, ou *projetividades*, que se caracterizam por preservar as relações de colinearidade: isto é, se três pontos estão alinhados, suas imagens também o são, e vice-versa.

O conceito de projetividade engloba muitas transformações geométricas importantes, como as translações, rotações, e mudanças de escala, que estudaremos a seguir.

2.8.1 Caracterização algébrica

Matriz de transformação

Pode-se provar que toda projetividade de $\mathbb{T}^{\nvDash}$ corresponde a uma transformação linear inversível das coordenadas homogêneas. Isto é, para toda projetividade F existe uma matriz real

$$\mathsf{F} = \begin{bmatrix} f_{ww} & f_{wx} & f_{wy} \\ f_{xw} & f_{xx} & f_{xy} \\ f_{yw} & f_{yx} & f_{yy} \end{bmatrix}$$

com dimensão 3×3 e determinante não nulo, tal que

$$\begin{aligned} F([w,x,y]) &= [w,x,y]\mathsf{F} \\ &= \begin{bmatrix} wf_{ww} + xf_{xw} + yf_{yw} \\ wf_{wx} + xf_{xx} + yf_{yx} \\ wf_{wy} + xf_{xy} + yf_{yy} \end{bmatrix} \end{aligned} \tag{2.18}$$

Note que, para fins desta fórmula, a tripla $[w,x,y]$ deve ser considerada um vetor linha, isto é, uma matriz 1×3.

A recíproca também é verdadeira: toda matriz 3×3 F, com determinante não nulo, define pela fórmula (2.18) uma projetividade de $\mathbb{T}^{\nvDash}$. Este fato é fácil de provar, usando a definição de colinearidade (equação (2.13)), e o fato que o determinante de um produto de matrizes é o produto dos seus determinantes. Veja o exercício 2.72.

Ex. 2.72: Seja F uma matriz 3×3. Prove que a função F de $\mathbb{T}^{\nvDash}$ para $\mathbb{T}^{\nvDash}$ definida pela fórmula (2.18) satisfaz

$$\Delta(F(p_0), F(p_1), F(p_2)) = \Delta(p_0, p_1, p_2) \cdot \operatorname{sgn} |\mathsf{F}|$$

Transformações positivas e negativas

O exercício 2.72 revela que as transformações projetivas se dividem em duas classes, as *positivas* e as *negativas*, conforme o sinal do determinante de sua matriz; sendo que uma projetividade positiva preserva as orientações de todos os triângulos, enquanto que uma projetividade negativa as inverte.

Note que, se aplicarmos a fórmula (2.18) a duas triplas homogêneas equivalentes, obteremos dois resultados equivalentes.

Note também que, se multiplicarmos todos os elementos da matriz F por um mesmo número $\alpha \neq 0$, a transformação projetiva F definida pela mesma não se altera. Portanto, duas matrizes F' e F'' determinam a mesma transformação se e somente se $\mathsf{F}' = \alpha\mathsf{F}''$ para algum $\alpha \neq 0$.

Ex. 2.73: Mostre que a função $F(p) = \neg p$, que leva cada ponto para seu antípoda, é uma projetividade negativa de $\mathbb{T}^{\nvDash}$.

Antes de estudar as propriedades gerais das projetividades, vamos conhecer alguns casos particulares, que são bastante importantes na prática.

2.8.2 Translações

Translação

Para deslocar uma figura no plano, mantendo-se sua orientação, basta somar às coordenadas cartesianas de cada um de seus pontos um mesmo vetor (X_0, Y_0). Isto é, basta aplicar a cada ponto da figura a função

$$(X, Y) \mapsto (X + X_0, Y + Y_0) = (X, Y) + (X_0, Y_0)$$

Uma função desta forma é chamada de *translação* do plano $\mathbb{R}^{\nvDash}$ pelo vetor (X_0, Y_0). Note que (X_0, Y_0) é também a imagem da origem $(0, 0)$.

Em termos de coordenadas homogêneas, a translação que leva a origem $[1, 0, 0]$ para o ponto $[w_0, x_0, y_0]$ (necessariamente finito) é dada pela fórmula

$$[w, x, y] \mapsto [\ ww_0, \quad xw_0 + wx_0, \quad yw_0 + wy_0\ ]$$

Note que as coordenadas do resultado são combinações lineares das coordenadas do argumento. Podemos portanto escrever a função acima como um produto da tripla homogênea $[w, x, y]$ (vista agora como um vetor linha do $\mathbb{R}^{\nvDash}$) por uma matriz 3×3:

$$[w, x, y] \mapsto [w, x, y] \begin{bmatrix} w_0 & x_0 & y_0 \\ 0 & w_0 & 0 \\ 0 & 0 & w_0 \end{bmatrix} \tag{2.19}$$

Note que um ponto infinito $p = [0, x, y]$ é levado para $[0, xw_0, yw_0] = [0, x, y] = p$. Ou seja, qualquer translação mantém a reta Ω fixa ponto-a-ponto. Segue-se que as translações também preservam todas as distâncias, direções, e ângulos de qualquer figura.

Ex. 2.74: Escreva a matriz da translação que desloca o ponto (X_0, Y_0) para o ponto (X_1, Y_1).

2.8.3 Rotações

Rotação Para rodar uma figura plana em torno da origem $(0, 0)$, por um ângulo θ em sentido anti-horário, basta aplicar a cada um de seus pontos a função

$$\begin{aligned} (X, Y) &\mapsto (X \cos\theta - Y \sin\theta, X \sin\theta + Y \cos\theta) \\ &= (X, Y) \begin{pmatrix} +\cos\theta & -\sin\theta \\ +\sin\theta & +\cos\theta \end{pmatrix} \end{aligned}$$

Em termos de coordenadas homogêneas, esta função é

$$[w, x, y] \mapsto [w, x, y] \begin{bmatrix} 1 & 0 & 0 \\ 0 & +\cos\theta & -\sin\theta \\ 0 & +\sin\theta & +\cos\theta \end{bmatrix} \tag{2.20}$$

Como é sabido, uma rotação por qualquer ângulo preserva todos os ângulos e distâncias entre os pontos do plano.

Ex. 2.75: Escreva as matrizes de rotação para os ângulos $45°$, $60°$, $90°$, $180°$, e $-90°$.

Ex. 2.76: Determine a matriz de rotação que transforma o eixo X na reta pela origem paralela ao vetor (X, Y).

Note que a fórmula (2.20) descreve apenas rotações cujo centro (ponto fixo) é a origem. Na seção 2.8.9 veremos como construir uma matriz de rotação cujo centro é um ponto arbitrário do aquém.

2.8.4 Transformações de escala

Transformação de escala

Para ampliar ou reduzir uma figura, mantendo-se sua orientação, basta multiplicar cada coordenada cartesiana de cada ponto por um fator de escala conveniente; ou seja, aplicar a cada ponto a *transformação de escala*

$$(X, Y) \mapsto (\alpha X, \beta Y)$$

O par de fatores de escala (α, β) pode ser entendido como sendo a imagem do ponto cartesiano $(1, 1)$. Em coordenadas homogêneas, esta transformação é dada por

$$[w, x, y] \mapsto [w, x, y] \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \beta \end{bmatrix} \tag{2.21}$$

Se o fator de escala é o mesmo para as duas coordenadas, a ampliação ou redução é dita *uniforme*, e preserva todos os ângulos e direções entre os pontos do plano. Caso contrário, apenas as direções horizontais e verticais são preservadas.

Ex. 2.77: Escreva a matriz de mudança de escala que leva o ponto (X_0, Y_0) para o ponto (X_1, Y_1) (suponha que nenhum desses números é zero.)

2.8.5 Reflexões

A transformação

$$(X, Y) \mapsto (-X, Y)$$

Reflexão

aplicada aos pontos de uma figura reflete a mesma em torno do eixo Y, invertendo o sentido do eixo X. É um caso particular de transformação de escala, com fatores $(-1, 1)$; sua versão homogênea é portanto

$$[w, x, y] \mapsto [w, x, y] \begin{bmatrix} +1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & +1 \end{bmatrix} \tag{2.22}$$

A reflexão em torno do eixo X é análoga. Na seção 2.8.9 veremos como construir uma matriz de reflexão cujo eixo é uma reta ordinária qualquer.

Note que reflexões preservam distâncias, e invertem o sentido dos ângulos, preservando seu valor absoluto.

2.8.6 Cisalhamentos

A transformação

$$(X, Y) \mapsto (X + \alpha Y, Y)$$

Cisalhamento

é chamada de *cisalhamento horizontal*: Esta transformação preserva a coordenada Y do argumento, e desloca a coordenada X por uma distância proporcional à coordenada Y. (Assim, ela poderia ser usada para converter letras "romanas" em "itálicas".) Em particular, o ponto $(0, 1)$ é levado para $(\alpha, 1)$. A forma homogênea desta transformação é

$$[w, x, y] \mapsto [w, x, y] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \alpha & 1 \end{bmatrix} \tag{2.23}$$

As transformações de *cisalhamento vertical* são definidas de modo análogo. O cisalhamento mais geral mantém fixos os pontos de uma reta ordinária r, e desloca qualquer outro ponto finito numa direção paralela a r, sendo o deslocamento proporcional à distância de p a r. Na seção 2.8.9 veremos como construir esta matriz, dadas r e a constante de proporcionalidade.

2.8.7 Composição de transformações

Composição

Se F e G são duas funções de $\mathbb{T}^{\nvDash}$ para $\mathbb{T}^{\nvDash}$, como as descritas acima, a *transformação composta* $p \mapsto G(F(p))$ equivale aplicar F e G em seqüência, nesta ordem, a cada ponto.

Ordem da composição

Observe que, na notação funcional ordinária $G(F(p))$, a ordem em que essas funções são escritas é oposta à ordem em que elas são aplicadas. Para evitar este inconveniente, adotaremos neste curso a notação funcional pós-fixa usada pelos algebristas. Ou seja, a aplicação de uma função F a um argumento x será denotada por xF, em vez de $F(x)$; e a composição de duas funções F e G, aplicadas nessa ordem, será denotada por FG, em vez de $G \circ F$. Portanto, em lugar de $F(G(x))$ escreveremos xFG, que pode ser lido tanto $(xF)G$ quanto $x(FG)$.

Isometria

Compondo as transformações "básicas" vistas nas seções 2.8.2–2.8.6, podemos obter outras classes interessantes. Por exemplo, combinando rotações, translações, e reflexões obtemos as chamadas *transformações isométricas*, ou *isometrias*, ou *movimentos rígidos* do plano, que são justamente todas as funções de $\mathbb{T}^{\not\models}$ em $\mathbb{T}^{\not\models}$ que preservam as distâncias e ângulos entre os pontos. Se juntarmos a essas as transformações uniformes de escala, obtemos as *transformações euclidianas*, ou *similaridades*, que preservam apenas os ângulos e as razões entre distâncias. Por outro lado, se combinarmos as isometrias com os cisalhamentos horizontais e verticais, obtemos as *transformações unitárias*, que preservam áreas e o paralelismo entre retas.

Similaridade

Transformação unitária

Em qualquer caso, note que toda transformação F assim obtida pode ser escrita na forma $[w, x, y] \mapsto [w, x, y]\mathsf{F}$, onde F é uma matriz 3×3. A composição FG de duas transformações F e G equivale obviamente ao produto FG das matrizes correspondentes, nessa ordem.

Ex. 2.78: Seja R a rotação por $90°$ em torno da origem, e T a translação pelo vetor $(2, 3)$. Calcule as matrizes para as transformações $A = TR$ e $B = RT$. Explique (geometricamente) o efeito de A e B, e a diferença entre as duas.

Ex. 2.79: Mostre que uma transformação é de similaridade se e somente se sua matriz tem a forma

$$\begin{bmatrix} 1 & x_d & y_d \\ 0 & +a & +b \\ 0 & -b & +a \end{bmatrix}$$

onde a e b são números reais, não ambos nulos, e x_d, y_d são números reais quaisquer. Qual o significado geométrico desses parâmetros?

2.8.8 Transformação inversa

Transformação inversa

Toda transformação F dentre as classes descritas acima é uma bijeção de $\mathbb{T}^{\nvDash}$ para $\mathbb{T}^{\nvDash}$; portanto, ela admite uma transformação inversa F^{-1}, tal que FF^{-1} e $F^{-1}F$ são a função identidade de $\mathbb{T}^{\nvDash}$. Obviamente, a matriz da inversa de F é a inversa da matriz de F.

Cada uma das classes de transformações mencionadas acima é fechada também sob inversão; por exemplo, a inversa da translação por (X_0, Y_0) é a translação por $(-X_0, -Y_0)$, etc.

Ex. 2.80: Para cada uma das transformações elementares vistas acima (translações, rotações, reflexões, mudanças de escala, e cisalhamento), dê a matriz homogênea da transformação inversa.

Como sabemos, a inversa da composição FG é a composição das inversas na ordem inversa, $G^{-1}F^{-1}$.

2.8.9 Transformações conjugadas

Composição e inversão são ferramentas extremamente úteis quando queremos construir projetividades mais complexas que as descritas acima.

Conjugação

Para esse fim, usamos freqüentemente o idioma $G^{-1}FG$, chamado de *conjugada de F por G*. Observe que se F leva o ponto p no ponto q, sua conjugada $G^{-1}FG$ leva o ponto pG no ponto qG; e se p é um ponto fixo de F, então pG é um ponto fixo de $G^{-1}FG$.

Por exemplo, a transformação que roda o plano de $30°$ em torno do ponto $(3, 5)$ pode ser obtida pela composição $T^{-1}RT$, onde T é a translação por $(3, 5)$, e R é a rotação de $30°$ em torno da origem.

Ex. 2.81: Para cada uma das transformações abaixo, diga como obtê-la através da composição de projetividades simples (translações, rotações em torno da origem, mudanças de escala, reflexões nos eixos, e cisalhamentos horizontais e verticais):

(a) Reflexão em torno da reta vertical de abscissa X.

(b) Reflexão em torno da reta paralela ao vetor (X_d, Y_d) passando pela origem.

(c) Reflexão em torno da reta paralela ao vetor (X_d, Y_d) passando pelo ponto (X_p, Y_p)

(d) Mudança de eixos e escalas que leva o retângulo cartesiano $[2,4] \times [3 \times 5]$ para o retângulo $[-1,+1] \times [0,1]$.

2.8.10 Efeito de projetividades em retas

Seja F uma projetividade de $\mathbb{T}^{\not\leq}$ e r uma reta. Por definição, a imagem de r por F é a única reta $F(r)$ tal que

$$r \diamond p = F(r) \diamond F(p)$$

para todo ponto p. Note, em particular, que p está em r se e somente se $F(p)$ está em $F(r)$.

Imagem de uma reta

Os coeficientes da reta $F(r)$ podem ser obtidos multiplicando-se a inversa da matriz de F pelos coeficientes de r, que devem ser considerados como um vetor coluna (isto é, uma matriz 3×1). Ou seja,

$$F(\langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle) = \mathsf{F}^{-1} \langle \mathcal{W}, \mathcal{X}, \mathcal{Y} \rangle \tag{2.24}$$

Ex. 2.82: Determine a imagem do ponto $[1,1,1]$ e da reta $\langle 1,1,1 \rangle$ sob uma translação pelo vetor $(2,3)$.

2.8.11 Transformações afins

Transformação afim

Todas as transformações vistas até agora são casos particulares das *transformações afins* do plano, cuja forma cartesiana geral é

$$\begin{aligned} (X,Y) &\mapsto (aX + bY + e, cX + dY + f) \\ &= (X,Y) \begin{pmatrix} a & b \\ c & d \end{pmatrix} + (e,f) \end{aligned}$$

onde $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ é uma matriz real não singular, e (e,f) um vetor real. A versão homogênea é

$$[w,x,y] \mapsto [w,x,y] \begin{bmatrix} 1 & e & f \\ 0 & a & b \\ 0 & c & d \end{bmatrix} \tag{2.25}$$

É fácil verificar que as transformações afins são fechadas sob composição e inversão.

Efeito na reta Ω

Mais ainda, toda transformação afim mantém a reta Ω fixa (como conjunto, não necessariamente ponto-a-ponto); ou seja, leva pontos finitos para pontos finitos, e pontos infinitos para pontos infinito. Na verdade, estas duas propriedades podem ser usadas para definir transformações afins do plano. Aliás, que as transformações afins são justamente todas projetividades que podem ser estudadas na geometria euclidiana (ou cartesiana), sem sair dos limites do $\mathbb{R}^{\nvDash}$.

Preservação de paralelismo

Pode-se concluir também que toda transformação afim preserva o paralelismo entre retas, pois duas retas finitas são paralelas se e somente se elas encontram num ponto infinito.

Existe uma única transformação afim do plano que leva os três pontos $(0,0)$, $(1,0)$, e $(0,1)$ para três pontos dados (finitos e não colineares) $p_0 = (X_0, Y_0)$, $p_1 = (X_1, Y_1)$, $p_2 = (X_2, Y_2)$. A matriz homogênea dessa transformação é

$$A_{p_0p_1p_2} \equiv [w,x,y] \mapsto [w,x,y] \begin{bmatrix} 1 & X_0 & Y_0 \\ 0 & X_1 - X_0 & Y_1 - Y_0 \\ 0 & X_2 - X_0 & Y_2 - Y_0 \end{bmatrix} \tag{2.26}$$

Portando, dados quaisquer dois triângulos abc e pqr (finitos e não-degenerados), existe uma única transformação afim do plano que leva $a \mapsto p$, $b \mapsto q$, e $c \mapsto r$, que é simplesmente a composição ${A_{abc}}^{-1} A_{pqr}$.

Ex. 2.83: Determine a matriz da transformação afim que leva os pontos

$$[1,2,3],\ [1,4,0],\ [1,0,4]$$

para

$$[1,2,0], [1,-1,1], [1,-1,-1]$$

2.8.12 Transformações projetivas gerais

A transformações afins são um subconjunto próprio das projetividades de $\mathbb{T}^{\nvDash}$. Em geral, projetividade pode levar pontos finitos para o infinito, e vice-versa. Considere por exemplo a transformação

$$[w,x,y] \mapsto [w,x,y]\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x,w,y]$$

que mantém o "pólo norte" $[0,0,1]$ fixo, e troca a origem $[1,0,0]$ com o "pólo leste" $[0,1,0]$. Esta projetividade troca o eixo Y com a reta no infinito Ω; portanto, ela transforma retas horizontais em retas que passam pela origem, e vice-versa. Em termos de coordenadas cartesianas, esta transformação é

$$(X,Y) \mapsto (1/X, Y/X)$$

Note que esta fórmula é indefinida quando $X = 0$; e, reciprocamente, não existe nenhum ponto do $\mathbb{R}^2$ cuja imagem tenha $X = 0$. Como este exemplo mostra, transformações projetivas não-afins — que levam pontos finitos para o infinito, ou vice-versa — só podem ser estudadas convenientemente no plano projetivo $\mathbb{T}^2$, em termos de coordenadas homogêneas.

2.9 Geometria projetiva tridimensional

A teoria de coordenadas homogêneas pode ser estendida para o espaço tridimensional (e, na verdade, n-dimensional) sem maiores problemas.

Apesar de nossa intenção declarada de estudar apenas problemas geométricos no plano, vale a pena descrever aqui as idéias essenciais dessa extensão, pois muitos problemas reais de geometria computacional, especialmente em computação gráfica, envolvem objetos tridimensionais. Além disso,alguns algoritmos que estudaremos no capítulo 6 utilizam internamente técnicas tridimensionais para resolver problemas bidimensionais.

2.9.1 Pontos

Espaço projetivo $\mathbb{T}^3$

No *espaço projetivo orientado* $\mathbb{T}^3$, os pontos são todas as quádruplas $[w,x,y,z]$ de números reais, exceto $[0,0,0,0]$, sendo que $[w,x,y,z]$ e $[\alpha w, \alpha x, \alpha y, \alpha z]$ denotam o mesmo ponto para todo $\alpha > 0$.

Modelo cartesiano

Um modelo geométrico de $\mathbb{T}^{\nVDash}$ (análogo ao modelo plano de $\mathbb{T}^{\nvDash}$) pode ser construído com duas cópias do espaço cartesiano $\mathbb{R}^{\nVDash}$, o *aquém* e o *além*; mais uma cópia da esfera $\mathbb{S}^{\nvDash}$, cada vetor unitário da qual representa um *ponto infinito* de $\mathbb{T}^{\nVDash}$. Se $w > 0$, a quádrupla $[w, x, y, z]$ denota o ponto de coordenadas cartesianas $(x/w, y/w, z/w)$ do aquém; se $w < 0$, ela denota o mesmo ponto no além; see $w = 0$, ela denota o ponto infinito na direção do vetor (x, y, z).

2.9.2 Planos

Planos

Um plano (orientado) de $\mathbb{T}^{\nVDash}$ é definido por quatro coeficientes reais $\langle \mathcal{W}, \mathcal{X}, \mathcal{Y}, \mathcal{Z} \rangle$; sendo que um ponto $[w, x, y, z]$ está nesse plano se e somente se

$$\mathcal{W}w + \mathcal{X}x + \mathcal{Y}y + \mathcal{Z}z = 0$$

Os demais pontos de $\mathbb{T}^{\nVDash}$ são separados pelo plano em dois conjuntos, os *lados* do plano, que podem ser distinguidos pelo sinal da fórmula acima. Observe que o sinal da fórmula não se altera se multiplicarmos todos os coeficientes do plano pelo mesmo real $\alpha > 0$.

2.9.3 Orientação de um tetraedro

Orientação de um tetraedro

Dados quatro pontos $p_0, .. p_3$ de $\mathbb{T}^{\nVDash}$, onde $p_i = [w_i, x_i, y_i, z_i]$, definimos a *orientação* dos mesmos como sendo

$$\Delta(p_0, p_1, p_2, p_3) = \operatorname{sgn} \begin{vmatrix} w_0 & x_0 & y_0 & z_0 \\ w_1 & x_1 & y_1 & z_1 \\ w_2 & x_2 & y_2 & z_2 \\ w_3 & x_3 & y_3 & z_3 \end{vmatrix} = 0 \tag{2.27}$$

Ex. 2.84: Determine a orientação dos quatro pontos

$$[1, 0, 0, 0] \quad [0, 1, 0, 0] \quad [0, 0, 1, 0] \quad [0, 0, 0, 1]$$

Ex. 2.85: Prove que a função $\Delta(p_0, p_1, p_2, p_3)$, definida pela fórmula (2.27), é zero se e somente se os quatro pontos pertencem a um mesmo plano.

Intuitivamente, para pontos no aquém, a fórmula (2.27) diz se a trajetória $p_0 \to p_1 \to p_2 \to p_3$ é uma hélice "direita" (como a da maioria dos parafusos) ou "esquerda"; isto é, se o triângulo $p_0 \to p_1 \to p_2$ roda no sentido anti-horário quando visto de p_3. Veja a figura 2.16. (Esta interpretação supõe que os eixos de $\mathbb{R}^{\nvDash}$ estão representados na posição costumeira, com o sentido de X para Y anti-horário quando visto de Z positivo.)

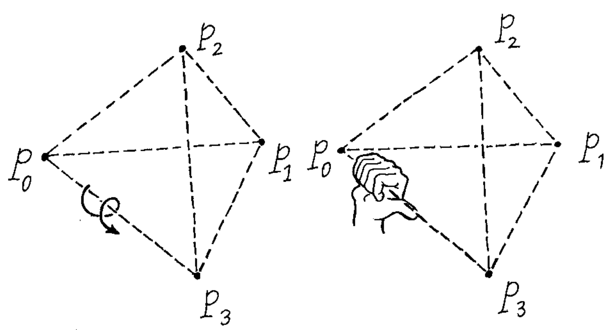


Figura 2.16: Orientação positiva de quatro pontos no espaço.

Ex. 2.86: (a) Mostre que a orientação de quatro pontos no espaço se inverte se trocarmos quaisquer dois pontos entre si. Assim, por exemplo,

$$\Delta(p_0, p_1, p_2, p_3) = -\,\Delta(p_0, p_3, p_2, p_1)$$

(b) Usando o resultado do item (a), mostre que uma permutação circular dos pontos também inverte sua orientação; isto é,

$$\Delta(p_0, p_1, p_2, p_3) = -\,\Delta(p_3, p_0, p_1, p_2)$$

Ex. 2.87: O que acontece com o valor de $\Delta(p_0, p_1, p_2, p_3)$ se trocarmos um dos pontos pelo seu antípoda? E se trocarmos todos os quatro pontos pelos seus antípodas?

2.9.4 Construindo planos e pontos

Fórmula do plano por três pontos

Por três pontos não-colineares p_0, p_1, p_2 de $\mathbb{T}^{\nvDash}$ passam exatamente dois planos, coincidentes e opostos. Um desses planos tem coeficientes $p_0 \vee p_1 \vee p_2 = \langle \mathcal{W}, \mathcal{X}, \mathcal{Y}, \mathcal{Z} \rangle$, onde

$$\mathcal{W} = +\begin{vmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} \qquad \mathcal{X} = -\begin{vmatrix} w_0 & y_0 & z_0 \\ w_1 & y_1 & z_1 \\ w_2 & y_2 & z_2 \end{vmatrix}$$

$$\mathcal{Y} = +\begin{vmatrix} w_0 & x_0 & z_0 \\ w_1 & x_1 & z_1 \\ w_2 & x_2 & z_2 \end{vmatrix} \qquad \mathcal{Z} = -\begin{vmatrix} w_0 & x_0 & y_0 \\ w_1 & x_1 & y_1 \\ w_2 & x_2 & y_2 \end{vmatrix}$$

Fórmula do ponto comum a três planos

Dualmente, três planos que não contém uma reta comum tem exatamente dois pontos antipodais em comum. As coordenadas destes pontos podem ser obtidas por uma fórmula semelhante (na verdade, dual) à fórmula de $p_0 \vee p_1 \vee p_2$ acima.

2.9.5 Transformações projetivas no espaço

A teoria das transformações projetivas pode ser facilmente generalizada para espaços projetivos de três dimensões, ou mais.

Transformações projetivas no espaço

Assim, as *transformações projetivas* de $\mathbb{T}^{\nvDash}$ são as transformações que preservam a coplanaridade de pontos. Qualquer transformação desse tipo equivale a multiplicar as coordenadas homogêneas de cada ponto por uma certa matriz real 4×4.

2.10 O plano projetivo clássico

A estas alturas, os leitores que já estavam familiarizados com geometria projetiva devem estar bastante perplexos, e esperando impacientes pelos esclarecimentos que prometemos na seção 2.2.1. Chegou a hora de cumprir essa promessa.

Plano projetivo não orientado

A verdade é que, para a grande maioria dos matemáticos e geômetras, a frase *plano projetivo* significa um espaço $\mathbb{P}^{\nvDash}$ que contém exatamente a metade dos pontos do nosso espaço $\mathbb{T}^{\nvDash}$. Mais precisamente, cada par de pontos antipodais de $\mathbb{T}^{\nvDash}$ corresponde a um único ponto de $\mathbb{P}^{\nvDash}$. Algebricamente, isto equivale a dizer que $[w, x, y]$ e $[\alpha w, \alpha x, \alpha y]$ são o mesmo ponto para todo $w \neq 0$, positivo ou negativo. Da mesma forma, cada par de retas opostas de $\mathbb{T}^{\nvDash}$ corresponde a uma única reta de $\mathbb{P}^{\nvDash}$ — isto é, as retas de $\mathbb{P}^{\nvDash}$ não são orientadas.

A simplicidade do plano projetivo "clássico" $\mathbb{P}^{\nvDash}$ tem suas vantagens. Por exemplo, as palavras "igual" e "coincidente" significam a mesma coisa. Lá não existem o aquém e o além, pois o conjunto dos pontos finitos de $\mathbb{P}^{\nvDash}$ equivale a uma única cópia de $\mathbb{R}^{\nvDash}$. Em $\mathbb{P}^{\nvDash}$, duas retas distintas se encontram num único ponto, e dois pontos distintos determinam uma única reta; com isto, os axiomas da geometria de $\mathbb{P}^{\nvDash}$ são mais próximos aos da geometria euclidiana.

Infinito

Lados

Entretanto, trabalhar no plano $\mathbb{P}^2$ também tem muitas desvantagens. Em primeiro lugar, não podemos mais distinguir o ponto no infinito com direção $+d$ do ponto com direção $-d$. Pior ainda, não podemos distinguir os dois lados de uma reta. Na verdade, no plano $\mathbb{P}^2$, toda reta tem apenas um lado; isto é, quaisquer dois pontos fora da reta podem ser ligados por um caminho contínuo que não cruza a reta.

Orientação de triângulos

Mais ainda, em $\mathbb{P}^2$ podemos transformar quaisquer triângulo próprio para qualquer outro, de maneira contínua, sem nunca passar por um triângulo degenerado. Isto significa que não podemos definir a orientação de um triângulo.

Segmentos

Convexidade

Finalmente, em $\mathbb{P}^2$ não é possível definir o segmento que liga dois pontos dados, pois existem em geral duas maneiras de se ir de um ponto para outro em linha reta, e não é possível distinguir entre essas duas alternativas de maneira consistente. Esta ambigüidade complica a definição de figura convexa, e invalida quase todas as propriedades que tornam esse conceito interessante na geometria euclidiana — por exemplo, a intersecção de duas figuras convexas de $\mathbb{P}^2$ pode não ser convexa.

Como veremos nos capítulos seguintes, os conceitos de segmento, convexidade, interior, lado, orientação, e derivados são fundamentais para a maioria dos problemas e métodos de geometria computacional. É por esta razão que, neste livro, escolhemos trabalhar com $\mathbb{T}^2$ em vez de $\mathbb{P}^2$ — isto é, com a geometria projetiva orientada em vez da clássica.

Capítulo 3

Problemas Geométricos Elementares

Neste capítulo estudamos dois problemas geométricos simples com o objetivo de dar ao leitor uma primeira idéia da natureza das questões *algorítmicas* que aparecem em geometria computacional. Ilustramos técnicas simples para projetos de algoritmos e descrevemos as análises de suas complexidades.

As simplicidades dos enunciados desses dois problemas e o fato de tão naturalmente apelarem à intuição geométrica cotidiana os tornam ideais para uma introdução à geometria computacional.

3.1 Introdução

Presumiremos apenas alguns conhecimentos básicos de matemática discreta, embora uma pequena experiência de programação e uma idéia intuitiva de eficiência de algoritmos além do material visto no capítulo 1 certamente seria um benefício.

3.2 Par mais próximo

Em um sistema de controle de tráfego aéreo auxiliado por computador, é de vital importância que a informação de qual par de aeronaves estão mais próximas seja computável do modo mais eficiente possível, já que estas seriam as mais prováveis de colidir.

O *problema do par mais próximo*, que consiste em determinar quais são dois pontos que estão à menor distância dentre todos os pares de pontos em uma coleção finita de pontos dados, é um problema fundamental em geometria computacional tanto pela simplicidade de seu enunciado quanto pela importância de suas aplicações.

Apesar da formulação do problema fazer sentido em qualquer dimensão ($d \geq 1$), vamos tratar do caso bi-dimensional. Para que possamos nos ater aos aspectos algorítmicos do problema, sem fazer o leitor se distrair com situações específicas de conjuntos de pontos dos dois lados do plano projetivo, restringiremos esta seção ao tratamento de conjuntos de pontos contidos no aquém.

3.2.1 Algoritmo ingênuo

A primeira idéia de solução para este problema é a aplicação de uma busca exaustiva em todos os pares de pontos da coleção dada à procura da distância mínima (ou de um par de pontos que cuja distância seja mínima).

Isso nos leva ao seguinte algoritmo ingênuo (ou por força bruta):

Algoritmo 3.1 $\mathrm{PMP}_1(S)$
Dada uma coleção $S = (p_1, p_2, \ldots, p_n)$ de n pontos devolve a distância mínima entre pontos de S.

1. $D \leftarrow \infty$
2. Para todo i em $[1..n]$ faça
   - 2.1. Para todo j em $[i+1..n-1]$ faça
     - 2.1.1. $D \leftarrow \min\{D, d(p_i, p_j)\}$
3. Devolva D.

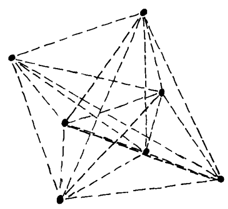


Figura 3.1: Busca em todos os pares.

Não há muito o que dizer sobre a correção ou a complexidade deste algoritmo. Como no passo 2.1.1 é calculado o mínimo entre a distância do par de pontos sendo considerado e a menor distância encontrada até o momento, D nunca cresce e eventualmente assume o valor mínimo desejado. Como este passo é executado $\binom{n}{2} = n(n-1)/2 \in O(n^2)$ vezes, este é um algoritmo de complexidade quadrática.

Ex. 3.1: No capítulo 1 vimos que se um algoritmo é projetado a partir de uma prova construtiva da existência de solução, tem-se essencialmente estabelecida a correção do procedimento.

(a) Faça uma demonstração por indução da existência de uma solução para o problema do par mais próximo que resulte no seguinte algoritmo:

Algoritmo 3.2 $\mathrm{PMP}_2(S)$
Dada uma coleção $S = (p_1, p_2, \ldots, p_n)$ de n pontos devolve um par de pontos de S que realiza a distância mínima entre pontos de S.

1. Se $|S| = 2$ então devolva (p_1, p_2).
2. Senão escolha um elemento arbitrário $p_i \in S$. Seja $S' = S - (p_i)$.
3. Obtenha recursivamente o par mais próximo de S': $(p_k, p_\ell) = \mathrm{PMP}_2(S')$.
4. Procure o ponto p_j de S' mais próximo de p_i.
5. Se $d(p_i, p_j) < d(p_k, p_\ell)$ então devolva (p_i, p_j) senão devolva (p_k, p_ℓ).

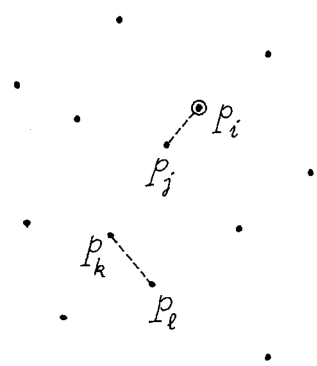


Figura 3.2: O passo da indução.

(b) Faça a análise de complexidade deste algoritmo obtendo e resolvendo a relação de recorrência correspondente.

3.2.2 Algoritmo por divisão e conquista

A razão para a complexidade quadrática do algoritmo 3.1 é que se procura o par mais próximo dentre todos os $\binom{n}{2}$ possíveis pares. O comportamento do algoritmo 3.2 do exercício 3.1 peca pela mesma razão pois ele gasta $O(n)$ comparações para determinar o ponto mais próximo de p_i. Será que é possível melhorar este tempo?

É em geral muito útil, em geometria computacional, fazer incursões em dimensões menores de modo a obter idéias que sugiram soluções eficientes para dimensões maiores.

Considerando o problema do par mais próximo em uma dimensão, vemos imediatamente que existe um simples algoritmo que o resolve de maneira muito eficiente: basta ordenar os "pontos" dados e procurar o par mais próximo através de uma varredura linear destes pontos. A observação de que o par de pontos mais próximos é necessariamente um par consecutivo no conjunto ordenado é o que permite desenvolver este algoritmo. Portanto, a noção de consecutividade é fundamental. Se tentamos generalizar o algoritmo acima para dimensões maiores, vemos que precisaríamos conhecer a priori uma direção na qual o par de vizinhos mais próximos seria consecutivo. Como isso não é possível para outras dimensões, onde o segmento de reta unindo um par de vizinhos mais próximos pode ter qualquer orientação, precisamos descobrir alguma outra propriedade que, servindo para o projeto de um algoritmo eficiente em uma dimensão, possa ser generalizada para dimensão dois.

Considere então a seguinte idéia para um algoritmo baseado no paradigma de divisão e conquista. Se dividimos um conjunto S de n pontos da reta pela sua mediana, temos dois subconjuntos disjuntos, digamos S_1 e S_2, com todos os pontos de S_1 à esquerda da mediana e os de S_2 à direita. O par de pontos mais próximos em S será um par contido em S_1 ou um par contido em S_2 ou será formado por um par muito particular de pontos de $S_1 \times S_2$. Este par necessariamente teria que ser composto do ponto mais à direita de S_1 e do mais à esquerda de S_2, isto é, *os dois pontos vizinhos da mediana — ponto de separação de S_1 e S_2.*

Vejamos agora como generalizar esta idéia para um algoritmo que resolve o problema do par mais próximo eficientemente em dimensão dois.

Começamos com uma outra prova da existência de solução para o problema:

Prova (por indução em n):

Base: Se $n = 2$, (p_1, p_2) é o par mais próximo de S.

Hipótese: Assuma que é possível determinar o par de pontos mais próximos em coleções de até $\lceil n/2 \rceil$ pontos no plano.

Passo: Particione a coleção S em duas sub-coleções S_1 e S_2 de $\lfloor n/2 \rfloor$ e $\lceil n/2 \rceil$ elementos, respectivamente, separados por uma reta vertical r. Por hipótese de indução, é possível determinar o par mais próximo (p_{i_1}, p_{j_1}) em S_1 e (p_{i_2}, p_{j_2}) em S_2. Seja $\delta = \min\{d(p_{i_1}, p_{j_1}), d(p_{i_2}, p_{j_2})\}$. Considere a faixa vertical $\mathcal{F}$ de largura 2δ centrada na reta r. Determine o par mais próximo formado de um ponto de S_1 e um ponto de S_2, ambos dentro da faixa $\mathcal{F}$. Seja (p_k, p_ℓ) este par. Se $d(p_k, p_\ell) \leq \delta$, então o par mais próximo de S é (p_k, p_ℓ), senão é aquele dentre (p_{i_1}, p_{j_1}) e (p_{i_2}, p_{j_2}) que realiza a distância δ. $\square$

Como no caso da prova anterior, obtemos facilmente um algoritmo a partir desta prova. O cuidado que temos que tomar é com a eficiência do passo de construção o qual essencialmente define a complexidade resultante do algoritmo. Vamos então explorar a seguinte idéia.

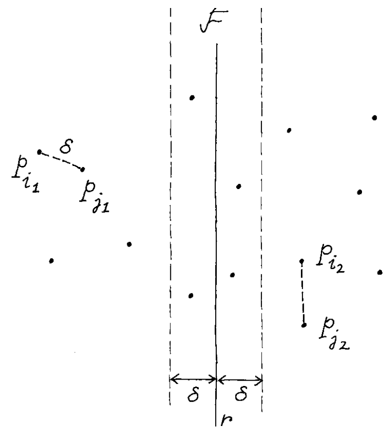


Figura 3.3: O processo de divisão.

Para cada ponto p_i de $S_1 \cap \mathcal{F}$ basta que se procure o ponto em $S_2 \cap \mathcal{F}$ mais próximo de p_i entre aqueles que estiverem dentro de um *quadrado* (com lados paralelos aos eixos coordenados) de altura e largura 2δ, centrado em p_i. Se o número de tais pontos fosse ainda $O(n)$ em nada adiantaria esse aprimoramento, mas felizmente é fácil ver que não pode haver mais de *seis* tais candidatos pois do contrário haveria um par de pontos em S_2 cuja distância seria menor que δ o que não pode ocorrer. O problema que resta é: como encontrar os pontos de S_2 dentro da faixa $\mathcal{F}$? E ainda mais, como realizar esse processamento para todos os pontos de $S_1 \cap \mathcal{F}$ em tempo total linear em n?

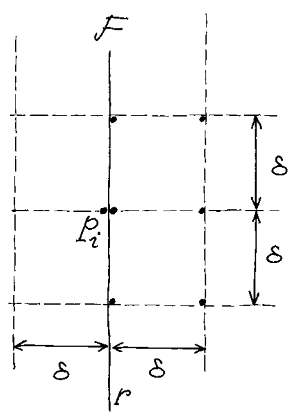


Figura 3.4: Cada $p_i \in S_1$ tem no máximo seis vizinhos em $\mathcal{F} \cap \mathcal{S}_{\in}$.

Justamente a classificação dos pontos de S em seqüência crescente por ordenadas permite que se faça isso de maneira eficiente.

Algoritmo 3.3 $\mathrm{PMP}_3(S)$
Dada uma coleção $S = (p_1, p_2, \ldots, p_n)$ de n pontos devolve a distância mínima entre pontos de S.

1. Ordene os pontos de S por abscissa e armazene num vetor $\mathcal{V}_x$.
2. Ordene os pontos de S por ordenada e armazene num vetor $\mathcal{V}_y$.
3. Devolva o par obtido pelo algoritmo $PMP_3\text{-}Aux(S)$ abaixo.

Algoritmo 3.4 $\mathrm{PMP}_3\text{-Aux}(S)$

1. Se $|S| = 2$ então devolva (p_1, p_2).
2. Senão, calcule a mediana m_x das abscissas de S. Seja r a reta vertical com abscissa m_x.
3. Divida S em duas coleções S_1 e S_2 com $\lfloor n/2 \rfloor$ e $\lceil n/2 \rceil$ pontos, respectivamente, sendo que todos os pontos de S_1 estão à esquerda de (ou sobre) a reta r e os pontos de S_2 estão à direita de (ou sobre) a reta r.
4. Obtenha recursivamente o par mais próximo de S_1:
$$(p_{i_1}, p_{j_1}) = PMP_3\text{-}Aux(S_1)$$
e o par mais próximo de S_2:
$$(p_{i_2}, p_{j_2}) = PMP_3\text{-}Aux(S_2).$$
5. Seja $\delta = \min\{d(p_{i_1}, p_{j_1}), d(p_{i_2}, p_{j_2})\}$
6. Seja $\mathcal{F}$ a faixa de largura 2δ centrada na reta r. Procure por varredura vertical, conforme detalhado abaixo, o par de pontos $p_k \in S_1 \cap \mathcal{F}$ e $p_\ell \in S_2 \cap \mathcal{F}$ mais próximos.
7. Dentre os três pares de pontos, (p_{i_1}, p_{j_1}), (p_{i_2}, p_{j_2}), (p_k, p_ℓ), devolva aquele que realiza a menor distância.

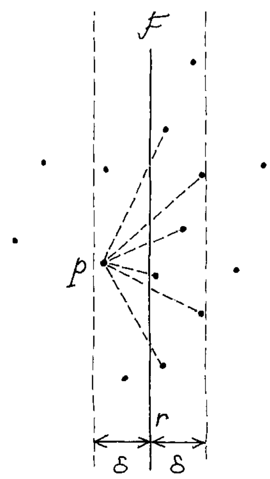


Figura 3.5: O passo de conquista.

Detalhamento do passo 6

Para se realizar o passo 6, basta fazer:

6.1 Utilizando o vetor ordenado $\mathcal{V}_y$ criado no passo 2 do algoritmo 3.3, percorra os pontos de $S_1 \cap \mathcal{F}$ em ordem ascendente de ordenadas.

Para cada ponto p visitado, tomamos em $\mathcal{V}_y$ os *três* pontos de $S_2 \cap \mathcal{F}$ acima e os *três* pontos abaixo de p. Como observamos anteriormente, o vizinho mais próximo de p em S_2 dentro da faixa $\mathcal{F}$ tem que ser um destes seis pontos, devido à escolha da largura δ da faixa $\mathcal{F}$.

6.2 Devolva o par (p_k, p_ℓ) que minimiza as distâncias encontradas.

Complexidade

Os passos 1, 2, 5 e 7 tomam tempo constante, enquanto o passo 3 pode ser feito em tempo linear.

Como no passo 6 para cada um dos $\lfloor n/2 \rfloor$ pontos de S_1 fazemos 6 comparações de distâncias computadas com a menor distância obtida até o momento, o tempo total despendido no passo 6 é da ordem de $O(n)$.

Como o passo 4 é uma chamada recursiva com duas instâncias de tamanho (essencialmente) $n/2$, a relação de recorrência que expressa a complexidade do algoritmo 3.4 é:

$$\begin{cases} T_{\mathrm{Aux}}(n) \leq 2\, T_{\mathrm{Aux}}(\lceil n/2 \rceil) + an \quad \text{para} \quad n > 2 \\ T_{\mathrm{Aux}}(2) \in O(1), \end{cases} \tag{3.1}$$

onde a é uma constante. A solução da equação 3.1 é da ordem de $O(n \log n)$.

Assim, a complexidade $T(n)$ do algoritmo 3.3 é a soma das complexidades dos seus dois passos de ordenação e de $T_{\mathrm{Aux}}(n)$. Portanto, $T(n) \in O(n \log n)$.

3.2.3 Cota inferior

Nesta seção vamos mostrar que $\Omega(n \log n)$ é uma cota inferior para o problema do par mais próximo.

Precisaremos conhecer o problema da *Unicidade de Elementos* que consiste de:

> Dada uma coleção de n valores reais, determinar se eles são todos distintos.

Este problema tem uma cota inferior de $\Omega(n \log n)$ no modelo de árvore decisões algébricas. A demonstração deste fato é demasiado envolvida para apresentação neste texto, mas como consiste de um exemplo primal de aplicação da técnica desenvolvida por Ben-Or, recomendamos ao leitor interessado a leitura do artigo original [BO83].

Faremos, então, uma redução do problema da unicidade de elementos para o problema do Par Mais Próximo, o que estabelecerá a cota inferior para este último, conforme seção 1.7.

Considere o diagrama:

$$\begin{tikzcd}
\mathrm{UE} & \propto_n & \mathrm{PMP} \\
\mathbb{I}_{\mathbb{UE}} \arrow[rr, "\tau_I"] & & \mathbb{I}_{\mathbb{PMP}} \arrow[d, "A_{PMP}"] \\
\mathbb{S}_{\mathbb{UE}} & & \mathbb{S}'_{\mathbb{PMP}} \arrow[ll, "\tau_S"']
\end{tikzcd}$$

que ilustra que basta construirmos os mapeamentos τ_I e τ_S. Para tanto, dada uma instância $I_{UE} = (x_1, x_2, \ldots, x_n) \in \mathbb{I}_{\mathbb{UE}}$, construiremos a instância $I_{PMP} = \tau_I(I_{UE}) \in \mathbb{I}_{\mathbb{PMP}}$ criando, para cada x_i em I_{UE} um ponto no plano. Note que, como a entrada para o problema do par mais próximo, conforme enunciado, é uma *coleção* (e não um conjunto) de pontos, se mapeamos (x_i) no ponto $(x_i, 0)$, não estaremos destruindo quaisquer repetições de pontos que estivessem presentes na coleção entrada do problema da unicidade de elementos. Assim, faremos $\tau_I(x_i) = (x_i, 0)$.

A segunda parte consiste apenas em fazer τ_S determinar se havia unicidade dos elementos de I_{UE} baseado na solução de I_{PMP}. Ora, mas os elementos de I_{UE} eram todos distintos se e somente se a menor distância que ocorre entre pares de elementos de I_{PMP} é não nula. Portanto, $\tau_S(0) =$ NÃO, e $\tau_S(d) =$ SIM para todo $d \neq 0$.

Como ambas as transformações τ_I e τ_S são realizáveis em tempo $O(n)$, isso completa a redução e prova a cota inferior $\Omega(n \log n)$ para o problema do par mais próximo. Conseqüentemente, o algoritmo 3.3 é ótimo no modelo de árvore de decisões algébricas.

Ex. 3.2: Considere o seguinte problema:

> *Dados n círculos no plano, todos de mesmo raio $r \geq 0$, determinar se existem dois deles que se interceptam.*

(a) Apresente um algoritmo de complexidade $O(n \log n)$ que resolve este problema.

(b) Mostre que $\Omega(n \log n)$ é cota inferior para o problema, estabelecendo assim a otimalidade assintótica de seu algoritmo.

3.3 Localização de pontos em relação a um polígono

O leitor deve se recordar que nossa opção pelo plano projetivo orientado visa evitar desvantagens inerentes à geometria euclidiana assim como outras que surgiriam se tivéssemos escolhido trabalhar com a geometria projetiva clássica. Uma das desvantagens desta última é que a remoção de uma reta deixa uma única componente conexa equivalente (topologicamente) a um disco. Portanto, nunca se pode dizer que dois pontos dados estão em lados opostos de uma reta. Da mesma forma, não podemos ali decidir se dois pontos estão do mesmo lado ou de lados opostos de um polígono.

No plano projetivo orientado este problema tem sentido, isto é, o Teorema da Curva de Jordan garante que uma curva fechada sem auto-intersecção (e, em particular, um polígono simples) divide o espaço em duas regiões disjuntas, que são os seus lados (positivo e negativo).

3.3.1 Conceitos básicos

Inicialmente, vamos formalizar alguns conceitos elementares necessários para o entendimento dos enunciados dos problemas.

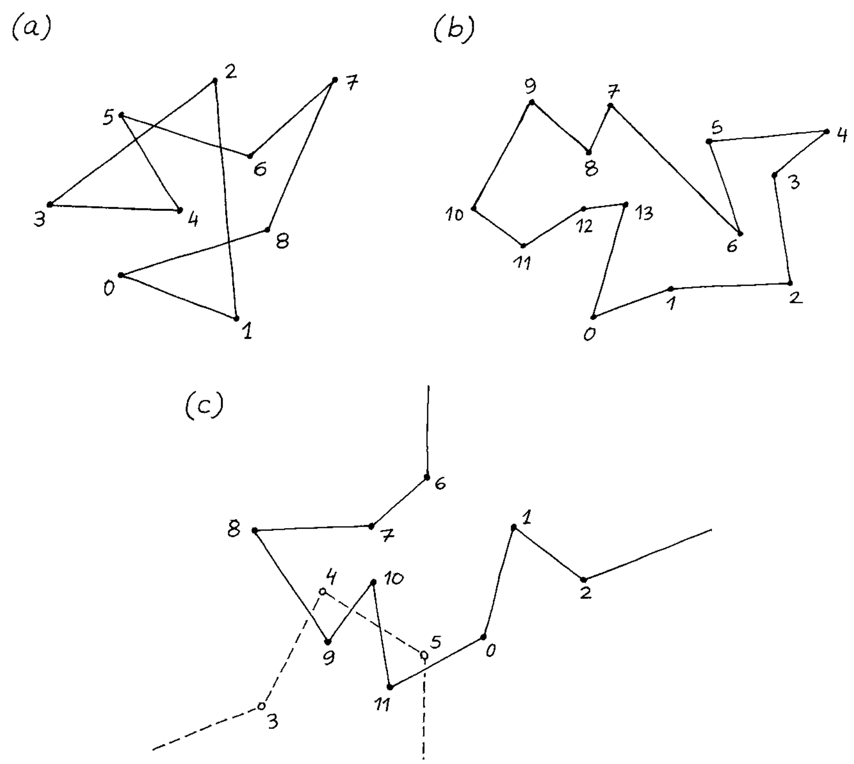


Figura 3.6: Três polígonos.

Polígono

Definição 3.1 *Dados n pontos distintos e não antipodais $p_0, p_1, \ldots,$ p_{n-1}, eles definem uma seqüência de segmentos de reta em ordem cíclica $e_i = p_i p_{i+1}$ para $i = 0, 1, \ldots, n-1$, onde as somas de índices são tomadas módulo n. Os pontos p_i's juntamente com os segmentos e_i's formam um* polígono. *Os pontos são chamados os* vértices *e os segmentos são as* arestas *do polígono. (Fig. 3.6 (a), (b) e (c).)*

Polígono simples

Dizemos que um polígono $P = (p_0, p_1, \ldots, p_{n-1})$ é um polígono simples *se ele satisfaz à seguinte propriedade:*

> *todas as arestas de P são disjuntas exceto pelos vértices que as arestas consecutivas compartilham, isto é, $e_i \cap e_{i+1} = \{p_{i+1}\}$ para $0 \leq i \leq n-1$ e $e_i \cap e_j = \emptyset$ para $i < j-1$ e $j < i+n-1$. (Fig. 3.6 (b) e (c).)*

Os lados de um polígono simples

Gostaríamos de ser capazes de distinguir os dois lados de polígonos simples em $\mathbb{T}^2$ da mesma forma que o fazemos para polígonos simples contidos no plano euclidiano.

Pelo Teorema da Curva de Jordan, o complemento de um polígono simples é formado por duas regiões disjuntas. No plano euclidiano, uma delas tem área limitada e a outra tem área ilimitada. Em $\mathbb{T}^2$ não podemos garantir isso, já que há polígonos simples que dividem o espaço em regiões de área infinita.

O que podemos fazer tanto em $\mathbb{R}^2$ quanto em $\mathbb{T}^2$ de modo consistente é distinguir os lados de polígonos simples baseado na ordem dada dos seus vértices.

Definição 3.2 *Definimos o* lado positivo *de um polígono simples $P = (p_0, p_1, \ldots, p_{n-1})$ em $\mathbb{T}^2$, como sendo a região que fica do lado esquerdo do perímetro, quando este é percorrido na ordem dada.*

Mais precisamente, observe que toda vizinhança V suficientemente pequena de um ponto p não extremo da aresta $p_0 p_1$ pode ser dividida em três partes:

V_0: *a intersecão de* V *com o segmento* p_0p_1,

V_+: *a parte de* V *inteiramente contida no lado positivo da reta* $r_0 = p_0 \vee p_1$, *e*

V_-: *a parte de* V *inteiramente contida no lado negativo de* r_0.

O lado positivo *do polígono* P *é aquele lado que inclui* V_+ *(e, portanto, exclui* $V-$*).*

O outro lado de P *é chamado* lado negativo.

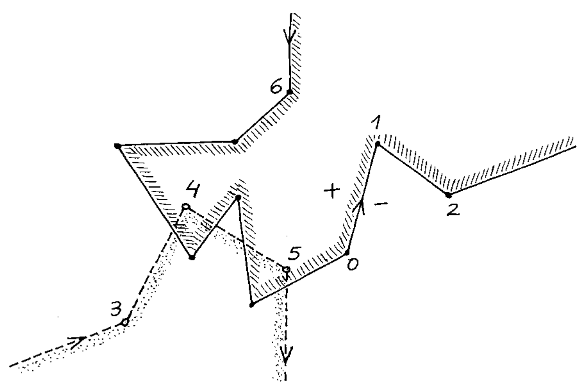


Figura 3.7: O lado positivo de um polígono contido em $\mathbb{T}^{\not\models}$.

No modelo esférico, o lado positivo de uma reta m, unido com a própria reta forma um hemisfério fechado. Referiremos a ele também como o lado não negativo de m. Dado um polígono simples P se existe uma reta m em cujo lado não negativo o polígono está contido, diremos que m é uma reta limitante de P.

Os polígonos simples para os quais existem retas limitantes têm propriedades interessantes. Eles gozam de praticamente todas as características de polígonos simples contidos no plano euclidiano. Por outro lado, os polígonos simples que são interceptados por todas as retas de $\mathbb{T}^{\not\models}$ são menos intuitivos.

Ex. 3.3: Mostre que os dois lados do seguinte polígono são congruentes (e portanto, essencialmente, indistinguíveis). Conclua que toda reta de $\mathbb{T}^{\nvDash}$ o intercepta.

$$P = (p_1, p_2, p_3, p_4) \quad \text{onde}$$

$$p_1 = [+1, +1, +1], \; p_2 = [-1, +1, -1]$$

$$p_3 = [+1, -1, -1], \; p_4 = [-1, -1, +1]$$

Uma das complicadas idiosincrasias de polígonos como este é o fato de que pelo menos um de seus lados contém pontos antipodais.

No próximo exercício, o leitor mostrará que sempre há um dos lados de um polígono que não pode conter reta alguma.

Ex. 3.4: Seja P um polígono simples em $\mathbb{T}^{\nvDash}$ e suponha que existe uma reta limitante m para Pe digamos que m está contida no lado negativo de P. Mostre que o lado positivo de P não pode conter nenhuma reta. [Sugestão: se houvesse tal reta, onde estariam os pontos onde ela interceptaria m?]

Ex. 3.5: Determine se cada um dos seguintes polígonos é simples e, caso afirmativo, desenhe-o hachuriando seu lado positivo. Decida também se existem retas limitantes para cada um deles.

(a)	$[1, 0, 0]$,	$[1, 1, 0]$,	$[1, 1, 1]$,	$[1, 1, 0]$
(b)	$[1, 0, 0]$,	$[1, 0, -1]$,	$[1, -1, -1]$,	$[1, -1, 0]$
(c)	$[1, 1, -1]$,	$[1, 1, 1]$,	$[-1, -1, 0]$	
(d)	$[0, 1, 0]$,	$[0, 0, 1]$,	$[0, -1, 0]$,	$[0, 0, -1]$

3.3.2 De que lado de um polígono simples um ponto está?

A aparente simplicidade de algumas soluções intuitivas para certos problemas geométricos se deve ao fato de que em geral tratamos apenas instâncias pequenas. Entretanto, quando projetamos algoritmos para soluções automáticas de problemas, pretendemos que estes operem corretamente em instâncias de tamanhos arbitrários.

O problema de determinar a que lado de um polígono simples um ponto pertence só nos apresenta desafio visual quando tratamos de polígonos com pelo menos várias dezenas de vértices. Ainda assim, a grande quantidade de informações obtidas pelo olho humano não é facilmente extraível para processamento automático a partir exclusivamente das coordenadas dos vértices. Por isso, é essencial a determinação de propriedades características que sejam computacionalmente eficientes.

Vamos tratar de obter uma tal propriedade para o problema da localização de um ponto com respeito ao lado positivo de um polígono simples.

Seja $P = (p_0, p_1, \ldots, p_{n-1})$ um polígono simples e q um ponto em $\mathbb{T}^{\neq}$. Se contarmos o número de interseções do segmento p_0q com as arestas de P, teremos o número de vezes que um ponto viajando sobre p_0q muda de um lado ao outro do polígono P. Se soubermos o primeiro lado encontrado pelo ponto viajante assim que ele deixa o ponto p_0, saberemos pela paridade do número de interseções se q está deste mesmo lado ou do lado oposto.

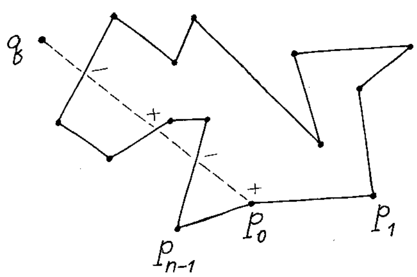


Figura 3.8: Localização de ponto em polígono simples.

Isso resulta no seguinte algoritmo:

Algoritmo 3.5 $\mathrm{LocPol}_1(q, P)$
Dado um polígono simples P e um ponto q, devolve o lado de P em que q se encontra.

1. Se o segmento p_0q, ao sair de p_0, entra no lado positivo de P, então faça $s \leftarrow +1$, senão, faça $s \leftarrow -1$.
2. Para cada i em $\{1 \,..\, n-2\}$, faça
   2.1. Se a aresta p_ip_{i+1} cruza qp_0 então inverta o sinal de s.
3. Devolva o sinal de s.

Detalhamento do passo 1
Para se realizar o passo 1, basta fazer:

1.1 $a \leftarrow \Delta(q, p_0, p_1)$
1.2 $b \leftarrow \Delta(p_{n-1}, p_0, q)$
1.3 $t \leftarrow \Delta(p_{n-1}, p_0, p_1)$
1.4 Se $a = b = t$ então $s \leftarrow t$ senão $s \leftarrow -t$.

Detalhamento do passo 2.1
Uma das maneiras de se executar o passo 2.1 é:

2.1.1 Calcule a reta $m = p_0 \vee q$.
2.1.2 Verifique a posição de p_i em relação m.
2.1.3 Se p_i está do lado positivo de m ou sobre m, e p_{i+1} está no lado negativo de m, então:
2.1.3.1 Se $\Delta(p_i, p_{i+1}, p_0) = -1$ e $\Delta(p_i, p_{i+1}, q) = +1$, então devolva "VERDADEIRO" (p_i, p_{i+1} cruza qp_0), senão devolva "FALSO" (p_i, p_{i+1} não cruza qp_0).

Complexidade
Os passos 1 e 3 tomam tempo constante. A busca por interseções feita no passo 2.1 toma tempo linear. Portanto, este algoritmo toma tempo total $T(n) \in O(n)$.

3.4 Convexidade facilita localização de pontos

Acabamos de ver como localizar um ponto em relação aos lados positivo e negativo de um polígono simples. Observe que o único uso que fizemos da geometria do polígono foi do fato de ele dividir o plano em duas partes disjuntas. A simplicidade do polígono não sugere nenhuma possibilidade de melhoria na eficiência do algoritmo de localização. Se, porém, tivéssemos mais estrutura geométrica num polígono (por exemplo, convexidade), talvez isso nos ajudasse a desenvolver algoritmos mais eficientes para o problema de localização de pontos. Antes de vermos o que se pode ganhar quando os polígonos são convexos, precisamos formalizar a noção de convexidade na geometria projetiva orientada.

Pretendemos estender de forma consistente a definição de convexidade do plano euclidiano para o plano projetivo orientado. Para isso, precisamos poder nos referir ao segmento que liga dois pontos. Recorde-se que o segmento de reta entre dois pontos em $\mathbb{T}^{\not\models}$ só está definido quando estes pontos são *não* antipodais.

Definição 3.3 *Seja C um subconjunto de $\mathbb{T}^{\not\models}$. Dizemos que C é* convexo *se para todo par de pontos em C o segmento de reta que eles determinam está inteiramente contido em C.*

Portanto, a definição de conjunto convexo exclui conjuntos que contenham pares de pontos antipodais. Em particular, não consideramos $\mathbb{T}^{\not\models}$ como um conjunto convexo. Isso pode parecer estranho, mas note, entretanto, que isso não torna esta definição incompatível com a de convexidade no plano euclidiano pois cada uma de suas cópias (o aquém e o além) contidas em $\mathbb{T}^{\not\models}$ são convexas. Da mesma forma, qualquer hemisfério aberto é convexo.

Ex. 3.6:

(a) Mostre que o polígono formado pelos seguintes vértices não é um conjunto convexo:

$$[1,0,0], [-1,0,2], [1,1,2], [-1,2,0].$$

(b) Este polígono é simples?

Os maiores conjuntos convexos em $\mathbb{T}^{\not\models}$ são constituídos, no modelo esférico, precisamente de um hemisfério, uma metade de sua fronteira — uma metade aberta de um círculo máximo — e apenas um dos pontos extremos deste meio-círculo máximo.

Polígono convexo

Definição 3.4 *Um polígono simples* $P = (p_0, p_1, \ldots, p_{n-1})$ *em* $\mathbb{T}^{\not\leq}$ *é chamado* polígono convexo *se*

$$\Delta(p_i, pi + 1, p_{i+2} = +1$$

para todo i *entre 0 e* $n-1$.

Há várias formas equivalentes de se definir polígono convexo. Como cada uma delas é se presta a alguma situação de modo mais conveniente que as outras, vejamos três destas. Ao resolver cada um dos exercícios a seguir, não deixe de extrair o significado geométrico de cada um.

Ex. 3.7: Mostre que um polígono simples $P = (p_0, p_1, \ldots, p_{n-1})$ é convexo se e somente se

$$\Delta(p_i, pi + 1, p_j = +1$$

para todo i, j com $i < j < i + n$.

Ex. 3.8: Mostre que um polígono simples $P = (p_0, p_1, \ldots, p_{n-1})$ é convexo se e somente se

$$\Delta(p_i, pj, p_k) = +1$$

para todo i, j, k com $i < j < k < i + n$.

Ex. 3.9: Mostre que um polígono simples $P = (p_0, p_1, \ldots, p_{n-1})$ é convexo se e somente se nenhum de seus vértices é combinação convexa dos outros.

Ex. 3.10:

(a) Mostre que a seguinte implicação é também verdadeira:
Se P é um polígono convexo, então seu lado positivo forma um conjunto convexo.

(b) Por que a recíproca não é verdadeira?

Interior de polígono convexo

Definição 3.5 *Chamamos o lado positivo de um polígono convexo de* interior *do polígono e o lado negativo de* exterior.

Vejamos agora como podemos determinar em tempo sublinear se um ponto dado está no interior ou exterior de um polígono convexo.

Considere um polígono convexo $P = (p_0, p_1, \ldots, p_{n-1})$. A convexidade de P garante que existe uma reta m tal que todos os vértices de P estão do lado não negativo de m.

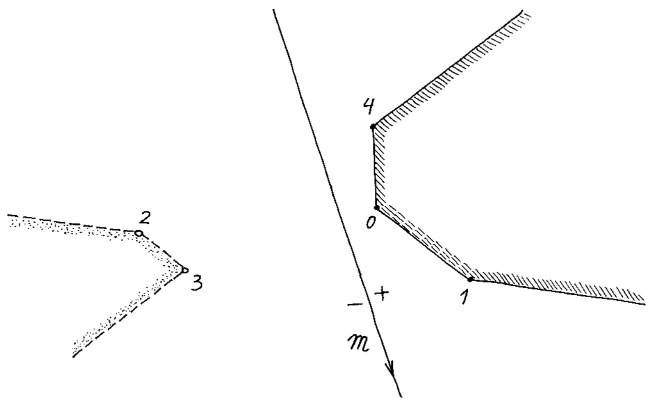


Figura 3.9: Polígono convexo contido do lado positivo de m.

Ex. 3.11: Dado um polígono convexo, mostre que é possível determinar em tempo constante uma reta em cujo lado não negativo estão todos os vértices do polígono. (Sugestão: se m é uma reta no lado exterior do polígono, então m ou $\neg m$ satisfaz a propriedade.)

Considere, em conseqüência do exercício 3.11 que temos, além do polígono convexo P, uma reta m cujo lado positivo contém P. Como o lado de m contendo P unido com m forma um hemisfério (topologicamente) fechado, temos uma situação completamente análoga à situação em que P está contido no aquém. A menos, portanto, de um homeomorfismo[1] que mapeia o hemisfério fechado determinado por m contendo P, no aquém estendido, podemos assumir que P está contido no aquém.

Em outras palavras, podemos assumir, para clareza de exposição, que nenhum dos vértices de P está no além. Deste modo, dado um ponto (de consulta) q que se deseja determinar se está no interior de P, caso q esteja no além, sabemos que q está no exterior de P.

[1]Um homeomorfismo é uma bijeção contínua cuja inversa é também contínua.

Resta, então, descrever um algoritmo que decide se um ponto q de peso positivo está no interior ou exterior de P.

Seja p um ponto no interior de P. Para cada vértice p_i de P, considere o segmento passando por p_i ligando p a um ponto no infinito — na direção do vetor $(p_i - p)$. Perceba que o aquém fica assim dividido em n triângulos todos tendo p como vértice comum. Chamemos de T_i o triângulo limitado pelos segmentos determinados por p e p_i e por p e p_{i+1}. Como estes triângulos estão ordenados circularmente em torno de p pela própria seqüência dos vértices de P dada, podemos determinar por busca binária qual o triângulo dentro do qual o ponto de consulta q está. Se q está dentro de um triângulo T_j então q estará no interior de P se e somente se q estiver do lado positivo da reta determinada pelo segmento $p_j p_{j+1}$. Veja figura 3.10.

Temos assim, o seguinte algoritmo que determina se um ponto q de peso positivo está do lado positivo (interior) ou negativo (exterior) de um polígono convexo P contido no aquém:

Algoritmo 3.6 LocPolConv(q, P)
Dado um polígono convexo P e um ponto q, devolve o lado de P em que q se encontra.

1. Seja p o baricentro de algum triângulo formado por vértices de P.
2. Como a ordem dos vértices $(p_0, p_1, \ldots, p_{n-1})$ de P estabelecem uma ordem cíclica para as retas $p \vee p_i$, determine por busca binária o índice j $(0 \leq j \leq n-1)$ tal que $(p \vee p_{j+1}) \diamond q > 0$ e $(p \vee p_j) \diamond q \leq 0$.
3. Devolva o sinal de $(p_j \vee p_{j+1}) \diamond q$.

Complexidade
Os passos 1 e 3 tomam tempo constante. A busca feita no passo 2 toma tempo logarítmico. Portanto, este algoritmo toma tempo total $T(n) \in O(\log n)$.

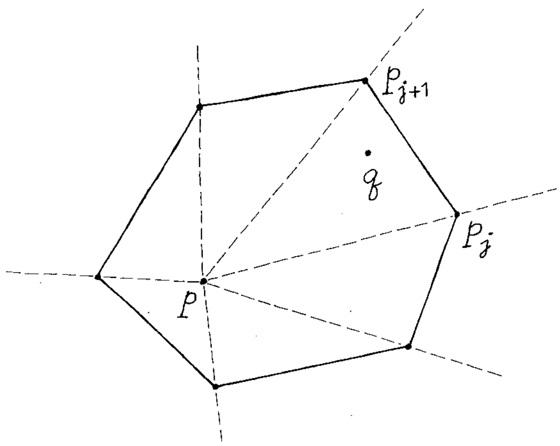


Figura 3.10: Localização de ponto em polígono convexo.

Obtivemos assim, um algoritmo que resolve o problema da localização de pontos com respeito ao interior de polígonos convexos de modo muito mais eficiente do que pudemos fazer para polígonos simples.

3.5 Pontos em estrelas

Uma classe de polígonos simples não convexos muito útil em geometria computacional é a dos polígonos estrelados que definiremos a seguir. Antes, porém, precisamos da noção de pontos mutuamente visíveis.

Ao longo desta seção, considere P um polígono simples.

Pontos que se enxergam

Definição 3.6 *Dados dois pontos p, q no interior de P, dizemos que p* enxerga *q (e vice-versa) se o segmento pq não intercepta o exterior de P. Neste caso, dizemos também que p e q são* mutuamente visíveis.

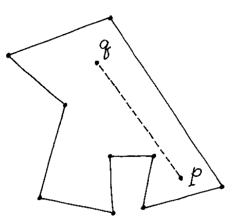


Figura 3.11: Pontos que se enxergam.

Núcleo de um polígono

Definição 3.7 *Definimos o* núcleo *do polígono P como sendo o conjunto dos pontos do seu interior que enxergam todos os pontos de P.*

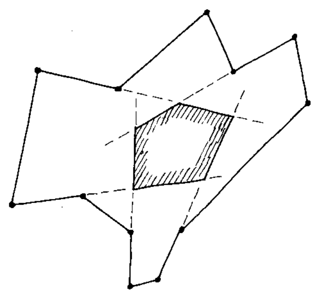


Figura 3.12: O núcleo de um polígono.

Polígono estrelado

Definição 3.8 *Dizemos que P é um polígono* estrelado *(ou* em forma de estrela*) se seu núcleo é não vazio.*

Ex. 3.12: Mostre que se existe um ponto no interior de um polígono que enxerga todos os seus **vértices**, então o polígono é estrelado.

Obviamente, todo polígono convexo é estrelado pois seu núcleo coincide com seu interior.

Ex. 3.13: Mostre que o núcleo de qualquer polígono estrelado é sempre um conjunto convexo.

Polígonos estrelados são dotados de estrutura suficiente para que se possa determinar se um ponto dado pertence a seu interior ou exterior tão eficientemente quanto foi possível fazer para os polígonos convexos. Nos exercícios a seguir o leitor mostrará isso.

Ex. 3.14: Mostre que a ordem dada dos vértices $(p_0, p_1, \ldots, p_{n-1})$ de um polígono estrelado estabelecem uma ordem cíclica para as retas $p \vee p_i$, em torno de um ponto p qualquer em seu núcleo.

Ex. 3.15: Generalize o algoritmo 3.6 para polígonos estrelados quando se conhece algum ponto de seu núcleo.

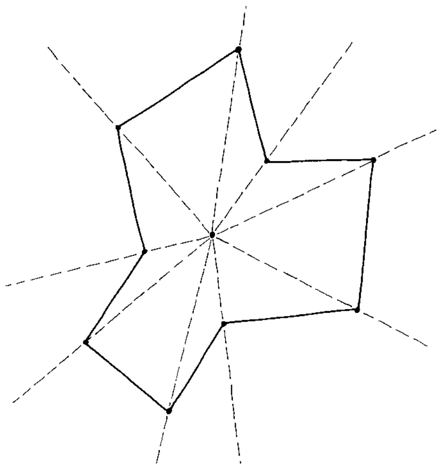


Figura 3.13: Generalização para polígonos estrelados.

Capítulo 4

Convexidade

A noção de convexidade é central a muitos problemas não apenas em geometria computacional mas também programação linear, processamento de imagens, reconhecimento de padrões, etc. Neste capítulo estudaremos principalmente o problema da construção de polígonos convexos que envolvam conjuntos de pontos minimizando a área envolvida. Embora aparentemente simples, este problema já recebeu na literatura atenção quase tão grande quanto o problema de ordenação no tocante ao projeto de soluções eficientes. Veremos aqui algumas das mais elucidativas.

4.1 Introdução

O singelo problema de determinar se um polígono dado é convexo é tão simples quanto o problema de determinar se uma seqüência finita de valores reais está ordenada. Da mesma forma, veremos que dados n pontos no plano, determinar o menor polígono convexo que os contém é da mesma ordem de dificuldade de ordenar n números reais. De fato, existem vários algoritmos de ordenação que têm correspondentes algoritmos para construção de polígonos convexos.

4.2 Determinação de convexidade

Considere inicialmente o problema de determinar se um polígono dado é convexo ou não.

Não é difícil ver que basta verificar se todas as diagonais do polígono estão inteiramente contidas no seu lado positivo. (As *diagonais* de um polígono são os segmentos que unem pares de vértices não ligados por arestas.) Por mais que se consiga fazer uma tal verificação de modo eficiente, esta abordagem nos requer fazer $\binom{n}{2} \in \Theta(n^2)$ tais verificações o que torna um tal algoritmo seriamente ineficiente. Uma melhoria desta abordagem é proposta no exercício 4.1.

Ex. 4.1:

(a) Prove que se um polígono é não convexo, então existe alguma diagonal entre vértices separados por exatamente duas arestas cujo ponto médio é externo ao polígono.

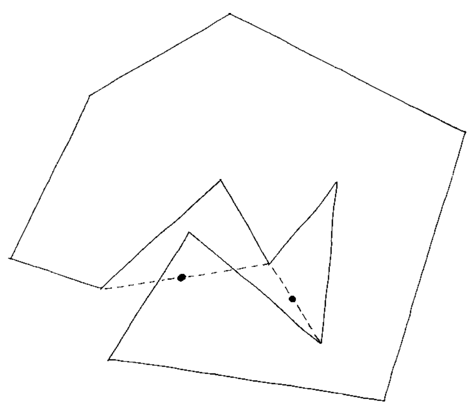


Figura 4.1: Pontos médios de diagonais

(b) Use esta propriedade para desenvolver um algoritmo para determinar se um polígono dado é convexo. A complexidade de seu algoritmo deve ser $O(n \log n)$ onde n é o número de vértices.

Computacionalmente é mais interessante explorar a propriedade de que todo polígono não convexo contém um ângulo reflexo no seu lado positivo (i.é., maior que 180°). Esta verificação deve porém ser levada a cabo sem a explícita computação de ângulos pois isto envolve recorrer a funções transcendentais (e.g., arctan) que não fazem parte dos modelos computacionais com que lidamos.

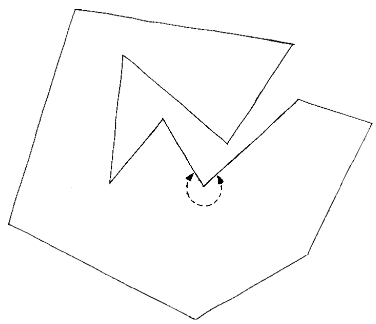


Figura 4.2: Um polígono não convexo e um de seus ângulos reflexos.

Uma maneira de atingir o mesmo objetivo é fazer uso do teste de determinação de orientação de triângulos.

Ex. 4.2:

(a) Escreva um algoritmo que determina se um polígono simples dado é convexo, utilizando a própria definição 3.4.

(b) Mostre que a complexidade de seu algoritmo é $O(n)$ onde n é o número de vértices.

Note que este algoritmo é *ótimo* pois a cota inferior $\Omega(n)$ segue do fato de que se algum algoritmo não olhar ambas as arestas incidentes a cada um dos vértices, ele não pode decidir corretamente a convexidade do polígono.

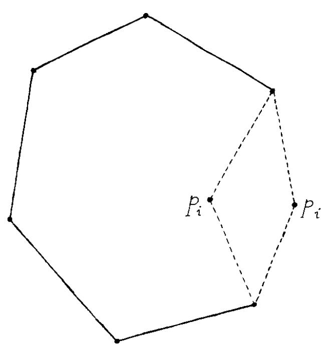


Figura 4.3: Um vértice não verificado basta para destruir a convexidade.

4.3 Casco convexo de conjuntos finitos de pontos

Dado um conjunto de pontos, definimos seu *casco convexo* (ou sua *envoltória convexa*), como sendo o menor conjunto convexo que o contém, ou mais formalmente:

Casco convexo

Definição 4.1 *Seja P um conjunto. O* casco convexo *(ou a* envoltória convexa*) de P, é o conjunto:*

$$Conv(P) = \bigcap_{X \in C_P} X,$$

onde $C_P = \{X \supset P \mid X \text{ é um conjunto convexo}\}$.

Em particular, estaremos interessados no casco convexo de conjuntos finitos de pontos, não todos colineares.

Observação 4.1 *O casco convexo de um conjunto finito de pontos, coincide com a união de um polígono convexo e seu interior.*

Ao longo deste capítulo, S denotará um conjunto de n pontos em $\mathbb{T}^{\neq}$, $S = \{p_0, p_1, \ldots, p_{n-1}\}$.

Como o casco convexo de S, $\mathrm{Conv}(S)$, pode ser determinado a partir do polígono que coincide com sua fronteira, nos algoritmos que determinam envoltórias convexas, basta produzirmos o conjunto de vértices deste polígono assumindo que a ordem apropriada dos vértices seja estabelecida. Por isso, nos referiremos às vezes ao casco convexo de S como o polígono convexo que coincide com sua fronteira.

Existem diversos algoritmos para a construção do casco convexo de conjuntos finitos de pontos, inclusive aqueles que exploram alguma eventual estrutura de que este conjunto seja dotado.

Como os vértices do casco convexo de S formam um subconjunto de S, há duas possíveis abordagens para a identificação destes: determinar os pontos de S que são vértices de $\mathrm{Conv}(S)$, ou identificar os pontos de S que *não* são vértices de $\mathrm{Conv}(S)$ de modo a eliminá-los.

Embora esta pareça uma observação trivial, veremos que tanto uma quanto a outra abordagem permitem identificar propriedades geométricas que produzem algoritmos para a construção do casco convexo.

4.3.1 Algoritmos simples para conjuntos de pontos

Como a definição de casco convexo de um conjunto de pontos (como o menor conjunto convexo que o contém) não sugere uma forma construtiva de obtê-lo, faz-se necessário determinar uma propriedade geométrica que induza a uma construção eficiente.

Note que, a princípio, não temos como decidir se o casco convexo de um conjunto de pontos de $\mathbb{T}^{\nvDash}$ é bem definido, isto é, pode acontecer que o conjunto não contenha nenhum par de pontos antipodais mas suas combinações convexas gerem um tal par.

No final deste capítulo (seção 4.3.6), veremos como resolver estas situações. Por enquanto, assuma que o conjunto de pontos dado esteja todo contido no aquém. Deste modo, o casco convexo do conjunto estará bem definido, isto é, seu interior será sempre um conjunto convexo.

Um algoritmo ingênuo

Objetivando eliminar os pontos de S que sejam interiores ao casco convexo de S, observamos que:

Propriedade 4.1 *Um ponto é interno à envoltória convexa de um conjunto se e somente se é interior a algum triângulo formado por alguma tripla de pontos do conjunto.*

Esta propriedade sugere um algoritmo trivial mas de alto custo:

Algoritmo 4.1 $\mathrm{EnvConv}_1(S)$
Dado um conjunto finito de pontos S, devolve os vértices do casco convexo de S, $Conv(S)$.

1. $C \leftarrow S$
2. Para cada tripla de pontos distintos q_1, q_2, q_3 de S faça
   - 2.1. Para cada ponto p de C, faça
     - 2.1.1. Se p é interno ao triângulo formado por q_1, q_2, q_3 então remova p de C.
3. Devolva $\text{Conv}(S) = C$.

Complexidade

Complexidade: $O(n^4)$

Como existem $\binom{n}{3} \in \Theta(n^3)$ triângulos, em cada um dos quais se verifica, no pior caso, a pertinência de cada um dos outros $n-3$ pontos, a complexidade total deste algoritmo é da ordem de n^4.

Devemos certamente conseguir fazer melhor que isso.

Outro algoritmo (não tão) ingênuo

Reta de suporte

Definição 4.2 *Seja R uma região qualquer de $\mathbb{T}^{\nvDash}$ e seja m uma reta que passa por algum ponto de R. Se um dos lados fechados determinados por m contém toda a região R, então dizemos que m é uma* reta de suporte *de R.*

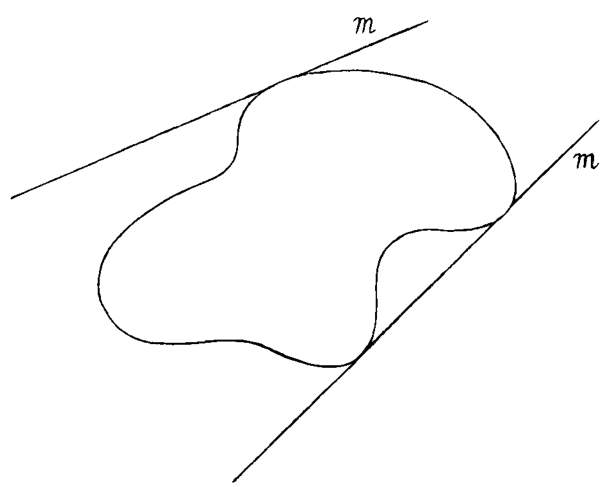


Figura 4.4: Retas de suporte.

Observe que as retas determinadas pelas arestas de um polígono convexo são todas retas de suporte do polígono. Reciprocamente:

Propriedade 4.2 *Se S é um conjunto de pontos em posição geral*[1] *então cada par de pontos de S pelos quais passa uma reta de suporte determina uma aresta de seu casco convexo.*

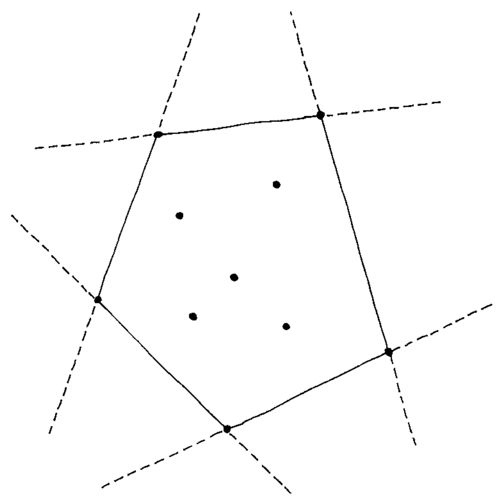


Figura 4.5: Um casco convexo e as linhas de suporte que determinam suas arestas.

Com base nesta propriedade, podemos desenvolver um algoritmo mais eficiente que o anterior. Este é o objetivo do exercício 4.3.

Ex. 4.3:

(a) Descreva um algoritmo que constrói a envoltória convexa de um conjunto de n pontos no aquém em posição geral, identificando as arestas através da propriedade 4.2.

(b) Faça a análise da complexidade de seu algoritmo e mostre que ele é mais eficiente que o algoritmo 4.1 por um fator linear.

Complexidade: $O(n^3)$

[1]Um conjunto de pontos em $\mathbb{R}^{\not\vDash}$ está em *posição geral* se ele não contém dois pontos coincidentes ou três pontos colineares.

Método Incremental

Conforme vimos na seção 1.6.1, uma das técnicas importantes de projeto de algoritmos, especialmente quando os dados não são todos conhecidos a priori, é a incremental. Vejamos como aplicá-la ao problema da construção de cascos convexos.

A correção do algoritmo abaixo é decorrência de uma prova por indução de que uma solução existe, a qual o leitor atento perceberá transparente no próprio algoritmo.

Construção incremental

Algoritmo 4.2 $\mathrm{EnvConv}_2(S)$
Dado um conjunto finito S de pelo menos três pontos, devolve o casco convexo de S, Conv(S).

1. Se $|S| = 3$ então
   1.1. Se $\Delta(p_0, p_1, p_2) > 0$ então devolva (p_0, p_1, p_2) senão devolva (p_2, p_1, p_0).
2. Escolha um ponto $q \in S$. Faça $S' \leftarrow S - \{q\}$.
3. Construa recursivamente $\mathrm{Conv}(S') = \mathrm{EnvConv}_2(S')$.
4. Se q pertence ao interior de $\mathrm{Conv}(S')$
   4.1. Então devolva $\mathrm{Conv}(S) = \mathrm{Conv}(S')$
   4.2. Senão
      4.2.1. Forme $\mathrm{Conv}(S)$ estendendo $\mathrm{Conv}(S')$ de modo que q seja um de seus vértices.
      4.2.2. Devolva $\mathrm{Conv}(S)$.

No passo 2 o fato de que q é um ponto arbitrário permite que este algoritmo seja aplicado ao caso em que os pontos são dados um a um numa ordem arbitrária.

Para cada ponto, a determinação de qual dos dois casos possíveis no passo 4 é o que ocorre pode ser feita usando-se os algoritmos discutidos na seção 3.3.2.

Detalhamento do passo 4.2.1
Tratemos portanto da situação em que q deve ser incluido na solução final (passo 4.2.1). Neste caso, duas novas arestas devem ser acrescentadas à envoltória convexa $\mathrm{Conv}(S')$ adjacentes a q, enquanto que uma cadeia de arestas (com pelo menos uma aresta) e possivelmente vértices deve ser eliminada.

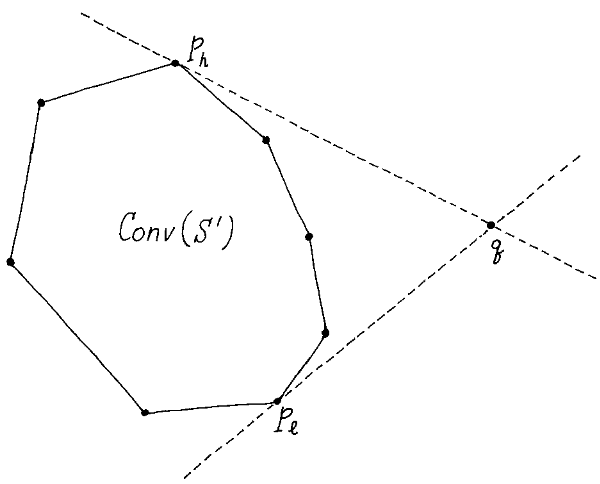


Figura 4.6: Algoritmo incremental.

Denotaremos esta cadeia por $p_\ell, p_{\ell+1}, \ldots, p_h$, onde todos os vértices $p_{\ell+1}, p_{\ell+2}, \ldots, p_{h-1}$ e as arestas neles incidentes devem ser eliminados. Portanto, basta determinarmos os pontos p_ℓ e p_h. Note que $p_\ell q$ determina uma reta de suporte para S, assim como qp_h.

A partir do vértice p_j de $\mathrm{Conv}(S')$ mais próximo de q, um caminhamento no sentido anti-horário sobre os vértices de $\mathrm{Conv}(S')$ nos permite determinar p_h da seguinte maneira. Enquanto p_{j+1} está à direita do raio de origem em q passando por p_j, isto é, enquanto $\Delta(p_j, q, p_{j+1}) > 0$, avança-se p_j. Quando p_j não puder mais ser avançado sem deixar de satisfazer esta propriedade, teremos encontrado p_h. Observe que, de fato, p_h assim obtido determina a reta suporte de $\mathrm{Conv}(S')$ passando por q. De maneira similar se determina p_ℓ. O passo restante consiste em substituir a cadeia $p_\ell, p_{\ell+1}, \ldots, p_h$ de $\mathrm{Conv}(S')$ por p_ℓ, q, p_h obtendo $\mathrm{Conv}(S)$.

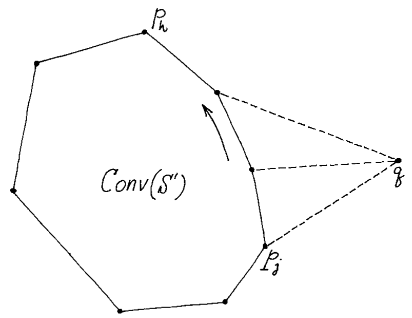


Figura 4.7: Determinação da reta de suporte superior de Conv(S') passando por q.

Complexidade

Complexidade: $O(n^2)$

Os passos 1 e 2 tomam tempo constante. Pelo detalhamento do passo 4.2.1, vemos que ele pode ser realizado em tempo linear na cardinalidade $|S| = n$. Logo, a relação de recorrência que descreve a complexidade do algoritmo é:

$$\begin{cases} T(n) \leq T(n-1) + an \text{ para } n > 3 \\ T(3) = b \end{cases} \tag{4.1}$$

onde a e b são constantes. Portanto, o tempo gasto pelo algoritmo 4.2 é da ordem de $O(n^2)$, superando, assim, a eficiência dos anteriores.

Varredura Planar

Perceba que se no passo 2 tomamos q como o ponto mais à direita de S, o algoritmo resultante é um algoritmo de varredura planar. Para que se possa fazer a escolha de q de maneira eficiente neste caso, deve-se pré-ordenar o conjunto S pelas abcissas de seus pontos, como discutimos na seção 1.6.1.

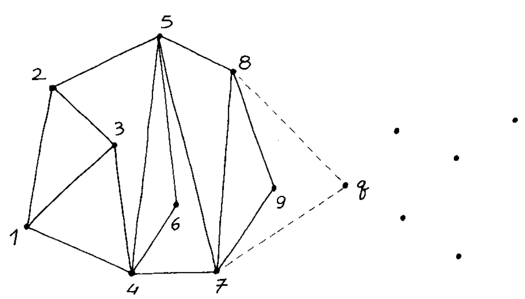


Figura 4.8: Construção incremental em ordem crescente de abcissas

A complexidade resultante depende crucialmente do processo de determinação da cadeia de vértices e arestas a ser eliminada do casco convexo intermediário (e da eliminação propriamente dita). Adiaremos a análise detalhada de uma maneira eficiente de realizar estas ações até mais adiante, mas o leitor ansioso é convidado a projetar e analisar a complexidade de um algoritmo por varredura planar baseado nestas observações.

4.3.2 Cota inferior

Convém analisarmos o problema em estudo de modo a determinar uma cota inferior que descreva sua complexidade intrínseca.

Será que se pode fazer melhor?

Como discutimos na seção 1.5, todo algoritmo que utiliza o teste de orientação de triângulo pressupõe que o modelo computacional admita produtos entre valores da entrada. Logo, tais algoritmos não são representáveis nos modelos de decisões binárias ou de decisões lineares.

Portanto, qualquer prova de cota inferior para o problema da construção da envoltória convexa deve ser feita dentro de um modelo pelo menos tão forte quanto o de árvore de decisões quadráticas. Consideraremos o modelo de decisões algébricas, já que este inclui todas as operações necessárias.

O leitor familiar com a demonstração de que o problema da ordenação requer pelo menos $\Omega(n \log n)$ comparações no modelo de árvore de decisões binárias não terá dificuldade de ver que a mesma prova vale também para o modelo de árvore de decisões algébricas.

Logo, para estabelecermos $\Omega(n \log n)$ como cota inferior para o problema da construção do casco convexo, basta reduzirmos a este, o problema da ordenação, conforme seção 1.7. Isso se pode fazer construindo-se (em tempo linear) uma instância do problema do casco convexo a partir de uma instância arbitrária do problema da ordenação de modo que se possa obter (em tempo linear) a solução desta a partir da solução daquela.

Dada uma seqüência S de n números reais (ou mesmo inteiros) $x_0, x_1, \ldots, x_{n-1}$, considere o conjunto $S = \{(x_0, x_0^2),\ (x_1, x_1^2), \ldots, (x_{n-1}, x_{n-1}^2)\}$ de pontos sobre a parábola $y = x^2$. Em primeiro lugar, note que todos os pontos de S são vértices de sua envoltória conve-

Cota inferior: $\Omega(n \log n)$

xa Conv(S). Ademais, percorrendo-se (duas vezes) a seqüência dos vértices de Conv(S) podemos produzir a seqüência S ordenada.

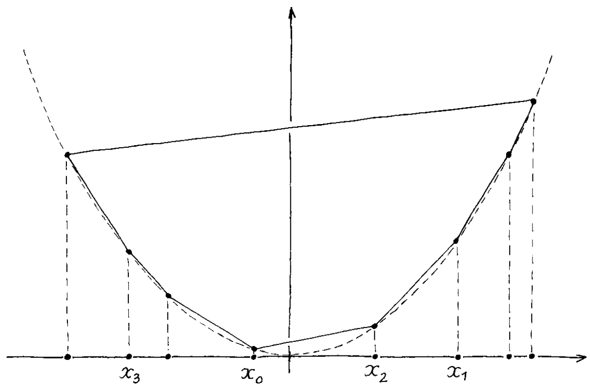


Figura 4.9: Redução de ordenação para casco convexo

Isso mostra que qualquer algoritmo que constrói envoltórias convexas pode ser usado para ordenar e, portanto, fica estabelecida a cota inferior $\Omega(n \log n)$ para o problema da construção do casco convexo de um conjunto de n pontos em $\mathbb{T}^{\nvDash}$, no modelo de árvore de decisões algébricas.

Portanto, o melhor que podemos esperar obter é um algoritmo de complexidade $O(n \log n)$. Isto é precisamente o que realizaremos nas seções seguintes.

4.3.3 Envolvendo estrelas

Um polígono em forma de estrela reune suficiente estrutura para permitir a construção da envoltória convexa de seus vértices de maneira muito mais eficiente do que se estes fossem considerados apenas como um conjunto desestruturado de pontos.

Envoltória convexa de um polígono em forma de estrela

Eliminação de ângulos reflexos

O trabalho de transformar um polígono em forma de estrela na envoltória convexa de seus vértices consiste essencialmente em eliminar os vértices nos quais os ângulos internos são reflexos.

Pela definição de polígono convexo, um polígono simples $P = (p_0, p_1, \ldots, p_{n-1})$ é convexo se e somente se $\Delta(p_{i-1}, p_i, p_{i+1}) = +1$ para $0 \leq i \leq n-1$.

O algoritmo que vamos descrever baseado nesta definição aparece na literatura já no artigo pioneiro de R. Graham [Gra72], e por essa razão ele é conhecido por varredura de Graham.

Observação 4.2 *Se um polígono estrelado não está contido num hemisfério, seu casco convexo não está definido pois não existe nenhum conjunto convexo que o contenha. A detecção desta situação é um aspecto importante do problema e pode ser tratada dentro de cada algoritmo que constrói a envoltória convexa. No entanto, nosso objetivo nesta seção é o de construir cascos convexos apenas de polígonos estrelados cujos vértices estão todos contidos num hemisfério, por isso, consideraremos que os polígonos em forma de estrela tratados aqui estão contidos no aquém.*

Considere um polígono em forma de estrela $P = (p_0, p_1, \ldots, p_{n-1})$. Se p é um dos vértices de P, denotaremos por $\mathrm{suc}(p)$ o sucessor de p e por $\mathrm{pred}(p)$ o seu predecessor na ordem em que os vértices são dados.

A idéia é percorrer os vértices de P, começando nalgum vértice que se saiba que pertencerá à envoltória convexa, de forma circular, verificando se cada seqüência de três vértices consecutivos forma uma curva para a esquerda ou para a direita. No primeiro caso, se prossegue e no segundo, elimina-se pelo menos um vértice.

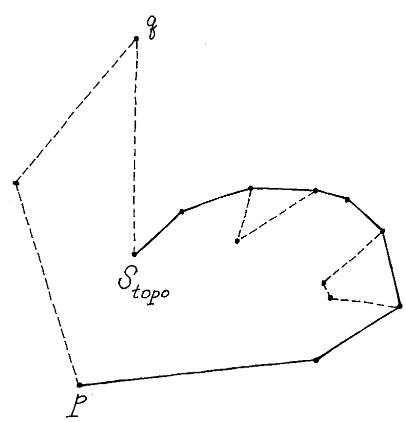


Figura 4.10: Ângulos reflexos devem ser eliminados

Durante este percurso, mantém-se uma pilha $\mathcal{S}$ dos vértices que formam o casco convexo dos pontos processados até o momento. Vértices que venham a ser encontrados mais tarde neste processo podem forçar a eliminação de alguns dos que se encontram na pilha devido à detecção de ângulos reflexos. Ao final, a pilha contém os vértices (na ordem inversa da que queremos!) do casco convexo de P. Denotemos por $\mathcal{S}_{topo}$ o ponto no topo da pilha $\mathcal{S}$ e por $\mathcal{S}_{topo-1}$ o ponto imediatamente abaixo de $\mathcal{S}_{topo}$ em $\mathcal{S}$.

Algoritmo 4.3 Varr-Graham(P)
Dado um polígono estrelado P com n vértices, devolve o casco convexo de P.

1. Dentre os vértices de maior abcissa, encontre o de menor ordenada; chame-o p.
2. Empilhe $(\mathcal{S}, p)$; empilhe $(\mathcal{S}, \mathrm{suc}(p))$.
3. $q \leftarrow \mathrm{suc}(\mathrm{suc}(p))$.
4. Enquanto $q \neq \mathrm{suc}(p)$ repita
   - 4.1. Se $\Delta(\mathcal{S}_{topo-1}, \mathcal{S}_{topo}, q) > 0$ então
     - 4.1.1. Empilhe $(\mathcal{S}, q)$;
     - 4.1.2. $q \leftarrow \mathrm{suc}(q)$
   - 4.2. Senão desempilhe $(\mathcal{S})$.
5. Devolva como Conv(P) os vértices de $\mathcal{S}$ na ordem inversa da que se encontram na pilha.

Complexidade

Complexidade: $O(n)$

Observe que tanto a determinação do vértice inicial p (passo 1) quanto o percurso (passo 4) são procedimentos realizáveis em tempo linear no número de vértices de P, já que cada vértice é empilhado exatamente uma vez.

É importante observar que a simplicidade do polígono resultante é decorrente do fato de que o polígono original P era *estrelado*. Se P fosse apenas um polígono simples não estrelado, o procedimento falharia, como mostra o exercício seguinte:

Ex. 4.4:

(a) Encontre um polígono simples *não estrelado* para o qual o procedimento acima *produz* a envoltória convexa.

(b) Encontre um polígono simples *não estrelado* para o qual o procedimento acima *não produz* a envoltória convexa.

(c) Mostre que o menor polígono que serve de solução para o item (b) tem seis vértices.

4.3.4 Fazendo estrelas

Em vista do que apresentamos no início desta seção, para que possamos obter um algoritmo subquadrático para construção do casco convexo de um conjunto S de n pontos do aquém, basta apenas que obtenhamos um algoritmo de complexidade $o(n^2)$ para construção de um polígono em forma de estrela cujos vértices sejam os pontos de S. O seguinte algoritmo atinge esse objetivo.

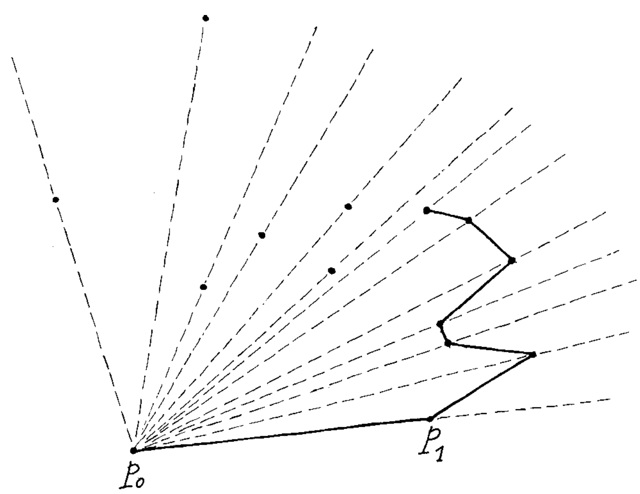


Figura 4.11: Construção de polígono em forma de estrela

Algoritmo 4.4 Estr(S)
Dado um conjunto S de n pontos, devolve um polígono estrelado cujo conjunto de vértices é S.

1. Escolha dois pontos distintos p_0 e p_1 quaisquer em S.
2. Ordene os demais pontos angularmente em relação a p_0 e $p_0 \vee p_1$.
3. Devolva o polígono cujos vértices são todos os pontos de S, na ordem obtida no passo anterior, começando e terminando em p_0.

Ordenação induz estrutura

A demonstração de que o vértice p_0 escolhido no primeiro passo deste algoritmo pode ser arbitrário é motivo do exercício 4.5.

Ex. 4.5: O propósito do algoritmo 4.4 é a construção de um polígono em forma de estrela cujos vértices são os pontos de um dado conjunto finito S.

(a) Mostre que, no primeiro passo do algoritmo, a escolha do ponto p_0 do conjunto S em torno do qual a ordenação circular dos outros pontos é realizada pode ser arbitrária. Isto é, qualquer que seja o ponto central da ordenação o resultado do algoritmo é sempre um polígono em forma de estrela.

(b) Mostre que podemos também escolher como ponto central qualquer ponto em $\mathbb{T}^{\not\models}$ que seja uma combinação convexa de pelo menos 2 pontos de S.

(c) O que acontece se escolhermos como ponto central da ordenação circular um ponto que seja uma combinação linear *não convexa* de pontos de S?

Como o leitor já deve suspeitar, a ordenação angular pode ser realizada sem a necessidade de cálculo explícito de ângulos. Veja exercício 4.6.

Ex. 4.6: Descreva um procedimento para realizar a ordenação angular do passo 2 do algoritmo 4.4 usando apenas o teste de orientação de triângulos.

Complexidade

Complexidade: $O(n \log n)$

A complexidade do algoritmo 4.4 é essencialmente devida ao passo 2 de ordenação que pode ser realizado em tempo $O(n \log n)$, o que é ótimo.

Otimalidade do algoritmo

Casco convexo em tempo ótimo: $\Theta(n \log n)$

Nas duas seções anteriores vimos que, dado um conjunto S de n pontos no aquém:

- podemos construir em tempo $O(n \log n)$ um polígono estrelado P tendo todos os pontos de S como vértices;

- a partir de um polígono estrelado é possível obter em tempo linear o seu casco convexo;

- $\Omega(n \log n)$ é uma cota inferior para a construção do casco convexo de um conjunto de n pontos em $\mathbb{T}^{\nvDash}$, no modelo de árvore de decisões algébricas.

Isso estabelece um algoritmo assintoticamente ótimo para a determinação da envoltória convexa de conjuntos finitos de pontos no plano projetivo orientado.

Varredura: uma questão de ótica

Vejamos agora como podemos transformar os procedimentos que levaram à construção de casco convexo em tempo ótimo em um algoritmo que utiliza o paradigma de varredura planar.

Seja, ainda, S um conjunto de n pontos no aquém. Chamemos de p_{min} e p_{max} os pontos de S de menor e de maior abcissa, respectivamente. Considere o subconjunto S^+ de S formado pelos pontos que estão do lado não-negativo da reta $p_{min} \vee p_{max}$. Similarmente, denote por S^- o conjunto dos pontos de S do lado não-positivo de $p_{min} \vee p_{max}$. Observe que o casco convexo de S pode ser obtido da união de $\mathrm{Conv}(S^+)$ e $\mathrm{Conv}(S^-)$ cuja única intersecção são os pontos p_{min} e p_{max}. Portanto, podemos construir $\mathrm{Conv}(S^+)$ e $\mathrm{Conv}(S^-)$ independentemente.

Imagine um ponto p_∞ de abcissa $p_\infty.x = (p_{min}.x + p_{max}.x)/2$ e ordenada $p_\infty.y = -\infty$, em torno do qual ordenamos angularmente os pontos de S^+. Esta ordenação corresponde exatamente à ordenação dos pontos de S^+ por suas abcissas.

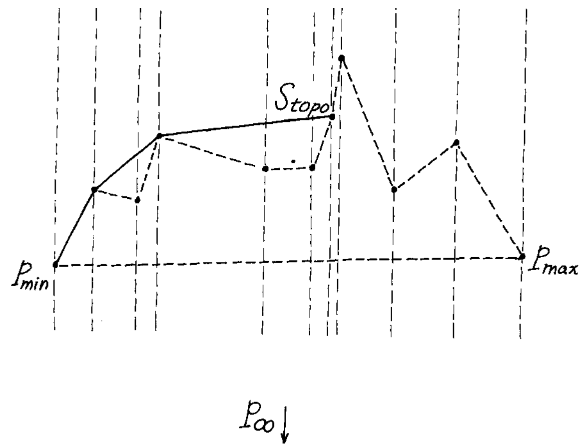


Figura 4.12: Construção da envoltória convexa por varredura.

Se construímos, conforme o passo 2 do algoritmo 4.4, o polígono estrelado P cujo conjunto de vértices é S^+, a aplicação do algoritmo 4.3 (de varredura de Graham) produz o casco convexo de S^+. Similarmente, podemos obter $\mathrm{Conv}(S^-)$.

Perceba que este procedimento é precisamente um de varredura planar no sentido descrito na seção 1.6.1, se desprezamos o artifício do ponto p_∞ e consideramos apenas que, uma vez ordenados os pontos de S^+, o algoritmo da varredura de Graham consiste de visitar os pontos de S^+ em seqüência processando-os de modo a eliminar ângulos reflexos.

A complexidade do algoritmo ainda é a mesma descrita anteriormente, isto é, $O(n \log n)$, e temos assim, um algoritmo ótimo baseado na técnica de varredura planar, só que apresentado como uma variação de um outro algoritmo de varredura angular.

4.3.5 Construção por divisão e conquista

Vamos agora utilizar outro importante paradigma de projeto de algoritmos, para apresentarmos mais um procedimento (também ótimo) para construção de casco convexo.

Por divisão e conquista

O algoritmo pelo método incremental que vimos na seção 4.3.1 para construção da envoltória convexa de um conjunto de pontos consiste essencialmente de reunir um ponto de cada vez a um casco convexo previamente construído.

Uma idéia semelhante é a de reunir *dois* cascos convexos parciais (disjuntos) de modo a formar o casco convexo da união de seus vértices. Isso nos leva à aplicação da técnica de *divisão e conquista* para projeto de um novo algoritmo.

Casco convexo por divisão e conquista

Ao invés de fazermos uma divisão do conjunto de pontos em dois subconjuntos arbitrários, usaremos uma divisão que produza dois subconjuntos separados por uma reta vertical mediana nas abcissas dos pontos. (Chamaremos E o do lado não negativo e D o do lado não positivo.)

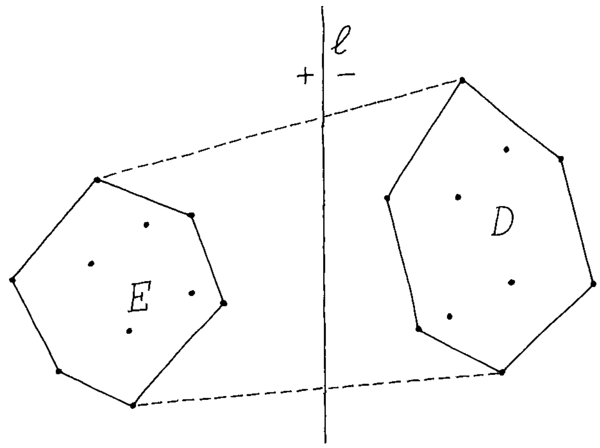


Figura 4.13: Casco convexo por divisão e conquista

Deste modo, garantimos que no passo de conquista teremos dois polígonos convexos disjuntos a serem processados. Este processamento consiste da construção de duas arestas de suporte (semelhante a como fizemos no caso do algoritmo incremental) e da eliminação dos vértices internos ao novo polígono. O algoritmo 4.5 atinge este propósito.

Algoritmo 4.5 ConvD&C(S)
Dado um conjunto S de pontos, devolve o casco convexo de S.

1. Ordene os n pontos do conjunto S por abcissa.
2. Devolva o casco convexo construído pelo algoritmo ConvD&C-Aux(S) abaixo.

Algoritmo 4.6 ConvD&C-Aux(S)

1. Se $|S| = 3$ então
   1.1. se $\Delta(p_0, p_1, p_2) > 0$ então devolva (p_0, p_1, p_2) senão devolva (p_2, p_1, p_0).
2. Divida os pontos em dois conjuntos E e D contendo os $\lfloor n/2 \rfloor$ pontos do lado não negativo e os $\lceil n/2 \rceil$ pontos do lado não positivo, respectivamente.
3. Construa os cascos convexos Conv(E)=ConvD&C-Aux(E) e Conv(D)=ConvD&C-Aux(D), recursivamente.
4. Intercale Conv(E) com Conv(D) formando Conv(S).
5. Devolva Conv(S).

Obviamente, o passo de intercalação requer detalhamento pois é ali que se dá a efetiva construção de novas arestas a cada passo de conquista. O algoritmo 4.7 a seguir determina o par de vértices dos conjuntos Conv(E) e Conv(D) que definem uma reta de suporte comum aos dois polígonos Conv(E) e Conv(D). Similarmente, deve-se determinar a aresta de suporte inferior e eliminar de cada um dos cascos convexos parciais anteriores os vértices interiores ao polígono Conv($E \cup D$) resultante, assim como as arestas neles incidentes.

Recorde-se que dado um vértice p de um polígono denotamos por suc(p) o vértice seguinte a p e por pred(p) o seu anterior na ordem em que os vértices são dados.

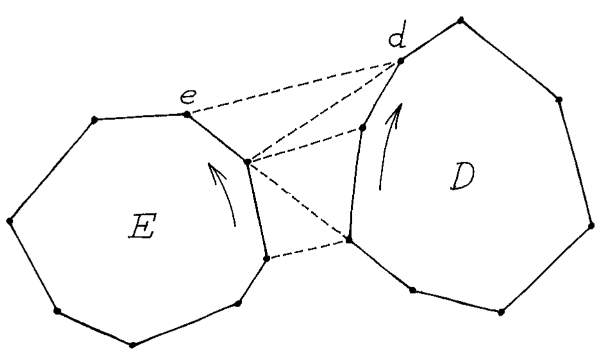


Figura 4.14: Construção da reta de suporte superior.

Algoritmo 4.7 Sup-Suporte(Conv(E),Conv(D))
Dados os cascos convexos Conv(E) e Conv(D) de dois conjuntos E e D, linearmente separáveis por uma reta ℓ, com E do lado não negativo de ℓ e D do lado não positivo de ℓ, devolve a aresta de suporte superior de Conv(E) e Conv(D).

1. Seja e um vértice de Conv(E) mais próximo de ℓ e d um vértice de Conv(D) mais próximo de ℓ.
2. Enquanto (e, d) não é reta de suporte superior tanto de Conv(E) quanto de Conv(D) repita
   - 2.1. Enquanto (e, d) não é reta de suporte superior de Conv(E) repita
     - 2.1.1. $e \leftarrow \mathrm{suc}(e)$.
   - 2.2. Enquanto (e, d) não é reta de suporte superior de Conv(D) repita
     - 2.2.1. $d \leftarrow \mathrm{pred}(d)$.
3. Devolva o par (e, d).

Determinação de reta de suporte de dois polígonos

Para determinar se (e, d), $e \in E$, $d \in D$, é reta de suporte superior de Conv(E) basta verificar se $\Delta(d, \mathrm{pred}(e), e) < 0$ e $\Delta(d, e, \mathrm{suc}(e)) > 0$, (isto é, o predecessor e o sucessor de e em Conv(E) devem estar ambos do lado positivo de $d \vee e$).

Complexidade

Complexidade: $\Theta(n \log n)$

Para se determinar a complexidade do algoritmo 4.7, note que cada vértice de uma sub-cadeia de Conv(E) é visitado exatamente uma vez. (Esta sub-cadeia é precisamente aquela que deve ser eliminada durante a intercalação de Conv(E) e Conv(D).) Portanto, o número de operações realizadas não ultrapassa $O(n)$.

Estabelecida a eficiência do procedimento que corresponde ao passo de conquista do algoritmo 4.6, podemos expressar a complexidade deste como a solução da relação de recorrência:

$$\begin{cases} T(n) \leq 2T(n/2) + an \text{ para } n > 3 \\ T(3) = b \end{cases} \tag{4.2}$$

onde a e b são constantes. Logo o tempo gasto pelo algoritmo 4.5 é da ordem de $O(n \log n)$, sendo, portanto, também ótimo.

4.3.6 E quando o casco convexo não existe?

Prometemos anteriormente esclarecer como determinar se o casco convexo de um conjunto finito de pontos é definido. A dificuldade da questão é que mesmo um conjunto sem pontos antipodais pode produzir um polígono (simples) contendo pares de pontos antipodais tanto em seu lado positivo quanto em seu lado negativo.

Qual é então uma caracterização dos conjuntos para os quais isso ocorre? Obviamente, pontos antipodais pertencem a hemisférios complementares e, ainda mais, exatamente um ponto dentre um par qualquer de antípodas pertence ao aquém e o outro ao além. Portanto, se uma região R de $\mathbb{T}^{\nvDash}$ contém pontos antipodais, o conjunto antipodal da parte de R contida no além tem que interceptar a porção de R contida no aquém.

Esta propriedade permite decidirmos se a envoltória convexa de um conjunto S de n pontos em $\mathbb{T}^{\nvDash}$ é definida ou não. O seguinte algoritmo decide esta questão:

Algoritmo 4.8 Existe-Conv(S)
Dado um conjunto S de pontos em $\mathbb{T}^{\nvDash}$, devolve "verdadeiro"se o casco convexo de S é definido e "falso"no caso contrário.

1. Seja S_+ (S_-) o conjunto de todos os pontos de S de peso positivo (negativo).
2. Seja $\neg S_-$ o conjunto antipodal de S_-.
3. Construa Conv(S_+) e Conv($\neg S_-$).
4. Se Conv(S_+) $\cap$ Conv($\neg S_-$) $\neq \emptyset$ então devolva FALSO, senão devolva VERDADEIRO.

O algoritmo usado no passo 3 pode ser qualquer um que construa casco convexo pois ambos os conjuntos S_+ e $\neg S_-$ têm envoltórias definidas.

A determinação da existência de intersecção entre Conv(S_+) e Conv($\neg S_-$) no passo 4 pode ser feita em tempo linear em $|S|$ e deixamos como um exercício para o leitor o projeto deste algoritmo.

Portanto, a aplicação de todos os algoritmos vistos neste capítulo para construção do casco convexo de um conjunto finito S de pontos pode ser feita assumindo que se decidiu antes que este casco é definido, já que esta decisão pode ser feita através da execução do algoritmo Existe-Conv(S) acima. A indefinição da envoltória convexa é indicada por uma mensagem de erro, quando apropriado.

Além disso, a menos de uma translação (realizável em tempo linear), podemos assumir que o conjunto de pontos cujo casco convexo se deseja construir está contido no aquém.

4.4 Extensões

Ilustramos alguns paradigmas de projetos de algoritmos na área de geometria computacional através da abordagem de alguns problemas relacionados com polígonos e, em particular, com a construção de envoltórias convexas. Dentre estes paradigmas, vimos:

- construção incremental,
- varredura planar,
- divisão e conquista.

Além disso, estabelecemos uma cota inferior não trivial para o problema da construção de cascos convexos e mostramos que ela pode ser atingida por algoritmos como aqueles baseados em varredura e divisão e conquista. Isso, no entanto, não encerra o estudo de algoritmos para este problema, pois há ainda muitas questões relevantes. Dentre elas, temos: análise comparativa de algoritmos para instâncias de tamanhos de interesse prático, manutenção de envoltória convexa na situação de conjuntos de pontos submetidos a alterações em tempo real (inserção, remoção), projeto de algoritmos aproximados com complexidade linear, etc.

4.4.1 Casco convexo de polígono simples

Vimos anteriormente (seção 4.3.3) que a estrutura de um polígono estrelado é suficiente para que se possa realizar a construção da envoltória convexa de seus vértices com uma quantidade linear de operações.

Casco convexo de polígono simples em tempo linear.

O casco convexo de um polígono simples (não necessariamente em forma de estrela), contido num hemisfério, também pode ser determinado em tempo linear por um método não muito mais complexo que o algoritmo de Graham descrito na seção 4.3.3. Um dos algoritmos para isso foi descoberto por D.T.Lee [Lee83]. Para uma descrição deste algoritmo assim como uma história de simplificações apresentadas por outros autores, veja [PS85].

Capítulo 5

Mapas

5.1 Introdução

O tema deste capítulo é a representação e manuseio de *mapas* no computador. Dada sua extensão, convém começar por uma breve sinopse de seu conteúdo.

5.1.1 Conceitos fundamentais

Informalmente, um mapa é uma divisão de coleção uma superfície — por exemplo, o plano ou a esfera — num número finito de regiões. Alguns exemplos típicos são um mapa do Brasil mostrando os estados, um globo terrestre mostrando os continentes e oceanos, o arranjo de faces e arestas na superfície de um poliedro, um logotipo multicolor, etc. Veja a figura 5.1.

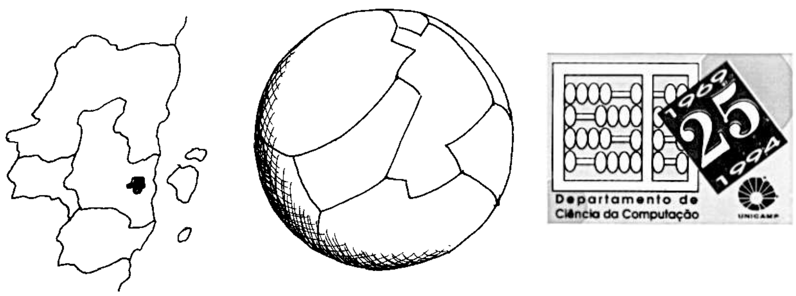


Figura 5.1: Exemplos de mapas.

Neste livro trataremos quase que exclusivamente de *mapas planares*, cuja superfície é o plano projetivo orientado $\mathbb{T}^{\nvDash}$. Dentro desta classe, estudaremos com mais detalhes a subclasse dos *mapas poligonais*, cujas arestas são segmentos de reta.

5.1.2 Aplicações

Polígonos generalizados

Mapas podem ser vistos como uma generalização dos conceitos de triângulo, polígono convexo, e polígono simples que estudamos nos capítulos anteriores. A justificativa para introduzir esse novo conceito geral é que muitas figuras geométricas que surgem na prática não se enquadram em nenhuma destas classes mais restritas. Por exemplo, o mapa de um país nem sempre pode ser representado por um polígono simples, pois ele pode ter ilhas, ou mesmo buracos (territórios internos pertencentes a outros países). Uma rede de estradas pode ter mais de duas linhas chegando na mesma cidade; uma parede de prédio pode ter janelas e portas; uma peça metálica estampada pode ter furos; o contorno de uma letra pode ter arcos de círculo ou curvas mais complexas; e assim por diante.

Aplicações práticas

Mapas são usados também para a modelagem de objetos tridimensionais em computação gráfica, robótica, e projeto industrial; para a modelagem de terrenos em geologia e topografia; no projeto e fabricação de circuitos integrados; na resolução numérica das equações diferenciais parciais da meteorologia, mecânica dos fluidos, e engenharia civil (método de elementos finitos); e muitas outras aplicações.

Ex. 5.1: Cite um país que tem pelo menos *dois* buracos no seu território.

5.1.3 Estruturas de dados

Estruturas de dados topológicas

Em muitas aplicações, as propriedades *topológicas* de um mapa (quais regiões são vizinhas entre si) podem ser mais importantes do que suas propriedades *geométricas* (a forma e o tamanho das regiões). Nesses casos, convém em geral representar um mapa por uma estrutura de dados ligada, onde as relações de vizinhança entre as regiões são explicitamente representadas por apontadores entre os registros correspondentes.

Tais apontadores também permitem melhorar a eficiência de certos algoritmos geométricos, como por exemplo localizar o estado que contém um ponto de latitude e longitude conhecidas. Além disso, o uso de apontadores geralmente permite economizar espaço, pois a fronteira comum ente duas regiões pode ser representada uma única vez.

5.1.4 Algoritmos

Quanto a algoritmos, este capítulo introduz dois paradigmas importantes para a manipulação de mapas planares: *superposição* de mapas e a *varredura* do plano.

Superposição

Muitos problemas geométricos envolvendo dois mapas planares podem ser resolvidos construindo-se primeiro um terceiro mapa que é meramente a superposição dos dois, e eliminando-se em seguida algumas linhas e regiões deste último, segundo certas regras simples. Com este método podemos facilmente calcular, por exemplo, a união, intersecção, e diferença de dois polígonos arbitrários.

Varredura

Num nível mais básico, o paradigma da varredura do plano é a base de algoritmos ótimos para várias operações com mapas planares, incluindo o cálculo da superposição de dois mapas, a decomposição de polígonos em triângulos ou quadriláteros, a localização de pontos num mapa, e muitos outros.

5.2 Conceitos fundamentais

O conceito informal de de mapa — uma superfície dividida em regiões — é vago e genérico demais para servir de base a algoritmos. Portanto, antes de mais nada precisamos formalizar um pouco mais esta definição.

Para este fim, e para outras definições que se seguem, vamos supor conhecidos certos conceitos elementares de topologia de conjuntos. Em particular, vamos usar os conceitos de *espaço topológico*, *conjunto aberto* e *fechado*, *sub-espaço*, *vizinhança* de um ponto, *ponto de acumulação* e *limite*, *fecho*, *interior*, e *fronteira* de um conjunto, e *continuidade* de uma função. Vamos também utilizar bastante o conceito de *homeomorfismo* (ou *equivalência*) entre dois espaços topológicos.

Leitores que tiveram algum contato com estes conceitos em cursos de análise real ou cálculo não devem ter dificuldade em acompanhar a discussão abaixo. Em todo caso, incluímos definições sucintas dos mesmos no fim deste capítulo (seção 5.9).

5.2.1 Arcos e discos

Arco aberto

Em topologia, define-se um *arco aberto* de um espaço topológico como sendo um conjunto de pontos equivalente ao intervalo aberto $(-1, +1)$.

Informalmente, um arco aberto é um pedaço único de reta ou de curva, que não se cruza e que não contém seus pontos extremos. Esta definição exclui portanto curvas sem pontas (por exemplo, um círculo), ou que se bifurcam.

Disco aberto

Define-se também um *disco aberto* como sendo um conjunto de pontos equivalente ao interior do círculo unitário de $\mathbb{R}^{\nvDash}$. Informalmente, um disco aberto é um pedaço de superfície com uma única borda, que pode ser esticado e desdobrado, sem rasgar, até ficar inteiramente plano.

Ex. 5.2: Mostre que todo arco aberto é equivalente à reta $\mathbb{R}$, e que todo disco aberto é equivalente ao plano $\mathbb{R}^{\nvDash}$.

Ex. 5.3: Quais dos subconjuntos abaixo são arcos de $\mathbb{R}^{\nvDash}$? (Justifique informalmente as respostas.)

(a) $\{\,(t, t^2) \;:\; 0 < t < 1\,\}$
(b) $\{\,(t, t^2) \;:\; 0 < t \leq 1\,\}$
(c) $\{\,(\cos\theta, \sin\theta) \;:\; 0 < \theta < 2\pi\,\}$
(d) $\{\,(t, \sin(1/t)) \;:\; 0 < t < 1\,\}$
(e) $\{\,(t, 0) \;:\; 0 < t \leq 1\,\} \cup \{\,(0, t) \;:\; 0 < t < 1\,\}$
(f) $\{\,(t, 0) \;:\; 0 < t \leq 1\,\} \cup \{\,(\cos\theta, \sin\theta) \;:\; 0 < \theta < 2\pi\,\}$

Ex. 5.4: Mostre que todo disco aberto é equivalente ao quadrado aberto $(-1, +1) \times (-1, +1)$.

5.2.2 Superfície

Superfície

Por *superfície* entendemos o que é tecnicamente chamado de *variedade topológica bidimensional*: isto é, um espaço topológico, cada ponto do qual tem uma vizinhança que é um disco aberto.

Informalmente, uma superfície é um conjunto de pontos que é bidimensional ao redor de qualquer de seus pontos. O plano cartesiano $\mathbb{R}^{\nvDash}$, a esfera $\mathbb{S}^{\nvDash}$, e a superfície de um toro são exemplos típicos. Outros exemplos são: um hemisfério de $\mathbb{S}^{\nvDash}$ (excluindo o 'equador'), a superfície de um cilindro de comprimento infinito, uma fita de Möbius (excluindo os pontos na borda). Veja a figura 5.2.

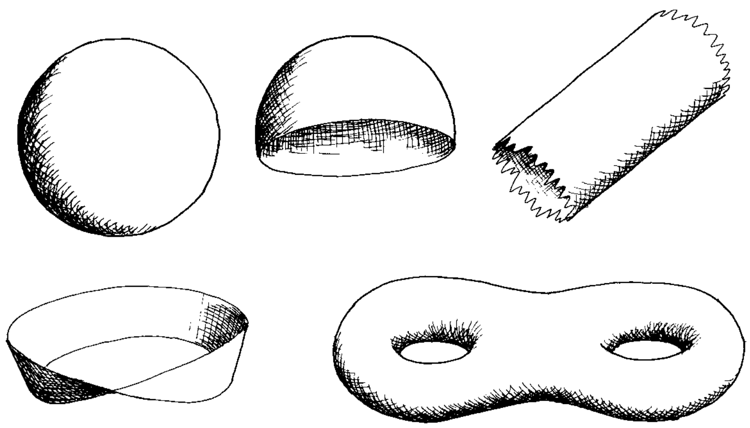


Figura 5.2: Exemplos de superfícies

Por outro lado, um hemisfério de $\mathbb{S}^{\nvDash}$, *incluindo* o equador, não é uma superfície; pois os pontos do equador não tem nenhuma vizinhança que seja equivalente a um disco aberto. Outros exemplos de conjuntos que não são superfícies, no nosso sentido, são: dois cones unidos pelo vértice, três semi-planos limitados pela mesma reta, ou uma esfera menos um arco aberto do equador.

5.2.3 Conjunto conexo

Em topologia, diz-se que um espaço X é *conexo* se não existe nenhum subconjunto próprio e não vazio Y de X que é ao mesmo tempo aberto e fechado. *Conjunto conexo*

No caso de X ser um subconjunto de uma superfície, isto equivale a dizer que é possível ir de qualquer ponto de X para qualquer outro ponto, por uma trajetória contínua, sem sair de X. (Mais precisamente, para quaisquer dois pontos u e v de X, existe uma função contínua f do intervalo fechado $[0,1]$ para X, tal que $f(0) = u$ e $f(1) = v$.)

As *componentes conexas* de um conjunto X são os seus subconjuntos conexos maximais. Ou seja, dois pontos de X estão na mesma componente se e somente se é possível ir de um par o outro segundo uma trajetória contínua em X.

5.2.4 Definição formal de mapa

Estamos finalmente em condições de definir o assunto deste capítulo:

Mapa e elemento

Definição 5.1 Um *mapa* sobre uma superfície S é uma partição de S numa coleção de subconjuntos conexos e não vazios — os *elementos* do mapa.

Note a palavra *partição*: ela significa que os elementos do mapa são disjuntos dois a dois, e cobrem completamente a superfície S. Ou seja, todo ponto da superfície pertence a exatamente um elemento do mapa.

Elemento que contém um ponto

Se M é um mapa sobre uma superfície S, e p é um ponto de S, denotaremos por $M(p)$ o único elemento de M que possui p. Mais geralmente, se X é um subconjunto de S que está contido num único elemento de M, denotaremos esse elemento por $M(X)$.

Ex. 5.5: Considere o mapa M sobre a esfera $\mathbb{S}^{\nvDash}$, onde dois pontos $[w, x, y]$ e $w', x', y']$ pertencem ao mesmo elemento de M se e somente se $\operatorname{sgn} w = \operatorname{sgn} w'$, $\operatorname{sgn} x = \operatorname{sgn} x'$ e $\operatorname{sgn} y = \operatorname{sgn} y'$. Descreva explicitamente os elementos de M.

5.2.5 Mapas equivalentes

Mapas equivalentes

Dizemos que dois mapas M e N são *equivalentes* (*topologicamente*) se existir um homeomorfismo entre as duas superfícies dos mesmos, tal que a imagem de cada elemento de M é um elemento de N.

Topologia de um mapa

Uma *propriedade topológica* de superfícies, mapas, pontos, ou conjuntos de pontos é uma afirmação que pode ser definida usando apenas álgebra de conjuntos e os conceitos de conjunto aberto e fechado. Ou seja, é uma propriedade que não é afetada se substituirmos cada superfície S por uma superfície homeomorfa S', e cada ponto ou conjunto de pontos de S pelo seu correspondente em S'.

A *topologia* de um mapa M sobre uma superfície S é o conjunto de todas as propriedades topológicas de S e M. Podemos portanto dizer que dois mapas equivalentes se eles têm a *mesma topologia*.

5.2.6 Incidência

Incidência entre elementos

Um exemplo de propriedade topológica são as *relações de incidência* entre elementos de um mapa. Diremos que um elemento a de um mapa M *incide* em outro elemento b, denotado por $a \to b$, se a fronteira de a contém algum ponto de b.

Ou seja, $a \to b$ se b contém um ponto de acumulação de a. No plano $\mathbb{R}^{\nvDash}$, por exemplo, o segmento aberto de entre $(0, 0)$ e $(1, 1)$ incide nesses dois pontos; e o semiplano $y > 0$ incide nos dois pontos e no segmento.

5.3 Mapas especiais

A definição de mapa que demos na seção 5.2.4 é geral demais para nossos propósitos. Por exemplo, ela permite um número infinito de elementos arbitrariamente complicados, em superfícies arbitrariamente complicadas.

Portanto, se quisermos representar e manipular mapas no computador, temos que limitar nossa atenção a alguma subclasse de mapas onde o tipo de superfície, a forma dos elementos, e as relações de incidência entre eles são suficientemente restritas.

Com esta motivação, definiremos a seguir algumas subclasses de mapas que são importantes na prática: os *mapas simples*, *algébricos*, *planares*, e *poligonais*.

5.3.1 Superfície compacta

Superfície compacta

Em primeiro lugar, convém trabalhar apenas com mapas sobre superfícies *compactas*. Formalmente, diz-se que um espaço topológico S é *compacto* se e somente se todo subconjunto infinito de S tem pelo menos um ponto de acumulação em S. Informalmente, um espaço é compacto se ele tem extensão finita e não tem beiradas “abertas’, de modo que é impossível andar para sempre sem passar infinitamente perto de algum ponto do espaço.

Exemplos de superfícies compactas são a esfera $\mathbb{S}^{\nvDash}$ (e, portanto, o plano projetivo orientado $\mathbb{T}^{\nvDash}$) e o toro. Por outro lado, o plano cartesiano $\mathbb{R}^{\nvDash}$ não é compacto, pois o conjunto dos pontos da forma $(i, 0)$, com i inteiro, é infinito mas não tem nenhum ponto de acumulação em $\mathbb{R}^{\nvDash}$. Outros exemplos de superfícies não compactas são um hemisfério de $\mathbb{S}^{\nvDash}$ excluindo o equador (que, aliás, é homeomorfo a $\mathbb{R}^{\nvDash}$), ou um cilindro infinito.

Entre outras qualidades, uma superfície compacta tem automaticamente um número finito de componentes, e um número finito de "alças" (tecnicamente, têm gênus finito.)

5.3.2 Mapas simples

Podemos então definir a classe que nos interessa:

Mapa simples

Definição 5.2 Um *mapa simples* é um mapa sobre uma superfície S tal que

(1) o número de elementos é finito;

(2) cada elemento é

Vértice, aresta, face

um *vértice*: um único ponto de S, ou
uma *aresta*: um arco aberto de S, ou
uma *face*: um disco aberto de S.

(3) a superfície S é compacta;

(4) a fronteira de toda aresta é um conjunto de vértices;

(5) todo vértice está na fronteira de alguma aresta.

A condição (1) nos permite representar cada elemento individualmente, por um registro de uma estrutura de dados. A condição (2) diz que há três tipos de elementos, com topologias bem definidas. A condição (3) garante que os dois extremos de cada aresta existem e são pontos de S, e que o mesmo ocorre com o perímetro completo de cada face. A condição (4) impede que a ponta de uma aresta termine no meio de outra; e a condição (5) exclui vértices isolados.

Como veremos, estas condições todas implicam que a topologia do mapa é inteiramente determinada pelas relações de incidência e ordem entre os elementos.

A figura 5.3 mostra três mapas simples. As superfícies são (a) a esfera, (b) o toro, e (c) a garrafa de Klein. Esta última não pode ser construída no espaço $\mathbb{R}^{\nvDash}$; o gargalo da garrafa precisa passar do lado de fora para o lado de dentro da garrafa sem cruzar a superfície da mesma, o que só pode ser conseguido passando-se por uma 4ª dimensão.

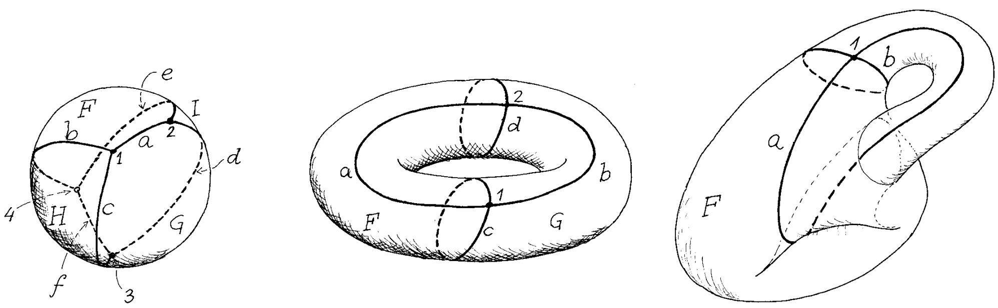


Figura 5.3: Três mapas simples.

A figura 5.4 é outra representação dos mapas da 5.3. Neste modelo, cada face do mapa é representada por um polígono, cujos lados devem ser "colados" dois a dois para formar as arestas e fechar superfície. Ao colar dois lados, eles devem ser justapostos de forma que as setas coincidam — cabeça com cabeça, cauda com cauda.

Ex. 5.6: Quais são os números mínimos de arestas, vértices, faces, e elementos de um mapa simples em $\mathbb{T}^{\nvDash}$? Dê exemplos de mapas que atingem esses mínimos.

Ex. 5.7: Descreva um mapa simples sobre o bitoro (uma esfera com duas alças).

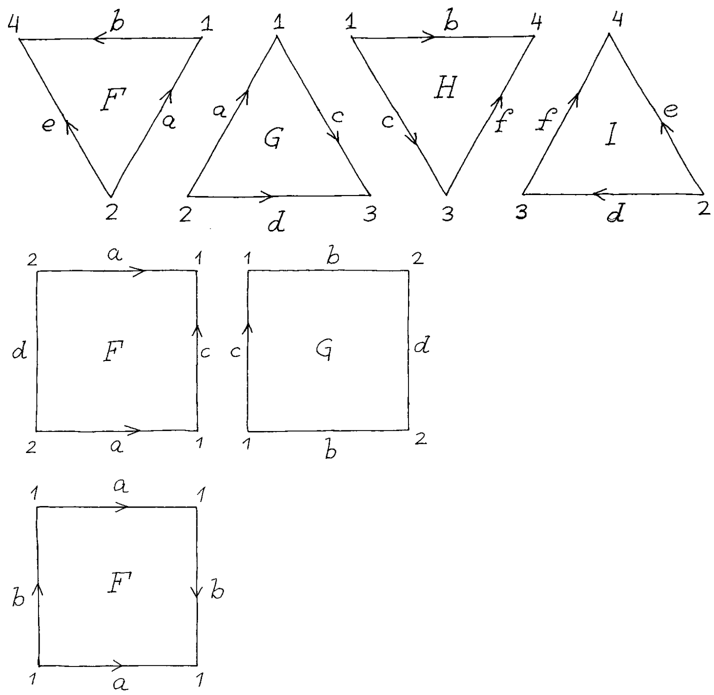


Figura 5.4: Os mapas da figura 5.3, no modelo de colagem de faces.

5.3.3 Mapas algébricos

A definição de mapa simples restringe apenas a topologia dos elementos. Vamos agora considerar restrições sobre a forma dos mesmos.

Conjunto algébrico

Dizemos que um subconjunto X de $\mathbb{R}^{\rceil}$ é *algébrico* se ele pode ser obtido por meio de uniões, intersecções, e complementos de conjuntos da forma

$$\{\, (x_1, .. x_k) \; : \; f(x_1, .. x_k) = 0 \,\} \tag{5.1}$$

ou

$$\{\, (x_1, .. x_k) \; : \; f(x_1, .. x_k) < 0 \,\} \tag{5.2}$$

onde as funções f_i são polinômios com coeficientes inteiros[1].

[1]Esta nomenclatura é exclusivamente nossa; o nome "oficial" para este conceito é *variedade semi-algébrica*.

Ex. 5.8: Prove que o lado positivo de um polígono convexo é um subconjunto algébrico de $\mathbb{R}^{\nvDash}$. Idem para um polígono simples.

Ex. 5.9: Prove que um subconjunto X de $\mathbb{T}^{\nvDash}$ é algébrico se e somente se ele consiste de pontos $[w, x, y]$ tais que $f(w, x, y) = \mathbf{true}$, sendo que f é uma fórmula envolvendo apenas as operações de soma, subtração, e multiplicação, as comparações $=$ e $<$, e as operações lógicas **e**, **ou**, **não**.

Superfície algébrica

Em particular, uma *superfície algébrica* é um subconjunto algébrico $S \subseteq \mathbb{R}^{\urcorner}$ (para algum k) que é uma superfície, no sentido topológico. A esfera unitária $\mathbb{S}^{\nvDash}$ de $\mathbb{R}^{\nVDash}$ é um exemplo típico de superfície algébrica. Outro exemplo é a fronteira de qualquer poliedro limitado de $\mathbb{R}^{\nVDash}$, por exemplo um cubo.

Ex. 5.10: Seja r um numero positivo menor que 1, e considere o subconjunto $T \subseteq \mathbb{R}^{\nVDash}$ definido por

$$T = \{\, ((1 + r\cos\theta)\cos\phi, (1 + r\cos\theta)\sin\phi, r\sin\theta) \;:\; 0 \leq \theta < 2\pi \,\}$$

Descreva a forma deste conjunto, e prove que ele é uma superfície algébrica. (Dica: observe que se $(x, y, z) \in T$, então $x^2 + y^2 + z^2 = 1 + r^2 + 2r\cos\theta$.)

Ex. 5.11: Considere o subconjunto $S \subset \mathbb{R}^{\ntrianglerighteq}$ definido por

$$S = \left\{ (u, v, x, y) \;:\; (u^2 + v^2 - 1)^2 + (x^2 + y^2 - 1)^2 = 0 \right\}$$

Mostre que ele é topologicamente equivalente ao toro T do exercício 5.10. (Dica: considere a correspondência $u = \cos\theta$, $v = \sin\theta$, $x = \cos\phi$, $y = \sin\phi$.)

Subconjunto algébrico de $\mathbb{T}^{\nvDash}$

Por extensão, diremos que o plano projetivo orientado $\mathbb{T}^{\nvDash}$ também é uma superfície algébrica; e que um subconjunto X de $\mathbb{T}^{\nvDash}$ é *algébrico* se, no modelo esférico, ele corresponder a um subconjunto algébrico da esfera $\mathbb{S}^{\nvDash}$.

Mapa algébrico

Com esses conceitos, podemos então definir um *mapa algébrico* como sendo um mapa sobre uma superfície algébrica cujos elementos são conjuntos algébricos.

5.3.4 Mapas planares e poligonais

Muitos mapas que aparecem na prática são mapas sobre o plano $\mathbb{R}^{\nvDash}$. Como $\mathbb{R}^{\nvDash}$ não é compacto, se quisermos utilizar estruturas de dados como a descrita acima precisamos "completar" $\mathbb{R}^{\nvDash}$ de alguma forma; ou seja, usar uma superfície compacta que contém $\mathbb{R}^{\nvDash}$ como sub-espaço.

Mapa planar

Neste livro, usaremos para esse fim o plano projetivo de dois lados $\mathbb{T}^{\nvDash}$. Definimos portanto um *mapa planar* como sendo um mapa sobre $\mathbb{T}^{\nvDash}$. Uma vantagem acidental dessa escolha é que muitos algoritmos para mapas planares podem ser aplicados também a mapas sobre a esfera $\mathbb{S}^{\nvDash}$.

Mapa poligonal

Uma subclasse importante dos mapas planares é a dos *mapas poligonais*, em que as arestas são segmentos de retas.

5.4 Aplicações

5.4.1 Mapas temáticos

Mapa temático

Um mapa que aparece num problema prático geralmente contém mais informação do que apenas a topologia e geometria dos elementos. Em muitos casos, essa informação adicional pode ser representada por uma coleção de "valores" ou "atributos" associados aos elementos. Emprestando a nomenclatura de cartografia, chamaremos esse mapa "decorado" de *mapa temático*.

Mais formalmente, definimos um mapa temático como sendo um par (A, v), onde A é um mapa, e v é uma função que a cada elemento e de A associa um *valor* $v(e)$, pertencente a um conjunto arbitrário Y.

Um exemplo de mapa temático é um mapa político dos estados do Brasil, onde o valor $v(f)$ de uma face f é a sigla do estado: **SP**, **MG**, etc.. Neste exemplo, os valores das arestas e vértices são irrelevantes, e podem ser qualquer valor "nulo" — por exemplo, a cadeia **??**.

Outro exemplo de mapa temático é um mapa rodoviário, onde as arestas são trechos de estrada e os vértices são cruzamentos. O valor de uma aresta pode então ser o código da estrada correspondente, e valor de um vértice pode ser o nome da cidade vizinha. Neste exemplo, as faces do mapa (regiões limitadas por estradas) não são importantes, e seu valor v_A pode ser um valor nulo qualquer **??**.

Outro exemplo ainda é a descrição de um desenho policromático (logotipo, cartaz, rótulo, capa de livro, etc.), onde o valor de cada face é a cor da mesma, e os valores de vértices e arestas são irrelevantes.

5.4.2 Campos de valores

Campo de valores

Um mapa temático (A, v) pode também ser visto como um *campo de valores*, uma função f que associa a cada ponto p do plano o valor $f(p) = v(A(p))$, onde $A(p)$ é o elemento de A que contém p. Na verdade, um mapa temático é a representação natural para qualquer função f deste tipo; desde que o conjunto de valores possíveis de f seja finito, e desde que a imagem inversa de qualquer valor seja uma região suficientemente simples, cuja fronteira pode ser representada de maneira compacta.

Função característica

Em particular, todo subconjunto S de pontos do plano define um campo de valores χ_S, a *função característica* de S, que vale **true** num ponto p se $p \in S$, e **false** caso contrário.

Campo algébrico por partes

Um mapa temático pode também ser usado para representar um campo de valores f com contra-domínio infinito, desde que f seja *algébrico por partes*. Com isto queremos dizer que o plano $\mathbb{T}^{\nvDash}$ pode ser dividido num número finito de regiões, delimitadas por curvas algébricas, tais que dentro de cada região o valor de $f(p)$ é dado por uma única fórmula algébrica envolvendo as coordenadas de p.

Por exemplo, as regiões podem ser triângulos, sendo que o valor do campo num ponto (X, Y) é uma função de primeiro grau $aX + bY + c$, onde os coeficientes a, b, e c dependem da região. Veja a figura 5.5.

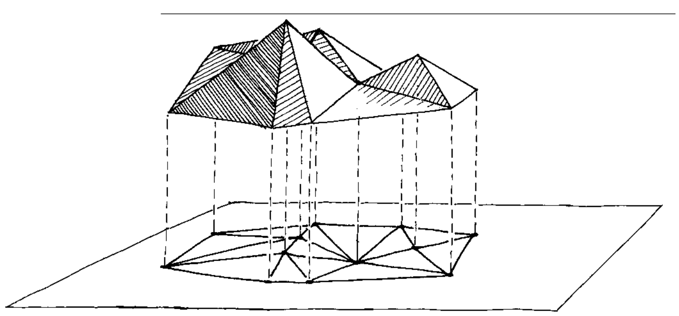


Figura 5.5: Um campo de valores linear por partes.

Para representar um campo de valores algébrico por partes, basta considerar o campo de valores F que, a cada ponto p, associa a *fórmula* (não o valor) de $f(p)$ válida na região que contém p. Obviamente, F tem contra-domínio finito (um conjunto finito de fórmulas), e portanto pode ser representado na forma de mapa temático.

Aplicações práticas

Esta representação de campos de valores é regularmente usada em geologia e topografia para representar o relevo de uma região; em engenharia civil, para representar as pressões e tensões dentro de uma estrutura; em mecânica dos fluidos, para representar o fluxo de líquidos e gases; em desenho industrial, para descrever o formato de chapas estampadas; e em muitas outras aplicações práticas.

Ex. 5.12: Seja f o campo de valores que a cada ponto $p = [w, x, y]$ de $\mathbb{T}^{\neq}$ associa o valor $f(p) = \operatorname{sgn}(x(x^2 + y^2 - w^2))$. Note que o contra-domínio de f é o conjunto $\{-1, 0, +1\}$. Descreva os vértices, arestas, e faces de um mapa temático que representa f.

5.5 Representação de mapas

Vejamos agora como descrever a topologia de um mapa no computador. Vamos nos limitar aqui a mapas simples, cuja topologia, como veremos, é determinada pelas relações de incidência e ordem de suas arestas.

5.5.1 Dardos

Orientação longitudinal e transversal

Para cada aresta a de um mapa simples podemos definir duas *orientações longitudinais*, que são os dois sentidos de percurso ao longo de a; e duas *orientações transversais*, que são as duas maneiras possíveis de definir o "lado esquerdo" e o "lado direito" da aresta, numa vizinhança suficientemente pequena da mesma.

Dardo

Um *dardo* de um mapa simples consiste de uma aresta com direções longitudinal e transversal especificadas. Podemos visualizar estas direções por um par de setas pequenas colocadas sobre um ponto qualquer da aresta: uma tangente à aresta, no sentido crescente da ordem dos pontos; e a outra perpendicular à primeira, no sentido do lado direito para o esquerdo. Outra maneira de visualizar estas direções é imaginar um observador minúsculo andando sobre a superfície, ao longo da aresta: a direção longitudinal diz em que sentido ele está caminhando, e a transversal diz em que lado da aresta está seu pé esquerdo (ou seja, sobre que lado da superfície ele está andando). Veja a figura 5.6.

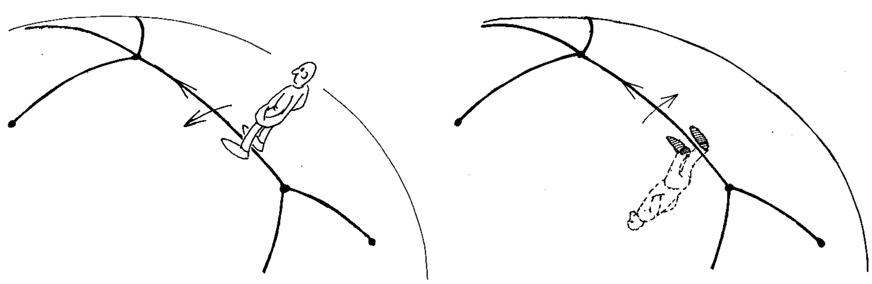


Figura 5.6: Dardos.

5.5.2 Funções ambulatórias

Para cada aresta de um mapa simples existem exatamente quatro dardos distintos. Note que isto é verdade mesmo que os dois extremos da aresta incidam no mesmo vértice, ou que os dois lados da aresta pertençam à mesma face.

Dardos opostos: Sim e Rev

Se d é um dardo, denotaremos por $d\,Sim$ o dardo que consiste da mesma aresta que d, mas com as duas orientações invertidas. Denotaremos também por $d\,Rev$ o resultado de inverter apenas a direção transversal do dardo d, mantendo a direção longitudinal. Veja a figura 5.7.

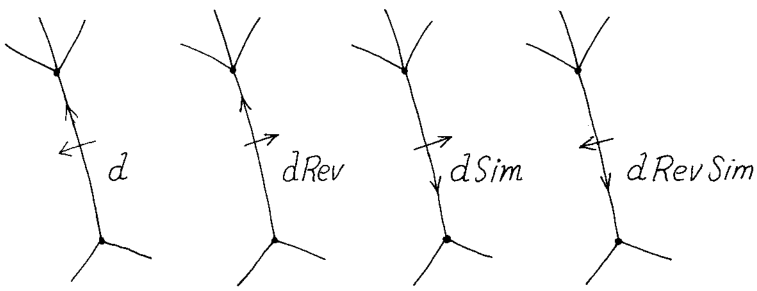


Figura 5.7: As funções *Rev* e *Sim*.

Vértices e faces de un dardo

Usando a direção longitudinal, podemos definir sem ambigüidade o *vértice de origem* e o *vértice de destino* de um dardo d, denotados respectivamente por $d\,Org$ e $d\,Dst$. Analogamente, usando a direção transversal, podemos definir a *face esquerda* e a *face direita* da aresta, denotadas por $d\,Esq$ e $d\,Dir$. Note que podemos ter $d\,Org = d\,Dst$, ou $d\,Esq = d\,Dir$.

Ex. 5.13: Simplifique estas expressões:

(a)	$d\,Sim\,Org$	(b)	$d\,Rev\,Org$
(c)	$d\,Sim\,Esq$	(d)	$d\,Sim\,Rev\,Esq$
(e)	$d\,Sim\,Rev\,Dst$	(f)	$d\,Rev\,Sim\,Dir$

Seja v a origem de um dardo d. A orientação transversal de d, considerada em pontos arbitrariamente próximos a v, define um sentido de rotação em torno de v. Considere todos os dardos com origem v que definem o mesmo sentido de rotação. Existe uma ordem cíclica natural para esses dardos, correspondente a esse sentido de rotação. Nesta ordem cíclica, podemos distinguir o *próximo dardo com mesma origem que* d, denotado por $d\,OProx$.

Próximo dardo com mesma origem: OProx

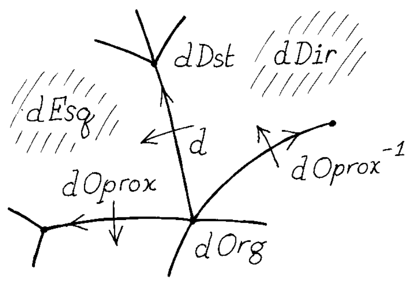


Figura 5.8: Funções ambulatórias.

A partir de qualquer dardo inicial d, podemos atingir todos os dardos na componente conexa do mapa que contém d, usando combinações apropriadas de *Rev*, *Sim*, d *OProx*. Por exemplo, podemos obter o *próximo dardo com mesma face esquerda* $d\,EProx$, no sentido anti-horário em torno da mesma, pela fórmula

Próximo dardo na face: EProx

$$d\,EProx = d\,Sim\,OProx^{-1}$$

Veja a figura 5.9.

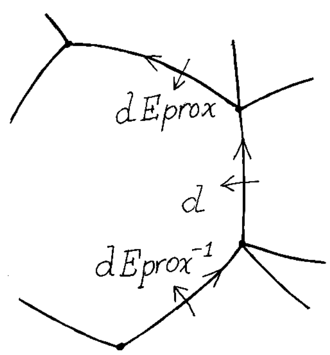
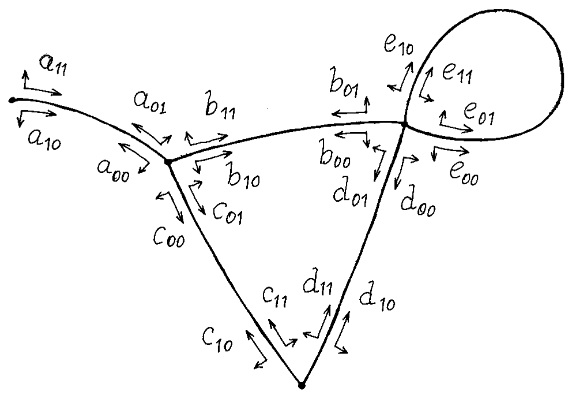


Figura 5.9: Percorrendo os dardos de uma face.

Exemplo A figura 5.10 é um exemplo de um mapa simples sobre a esfera, com 4 vértices, 5 arestas, e 3 faces. Nesta figura, os quatro dardos de uma aresta d são denotados por d_{lt}, onde l e t indicam respectivamente a orientação longitudinal e transversal; sendo que a orientação de d_{00} é escolhida arbitrariamente. A tabela lista os valores de *OProx* de todos esses dardos.



x	a_{00}	a_{01}	a_{10}	a_{11}
x *OProx*	c_{01}	b_{11}	a_{10}	a_{11}
x	b_{00}	b_{01}	b_{10}	b_{11}
x *OProx*	d_{00}	e_{11}	c_{00}	a_{00}
x	c_{00}	c_{01}	c_{10}	c_{11}
x *OProx*	a_{01}	b_{11}	d_{11}	d_{10}
x	d_{00}	d_{01}	d_{10}	d_{11}
x *OProx*	e_{01}	b_{01}	c_{11}	c_{10}
x	e_{00}	e_{01}	e_{10}	e_{11}
x *OProx*	d_{01}	e_{10}	b_{00}	e_{00}

Figura 5.10: Um mapa simples sobre a esfera.

Ex. 5.14: Enumere todos os dardos dos mapas simples ilustrados na figura 5.3, e construa a tabela da função *OProx* para os mesmos

5.5.3 Identidades ambulatórias

Note que, para todo dardo d, vale

Identidades fundamentais

(MS1) $d\,Rev \neq d$

(MS2) $d\,Rev\,Rev = d$

(MS3) $d\,Sim \neq d$

(MS4) $d\,Sim \neq d\,Rev$

(MS5) $d\,Sim\,Sim = d$

(MS6) $d\,Sim\,Rev\,Sim\,Rev = d$

(MS7) $d\,OProx \neq d\,Rev$

(MS8) $d\,OProx\,Rev\,OProx\,Rev = d$

As propriedades (MS2), (MS5), e (MS8) afirmam que as funções *Sim*, *Rev*, e *OProx* são permutações do conjunto de dardos; isto é, funções bijetoras de dardos para dardos (veja exercício 5.15). Mais exatamente, essas identidades afirmam que *Sim* e *Rev* são suas próprias inversas; e a inversa de *OProx* é *Rev OProx Rev*. Note que esta última devolve o *dardo anterior com mesma origem* que o argumento, na ordem cíclica dos dardos com mesma origem.

Ex. 5.15: Usando as propriedades (MS1–MS8), prove que

Identidades auxiliares

(MS9) $d\,OProx^k \neq d\,Rev$ para todo inteiro k

(MS10) $d\,Rev\,Sim = d\,Sim\,Rev$

(MS11) $d\,Sim\,Rev \neq d$

(MS12) $d\,Rev\,OProx\,Rev\,OProx = d$

Ex. 5.16: Prove a seguintes propriedade:

(MS14) $d\,Rev\,EProx\,Rev = d\,EProx^{-1}$

Ex. 5.17: Usando as funções *Sim*, *Rev*, e *OProx* apenas, diga como obter

(a) o próximo dardo com mesma face direita que d,

(b) o próximo dardo com mesmo vértice de destino que d

Diga também como obter as inversas destas funções (isto é, o dardo *anterior* a d com mesma face direita ou vértice de destino).

Ex. 5.18: Seja M o mapa simples que consiste dos vértices, arestas, e faces de um dodecaedro regular. Note que podemos dividir a superfície do dodecaedro em dois "hemisférios" iguais, cada um com 6 faces, cortando o mesmo ao longo de um circuito "equatorial" com 10 arestas. Usando as funções ambulatórias, escreva um algoritmo que enumera as arestas de tal circuito, a partir de um dardo conhecido do mesmo.

5.5.4 Álgebra de dardos

Pode-se provar que as funções *Sim*, *OProx*, e *Rev* definem completamente a topologia de um mapa simples. Isto é, se existe uma bijeção ϕ entre os dardos de dois mapas M e N, tal que $d\phi\, OProx = d\, OProx\, \phi$, $d\phi\, Sim = d\, Sim\, \phi$, e $d\phi\, Rev = d\, Rev\, \phi$, para todo dardo d, então M e N são topologicamente equivalentes.

Álgebra de dardos

Uma *álgebra de dardos* é uma estrutura matemática que consiste de um conjunto arbitrário D, com três funções arbitrárias *Sim*, *Rev*, e *OProx* de D para D que satisfazem as condições (MS1–MS8) acima. Pois bem, pode-se provar que, para toda álgebra de arestas, existe um mapa simples M cujos dardos e funções ambulatórias se comportam como os elementos e funções dessa álgebra.

5.5.5 Estrutura para mapas simples

Portanto, para representar fielmente a topologia de um mapa simples no computador, podemos usar uma estrutura de dados ligada, em que cada registro representa um dardo d, com três apontadores para os registros correspondentes aos dardos $d\, OProx$, $d\, Sim$, e $d\, Rev$.

Registro de aresta

Na verdade, podemos economizar muitos desses apontadores, se representarmos os quatro dardos originários da mesma aresta por um único registro dividido em quatro partes na forma de uma matriz 2×2. Veja a figura 5.11. As linhas desta matriz correspondem às duas orientações longitudinais da aresta, rotuladas arbitrariamente com 0 e 1; e as colunas correspondem às orientações transversais. Desta forma, um dardo d da aresta, com orientação longitudinal l e orientação transversal t, está representado pelo elemento $[g, v]$ desta matriz.

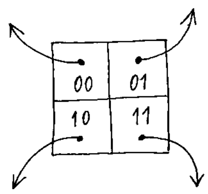


Figura 5.11: Representação compacta dos dardos de uma aresta.

Neste elemento armazenamos um apontador para o dardo $d\,OProx$, e possivelmente outras informações sobre dardo d, específicas da a aplicação. Note que não precisamos armazenar apontadores para os dardos $d\,Sim$ d $d\,Rev$, pois, como veremos na seção 5.5.6, eles podem ser calculados a partir do endereço de d.

Registros de vértices e faces

Os vértices e faces da estrutura geralmente são representados por registros separados. Nesse caso, em cada registro de aresta convém armazenar apontadores para os dois vértices em que a aresta incide (indexados pelas orientações longitudinais da aresta), e para as duas faces que incidem nela (indexadas pelas orientações transversais).

Conforme observamos anteriormente, as relações Sim, Rev, e $OProx$ entre os dardos determinam completamente a topologia do mapa; portanto, os registros de vértices e faces não precisam carregar informações topológicas. Tipicamente, estes registros contém dados geométricos (coordenadas do vértice, equação da face) ou dados específicos da aplicação.

5.5.6 Definição em Pascal

Representação de um dardo

Em linguagens de programação que permitem apontadores para lugares arbitrários da memória (como C, por exemplo), é possível representar um dardo pelo endereço do sub-registro correspondente. O endereço dos demais dardos da mesma aresta pode então ser obtido por aritmética de endereços.

Em linguagens como Pascal e Modula-2, que não permitem tais operações, os bits de orientação precisam ser especificados separadamente do o endereço do registro. Ou seja, para identificar um dardo precisamos fornecer uma tripla (p, l, t), onde p é um apontador para o registro da aresta, e l, t são os bits de orientação.

Na linguagem Pascal, por exemplo, a estrutura de dados poderia ser declarada da seguinte maneira:

```
type
  Bit = 0..1;
  Dardo = record
      p: ^Aresta;
      l: Bit;  { Orientacao longitudinal }
      t: Bit;  { Orientacao transversal }
    end;
  Aresta = record
      oprox: array [Bit, Bit] of Dardo;  { Indices [l,t] }
      org: array Bit of ^Vertice;        { Indice  [l] }
      esq: array Bit of ^Face;           { Indice  [l] }
    end;
  Vertice = record ... end;
  Face = record ... end;
```

Para obter os outros dardos da mesma aresta, basta manipular os bits de orientação l e t:

```
function Rev(d: Dardo): Dardo;
  begin
    Rev.p := d.p;
    Rev.l := d.l;
    Rev.t := 1 - d.t;
  end;

function Sim(d: Dardo): Dardo;
  begin
    Sim.p := d.p;
    Sim.l := 1 - d.l;
    Sim.t := 1 - d.t;
  end;
```

Por outro lado, o dardo $d\,OProx$ está armazenado explicitamente no sub-registro do dardo d:

```
function Oprox(d: Dardo): Dardo;
  begin
    Oprox := d.p^.oprox[d.l, d.t]
  end;
```

Os vértices e faces vizinhos a d são também extraídos do registro da aresta:

```
function Org(d: Dardo): ^Vertice;
  begin
    Org := d.p^.org[d.l]
  end;

function Dst(d: Dardo): ^Vertice;
  begin
    Dst := d.p^.org[1 - d.ll]
  end;

function Esq(d: Dardo): ^Face;
  begin
    Esq := d.p^.esq[d.t]
  end;

function Dir(d: Dardo): ^Face;
  begin
    Dir := d.p^.esq[1 - d.t]
  end;
```

Como vimos, as demais funções ambulatórias podem ser definidas a partir de *Sim*, *Rev*, e *OProx*:

```
function Oprev(d: Dardo): Dardo;   { Inversa de Oprox }
  begin
    Oprev := Rev(Oprox(Rev(d)))
  end;
```

```
function Eprox(d: Dardo): Dardo;
  begin
    Eprox := Oprev(Sim(d))
  end;

function Eprev(d: Dardo): Dardo;  { Inversa de Eprox }
  begin
    Eprev := Sim(Onext(d))
  end;
```

E assim por diante.

Ex. 5.19: Escreva um procedimento *InsereDiagonal*(r, s), que acrescenta ao mapa uma nova aresta t, ligando os vértices $r\,Org$ e $s\,Org$ através da face $r\,Esq$. Especifique detalhadamente o efeito desse procedimento.

Ex. 5.20: Escreva um algoritmo que constrói um mapa com 6 faces, 12 arestas, e 8 vértices, incidentes entre si como as faces, arestas e vértices de um cubo.

Ex. 5.21: Seja d um dardo de um mapa M, e ϕ é uma seqüência qualquer de operações *OProx*, *Sim*, *Rev*. Mostre que o efeito de *Rev* equivale a refletir o mapa M no espelho: isto é, se $d\phi = e$, então $d\,Rev\,\phi' = e\,Rev$, onde ϕ' é a seqüência obtida de ϕ trocando-se toda ocorrência de *OProx* por $OProx^{-1}$. Mais ainda, se $d\phi\,Esq = f$, então $d\,Rev\,\phi'\,Dir = f$

Ex. 5.22: Escreva um procedimento *Percorre*(r, VP, AP, FP), que percorre todos os elementos do mapa que contém a aresta r, chamando os procedimentos $VP(v)$ para cada vértice v, $AP(d)$ para cada dardo d, $FP(f)$ para cada face f, sem chamar duas vezes o mesmo procedimento com o mesmo argumento. (Para facilitar sua tarefa, suponha uma versão da linguagem Pascal na qual é permitido declarar variáveis do tipo **set of** T para qualquer tipo T, inclusive apontadores.)

Ex. 5.23: Escreva um algoritmo que, dado um mapa temático (A, v_A), devolve o mapa temático (B, v_B) mais grosseiro que representa o mesmo campo de valores.

5.5.7 Superfícies orientáveis

Superfície orientável

Neste livro vamos considerar apenas mapas em superfícies *orientáveis*. Informalmente, uma superfície é orientável se é possível definir em torno de cada ponto um *sentido positivo de rotação*, de maneira contínua, sobre toda a superfície.

A definição formal deste conceito é demasiado técnica para este livro. Entretanto, pode-se provar que uma superfície compacta S é orientável se e somente se ela é equivalente a alguma superfície $S' \subseteq \mathbb{R}^3$. Nesse caso, S' é a fronteira de algum sólido de extensão limitada contido em $\mathbb{R}^3$.

Portanto, a esfera $\mathbb{S}^2$ e o toro são orientáveis. Para comprovar esta afirmação, podemos definir o sentido positivo de rotação em torno de um ponto qualquer p como sendo o sentido anti-horário, visto por um observador no lado de fora da superfície e próximo ao ponto p.

Na maioria das aplicações práticas, a restrição a superfícies orientáveis é perfeitamente inócua. Em desenho industrial, por exemplo, mapas simples são usados principalmente para representar a superfície de peças mecânicas e outros objetos sólidos — que, como observamos acima, é sempre orientável. Em geoprocessamento, robótica, tipografia, e muitas outras aplicações, a única superfície de interesse é a esfera $\mathbb{S}^2$, ou, o que dá na mesma, o plano de dois lados $\mathbb{T}^2$. (Aliás, esta é mais uma vantagem de se trabalhar com $\mathbb{T}^2$, em vez do plano projetivo clássico $\mathbb{P}^2$.)

5.5.8 Representação de mapas orientáveis

Mapa orientável

A superfície de um mapa simples é orientável se e somente se o conjunto dos dardos do mesmo pode ser dividido em duas classes de igual tamanho, tais que cada classe é fechada sob *Sim* e *OProx*; sendo que *Rev* leva sempre de uma classe para a outra. Cada uma dessas classes de dardos corresponde a uma orientação da superfície, e pode ser interpretada como um lado da mesma. Numa superfície não-orientável, pelo contrário, é possível ir de algum dardo d para o dardo $d\,Rev$ por meio de uma seqüência de operações *Sim* e *OProx*.

Estrutura para mapas orientáveis

Se a superfície é orientável, a função *Rev* é em princípio supérflua; a topologia do mapa é inteiramente descrita pelas funções *Sim* e *OProx*, restritas a uma das duas classes de dardos. Portanto, numa aplicação em que todos os mapas são orientáveis, poderíamos simplificar um pouco a estrutura de dados da seção 5.5.6, omitindo o bit de orientação transversal na representação de dardos. Nesse caso, teríamos que substituir os quatro apontadores `oprox` por dois apontadores `oprox` e dois `oprev`. Ou seja, teríamos que declarar:

```
type
  Dardo = record
      p: ^Aresta;
      l: Bit;  { Orientacao longitudinal }
    end
  Aresta = record
      oprox: array Bit of Dardo;  { Indexado por [l] }
      oprev: array Bit of Dardo;  { Indexado por [l] }
      org: array Bit of ^Vertice; { Indexado por [l] }
      esq: array Bit of ^Face;    { Indexado por [l] }
    end;
```

Entretanto, esta simplificação economiza muito pouco espaço (4 bits por aresta), e tem algumas desvantagens. Veja os exercícios 5.24 e 5.25.

Ex. 5.24: Re-codifique as funções ambulatórias — *Sim*, *OProx*, $OProx^{-1}$, *EProx*, $EProx^{-1}$, *Org*, *Dst*, *Esq*, e *Dir* — para mapas orientáveis, supondo a estrutura de dados simplificada como acima.

Ex. 5.25: Suponha que queremos construir a imagem especular M' de um mapa M dado. Com a estrutura de dados descrita na seção 5.5.6, basta usar a função *Rev* (veja o exercício 5.21). Com a estrutura otimizada para superfícies orientáveis, descrita acima, é preciso construir uma nova estrutura para M'.

Escreva um algoritmo que constrói tal estrutura. Para facilitar sua tarefa, suponha que cada registro `Aresta` r de M contém um campo r`.num : integer`, entre 1 e o número de arestas de M, que identifica unicamente a aresta.

5.6 Superposição de mapas

Refinamento

Dizemos que um mapa A é um *refinamento* de um mapa B se todo elemento de A está inteiramente contido em algum elemento de B. Ou seja, se todo elemento de B é a união de um ou mais elementos de A.

Superposição de mapas

Definimos também a *superposição* $A \sqcap B$ de dois mapas A e B como sendo o mapa menos refinado que é um refinamento de A e de B.

Informalmente, $A \sqcap B$ é o resultado de repartir a superfície segundo os dois mapas ao mesmo tempo. Veja a figura 5.12.

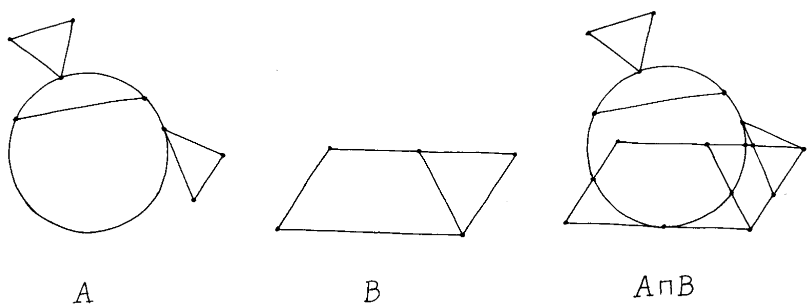


Figura 5.12: A superposição de dois mapas.

Elementos da superposição

É fácil ver que para cada elemento c de $A \sqcap B$ existe um único elemento de A e um único elemento de B que contém c. Na verdade, pode-se provar facilmente que todo elemento de $A \sqcap B$ é uma componente conexa de $a \cap b$, onde a é um elemento de A, e b é um elemento de B; e vice-versa.

5.6.1 Intersecção de polígonos

Intersecção de polígonos

Já mencionamos que a superposição de mapas oferece um caminho simples e eficiente para muitas operações geométricas. Um exemplo típico é o cálculo da intersecção de dois polígonos A e B, vistos como conjuntos de pontos de $\mathbb{T}^{\nvDash}$. Este problema é bastante complicado, mesmo quando A e B são polígonos simples com arestas retilíneas. Para começo de conversa, a intersecção pode consistir de mais de um polígono. Além disso, no caso de polígonos em $\mathbb{S}^{\nvDash}$ ou $\mathbb{T}^{\nvDash}$, a intersecção pode ter buracos. E, naturalmente, muitos polígonos que aparecem na prática não são simples.

Porém, o problema fica quase que trivial se tivermos à mão um algoritmo que constrói a superposição de dois mapas dados. Nesse caso, podemos considerar cada um dos polígonos como sendo um mapa temático, onde cada face f tem valor **true** para o interior do polígono, e **false** para o exterior. Veja a figura 5.13.

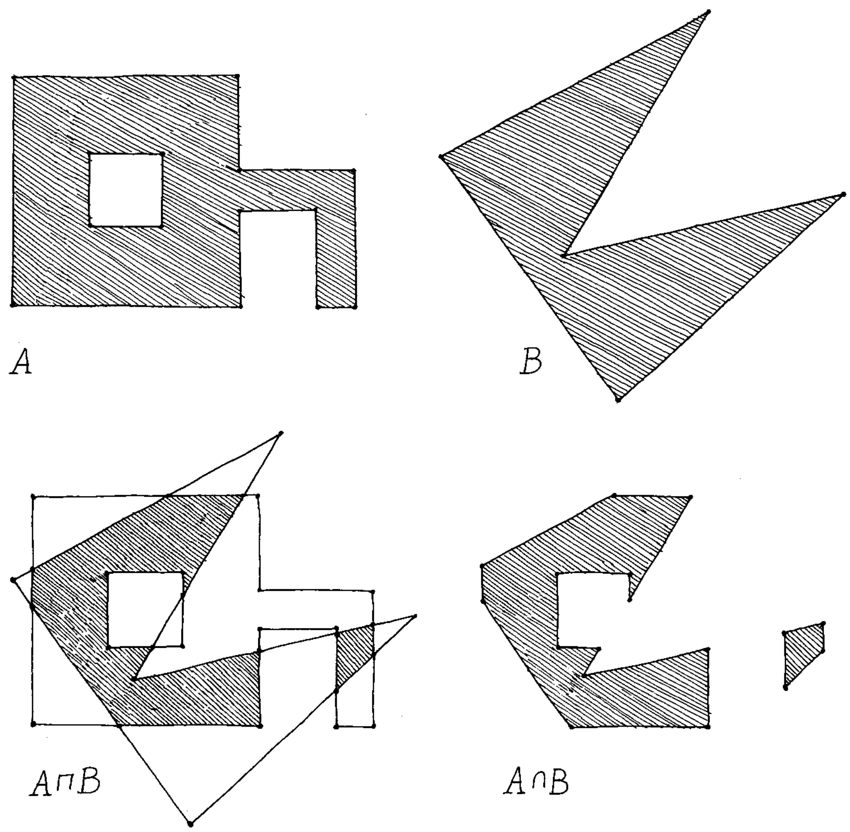


Figura 5.13: Intersecção de polígonos: preto é interior (**true**), branco é exterior (**false**).

Para obter a intersecção dos polígonos, construímos primeiro a superposição $C = A \sqcap B$ dos mapas. Note que cada face c de C é a intersecção de uma face a de A, e uma face b de B. Portanto, podemos atribuir a c um valor bem definido: **true** se tanto a quanto b forem **true**s nos respectivos mapas; e de **false** caso contrário. Ou seja, pintamos de preto as faces de C que eram pretas em ambos os mapas. Uma vez pintadas todas as faces do mapa C, basta eliminar do mesmo todas as arestas a que separam duas faces de mesma cor, e todos os vértices que não são ponta de nenhuma aresta.

Ex. 5.26: No algoritmo acima, os polígonos dados são considerados conjuntos abertos — isto é, as arestas e vértices de A não fazem parte do conjunto de pontos de A. Modifique o algoritmo de forma a (a) tratar arestas e vértices como parte dos polígonos; (b) permitir que o cliente especifique, para cada aresta ou vértice de entrada, se esse elemento pertence ou não ao conjunto.

A mesma técnica pode ser usada para calcular a união ou subtração de dois polígonos: a parte difícil do processo — calcular o mapa $C = A \sqcap B$ — é exatamente a mesma, muda apenas o critério para colorir as faces de C.

5.6.2 Operações pontuais

Operação pontual

Podemos generalizar este exemplo, da seguinte forma. Sejam f e g dois campos de valores com contra-domínios X e Y. Seja ϕ uma função que a cada par $x \in X$, $y \in Y$ associa um valor $x \, \phi \, y$ de um terceiro conjunto de valores Z. Estes dados definem um terceiro campo de valores $h = f \, \phi \, g$, com contra-domínio Z, que a cada ponto p de $\mathbb{T}^{\nvDash}$ associa o valor $h(p) = f(p) \, \phi \, g(p)$.

Note que o valor do campo h em cada ponto depende apenas dos valores de f e g nesse mesmo ponto, independentemente do que acontece no resto do plano. Dizemos portanto que h é o resultado da *operação pontual* ϕ aplicada aos dois campos f e g. Este conceito pode ser generalizado para funções ϕ com qualquer número de argumentos, de maneira óbvia.

Suponha agora que os campos f e g estão representados por mapas temáticos (A, u) e (B, v), e queremos construir um mapa temático (C, w) que representa h, Obviamente, basta tomar $C = A \sqcap B$, e associar a cada $c \in C$ o valor $w(c) = u(A(c)) \, \phi \, v(B(c))$.

A união, intersecção, e subtração de polígonos, discutidas acima, são exemplos óbvios de operações pontuais. Mais geralmente, toda operação booleana aplicada a conjuntos de pontos R, S equivale a uma operação pontual sobre suas funções características χ_R, χ_S.

Ex. 5.27: Seja $\mathbb{B} = \{\mathbf{true}, \mathbf{false}\}$. Dê as funções $x \phi y$, de $\mathbb{B} \times \mathbb{B}$ para $\mathbb{B}$, que definem operações pontuais sobre campos característicos equivalentes à união, intersecção, e subtração dos conjuntos correspondentes.

5.6.3 Algoritmo trivial

Algoritmo trivial de superposição

A definição de $A \sqcap B$ sugere um algoritmo trivial para sua construção: para cada elemento $a \in A$, e cada elemento $b \in B$, calcule a intersecção $a \cap b$, e enumere as componentes conexas da mesma. O conjunto de todas essas componentes é o mapa $A \sqcap B$.

Um possível problema com este algoritmo é que a intersecção de dois conjuntos conexos pode ter um número arbitrário — talvez infinito — de componentes conexas. Portanto, a topologia de $A \sqcap B$ pode ser arbitrariamente mais complicada que A ou B. Por exemplo, a intersecção dos arcos

$$\begin{aligned} a &= \{\, [1, t, t\sin(1/t)] \;:\; 0 < t < 1 \,\} \\ b &= \{\, [1, t, 0] \;:\; 0 < t < 1 \,\} \end{aligned}$$

é um conjunto infinito de pontos.

5.6.4 Superposição de mapas algébricos

Superposição de mapas algébricos

Entretanto, se os mapas A e B são mapas algébricos, é fácil ver que a intersecção de elementos $a \in A$ e $b \in B$ é um conjunto algébrico. Pode-se demonstrar que todo conjunto algébrico tem um número finito de componentes conexas, e que cada uma dessas componentes por si só é um conjunto algébrico. Portanto, concluímos que *a superposição de dois mapas simples algébricos é um mapa algébrico.* (A demonstração destes fatos está longe de ser trivial.)

Se considerarmos mapas algébricos formados por elementos "canônicos" — vértices, arcos, e discos — podemos dizer algo mais:

- A intersecção de um ponto isolado com qualquer conjunto é (obviamente) um ponto isolado;

- A intersecção de dois arcos algébricos um número finito de arcos algébricos, ou um número finito de pontos isolados;

- A intersecção de um arco algébrico e um disco algébrico é um número finito de arcos algébricos;

- A interseção de dois discos algébricos, cuja união não é a superfície toda, é um número finito de discos algébricos.

Ex. 5.28: Dê um exemplo de dois discos algébricos na esfera $\mathbb{S}^{\nvDash}$ cuja interseção não é um disco.

Destes resultados, podemos concluir que, para dois mapas simples algébricos A e B, todo vértice de $A \sqcap B$ é um vértice de A ou de B, ou um ponto isolado da intersecção de duas arestas. Podemos concluir também que toda aresta de $A \sqcap B$ é um pedaço de uma aresta de A ou de B.

5.6.5 Algoritmo menos trivial

As observações da seção anterior sugerem o seguinte algoritmo, um pouco menos trivial:

Algoritmo 5.1 (*SupTriv*) *Dados dois mapas simples algébricos A e B, na mesma superfície, devolve o mapa $C = A \sqcap B$.*

1. Faça $V \leftarrow\ VA \cup VB$.
2. Para cada par de arestas $a \in EA$, $b \in EB$, tal que $a \cap b$ consiste de pontos discretos, faça $V \leftarrow V \cup (a \cap b)$.
3. Faça $E \leftarrow \{\}$.
4. Para cada aresta a de $EA \cup EB$, determine todos os pontos de V que estão em A. Para cada trecho c de a entre dois esses pontos, crie um novo registro de aresta e acrescente-o a E.

Construindo a estrutura de $A \sqcap B$

Não é difícil completar este algoritmo de modo a registrar, para todo vértice e aresta c de C, os elementos $A(c)$ e $B(c)$ dos mapas originais que contém c. Com um pouco mais de trabalho. é possível também determinar o sucessor $c\,\mathit{OProx}$ para todo dardo c de C. Com esta informação, é possível determinar as faces de C; e registrar, para cada face f, as faces originais $A(f)$ e $B(f)$ que contêm f.

Em suma, se A e B são representados por estruturas de arestas como a descrita na seção 5.5.6; o algoritmo acima permite construir uma representação análoga para o mapa C. Isto pressupõe que C é um mapa simples, e em particular que toda face de C seja um disco. No caso de uma superfície esférica, isto é verdade se A e B forem mapas simples, e toda face dos mesmos estiver contida num hemisfério.

Custo da superposição

Qual é o custo deste algoritmo? Se A e B tem n arestas, no total, então no pior caso temos que examinar $\Theta(n^2)$ pares de arestas. Se o custo de cada par é $O(1)$ (isto é, independente de n), então o custo total do algoritmo também será $\Theta(n^2)$.

Cota inferior

É fácil dar exemplos de dois mapas poligonais simples com n elementos cuja superposição tem $\Theta(n^2)$ elementos. Portanto, não existe nenhum algoritmo que seja assintoticamente mais rápido que o algoritmo 5.1 para todos os casos.

Entretanto, na prática a complexidade do mapa $A \sqcap B$ raramente atinge o limite teórico. Numa grande percentagem dos casos, o número de elementos de $A \sqcap B$ não é muito maior que número de elementos de A e B — ou seja, cada aresta de A cruza em média umas poucas arestas de B.

Portanto, para fins práticos é desejável um algoritmo de superposição cujo custo seja mais ou menos proporcional à complexidade do mapa final. Nas próximas seções desenvolveremos tal algoritmo.

5.7 O paradigma da varredura

Como vimos na seção anterior, o passo mais crítico no cálculo de $A \sqcap B$ é determinar todos cruzamentos entre arestas de A e de B.

Determinação eficiente dos cruzamentos

A técnica de varredura do plano permite calcular essas intersecções a um custo geralmente menor que o do algoritmo 5.1. Mais precisamente, se n é o número de arestas de A e B, e m o número de intersecções, a técnica de varredura permite determinar todos os cruzamentos em tempo $O((m+n)\log(m+n))$ — ou seja, a um custo médio de $O(\log(m+n))$ por arestas.

Uma vez conhecidos todos esses cruzamentos, não é difícil determinar as demais propriedades do mapa $A \sqcap B$, e construir a estrutura de dados para o mesmo, a um custo adicional de apenas $O(\log(m + n))$.

5.7.1 O algoritmo de Bentley e Ottmann

Problema simplificado

Para explicar o paradigma de varredura, vamos começar por um problema bem simples: dado um conjunto S de segmentos de $\mathbb{R}^2$, determinar todos os pares de segmentos que se cruzam. A solução clássica para este problema é o algoritmo Bentley e Ottmann [BO79], que descrevemos a seguir.

Reta de varredura

Imagine uma reta vertical varrendo o plano $\mathbb{R}^2$ da esquerda para a direita. Em cada posição, a reta r encontra um certo subconjunto de segmentos, que chamaremos de *segmentos ativos*. Seja $\mathcal{A}(r)$ a lista desses segmentos, ordenados pela posição y em que elas cruzam r.

Eventos

Vamos supor não há dois vértices ou cruzamentos com mesma abscissa, ou dois segmentos colineares. Nesse caso, a ordem dos segmentos em $\mathcal{A}(r)$ só muda em dois tipos de *eventos*: quanto a reta passa pelo extremo de um segmento, ou pelo cruzamento de dois segmentos. No primeiro caso, um segmento sai da lista, ou entra nela. No segundo caso, os dois segmentos trocam de lugar na lista. Além disso, apenas um destes eventos pode acontecer de cada vez.

Uma vez que os número de eventos é finito, é possível simular esta varredura num computador. No início de cada iteração, o conteúdo e ordem da lista $\mathcal{A}$ corresponde à reta r posicionada entre dois eventos sucessivos. O algoritmo determina o próximo evento, e simula a passagem da reta por sobre o mesmo. Esta simulação implica em retirar, acrescentar, e permutar segmentos da lista $\mathcal{A}$ de modo a refletir a situação com a reta posicionada logo após o evento.

5.7.2 A agenda de eventos futuros

Agenda de eventos

A dificuldade toda está em determinar o próximo evento, de maneira eficiente. Para isso, usaremos uma lista $\mathcal{E}$ — a *agenda* — que contém todos os eventos futuros conhecidos, ordenados por abscissa crescente. Assim, o próximo evento será sempre o primeiro item da agenda.

No início do algoritmo, a agenda contém apenas os eventos do primeiro tipo — os extremos de todos os segmentos. Os eventos de segundo tipo — cruzamentos — serão descobertos no decorrer da varredura, e inseridos na agenda, na posição adequada.

Previsão de eventos

Obviamente, para que esta idéia funcione, precisamos descobrir cada cruzamento *antes* da reta r passar pelo mesmo. A esse respeito, observe que, logo antes da reta r passar por um cruzamento, os dois segmentos estarão obrigatoriamente em posições adjacentes na lista. Portanto, para descobrir todos os cruzamentos, basta examinar todos os pares de segmentos *consecutivos* da lista $\mathcal{A}$.

Mais ainda, uma vez que o processamento de um evento só afeta poucos elementos da lista, basta examinar apenas os novos pares de elementos consecutivos que resultaram desse processamento.

Processamento de um evento

Por exemplo, no caso da figura 5.14(a), quando a reta passa pelo ponto v, o segmento s sai da lista, e os segmentos a e c passam a ser vizinhos. Portanto, nessa hora devemos verificar se há algum cruzamento entre as arestas a e c com abscissa maior que a de v.

No caso da figura 5.14(b), quando a reta passa pelo extremo v, o segmento s é acrescentado à lista, entre os segmentos a e c. Nesse momento, devemos verificar se s cruza a ou c.

Finalmente, na figura 5.14(c), quando a reta passa pelo cruzamento v de r com s, estes dois segmentos trocam de lugar na lista. Devemos então verificar se há cruzamentos entre s e a, e/ou entre r e b, com abscissa maior que a de v; sendo que a e b são os segmentos ativos imediatamente acima e abaixo do cruzamento v.

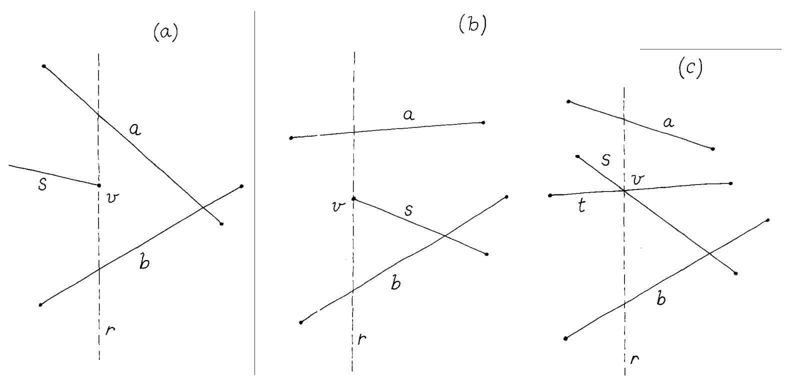


Figura 5.14: Os três tipos de eventos: (a) extremo esquerdo, (b) extremo direito, (c) cruzamento de segmentos.

Variáveis do algoritmo

Abscissa de um evento

Na descrição formal do algoritmo, abaixo, cada evento da agenda $\mathcal{E}$ é uma ênupla da forma $(\mathbf{esq}, s)$, $(\mathbf{dir}, s)$, ou $e = (\mathbf{crz}, r, s)$, onde r e s são segmentos. A *abscissa* do evento é, nos dois primeiros casos, a do extremo correspondente de s; e, no terceiro, a do ponto onde r e s se cruzam.

Algoritmo 5.2 (Bentley-Ottmann) *Dado um conjunto S de segmentos com extremos em $\mathbb{R}^{\not\models}$, devolve uma lista $\mathcal{C}$ de pares de segmentos que se cruzam, em ordem de abscissa crescente.*

1. Inicialize $\mathcal{E} \leftarrow \{\}$, $\mathcal{A} \leftarrow ()$, e $\mathcal{C} \leftarrow ()$.
2. Para todo segmento $s \in S$, acrescente $(\mathbf{esq}, s)$ à agenda $\mathcal{E}$.
3. enquanto $\mathcal{E} \neq \{\}$ faça
   - 3.1. Retire de $\mathcal{E}$ o evento e de menor abscissa.
   - 3.2. Se e tem a forma $(\mathbf{esq}, s)$, então:
     - 3.2.1. Seja v o extremo esquerdo de s. Localize os segmentos a e b de $\mathcal{A}$ imediatamente acima e abaixo do ponto v.
     - 3.2.2. Insira s em $\mathcal{A}$ entre a e b.
     - 3.2.3. Acrescente o evento $(\mathbf{dir}, s)$ à agenda $\mathcal{E}$.
     - 3.2.4. Se s cruza a à direita de v, acrescente o evento $(\mathbf{crz}, s, a)$ à agenda $\mathcal{E}$. Idem para b e s.
   - 3.3. Senão, se e tem a forma $(\mathbf{dir}, s)$, então:
     - 3.3.1. O segmento s deve estar em $\mathcal{A}$. Sejam a e b os segmentos de $\mathcal{A}$ imediatamente acima e abaixo de s.
     - 3.3.2. Remova s de $\mathcal{A}$.
     - 3.3.3. Se a cruza b à direita de v, acrescente o evento $(\mathbf{crz}, a, b)$ à agenda $\mathcal{E}$ e à lista $\mathcal{C}$.
   - 3.4. Senão, se e tem a forma $(\mathbf{crz}, r, s)$, então:
     - 3.4.1. Os segmentos r e s devem estar em posições consecutivas de $\mathcal{A}$, com r abaixo de s. Sejam a e b os segmentos de $\mathcal{A}$ imediatamente acima de s e abaixo de r. Seja v o ponto de cruzamento.
     - 3.4.2. Acrescente o par (r, s) ao fim da lista $\mathcal{C}$.
     - 3.4.3. Troque as posições de r e s em $\mathcal{A}$.
     - 3.4.4. Se r cruza a à direita de v, acrescente o evento $(\mathbf{crz}, r, a)$ à agenda $\mathcal{E}$. Idem para b e s.
4. devolva a lista $\mathcal{C}$

Ex. 5.29: No algoritmo 5.2, mostre que o mesmo cruzamento $(\mathbf{crz}, a, b)$ pode ser inserido mais de uma vez na agenda $\mathcal{E}$. (Dê um conjunto de segmentos S para o qual isto acontece).

5.7.3 Custo do algoritmo

O tempo gasto pelo algoritmo de Bentley e Ottmann é menor ou igual ao número de eventos processados, vezes o tempo máximo gasto para processar cada evento.

Custo do algoritmo

Se o conjunto de entrada S tem n segmentos, e k pares desses se cruzam, o número de eventos processados pelo algoritmo é exatamente $2n+k$. Como já observamos, o número de cruzamentos k pode variar entre 0 e $\binom{n}{2} = n(n-1)/2$.

O tempo necessário para processar um evento depende da maneira como as listas $\mathcal{A}$ e $\mathcal{E}$ são armazenadas. Se usássemos uma representação trivial para a agenda — um vetor, ou uma lista ligada — o custo para inserir um novo evento ou para extrair o evento de menor abscissa seria proporcional ao número de eventos na mesma. Nessas condição, não é difícil de mostrar que, no pior caso, o custo total do algoritmo seria $\Omega(n^3)$ — maior que o do algoritmo ingênuo, que testa todos os pares de segmentos. Veja o exercício 5.33

Representação eficiente da agenda

Felizmente, se armazenarmos a agenda $\mathcal{E}$ a forma de uma estrutura um pouco mais sofisticada — uma *fila de prioridade* — é possível efetuar cada uma dessas duas operações em tempo $O(\log m)$, onde m é o número de eventos na agenda. Uma vez que $m = O(n^2)$, e $\log(n^2) = O(\log n)$, concluímos que o custo de cada uma destas operações é no máximo $O(\log n)$.

Representação eficiente da lista ativa

Analogamente, se usássemos uma representação trivial para a lista $\mathcal{A}$, teríamos no pior caso que gastar tempo $\Omega(n)$ para inserir um novo segmento na mesma, ou para localizar os segmentos adjacentes ao mesmo (passo 3.2.1). Podemos resolver este problema usando uma estrutura de dados mais sofisticada, como por exemplo as *árvores auto-balanceadas* de Sleator e Tarjan [ST83].

Com estas modificações, o tempo gasto pelo algoritmo de Bentley e Ottmann é no máximo $O((n+k)\log n)$.

Pior caso

Note que este limite ainda é assintoticamente maior que o do algoritmo 5.1 quando há muitas intersecções. Entretanto, em muitas aplicações práticas, o número esperado de cruzamentos k é $O(n)$ ou menos. Nessas condições, o algoritmo de Bentley-Ottmann roda em tempo $O(n \log n)$, enquanto que o algoritmo trivial ainda exige tempo $O(n^2)$.

Existem algoritmos mais recentes que resolvem este mesmo problema em tempo $O(n \log n + k)$ — isto é, $O(\log n)$ por segmento, mais $O(1)$ por cruzamento. Entretanto, estes algoritmos são relativamente complicados, e não se sabe se eles são realmente mais eficientes que o de Bentley-Ottmann em problemas práticos.

Ex. 5.30: (a) Se o conjunto S tem n segmentos, qual o comprimento máximo da lista $\mathcal{A}$, no decorrer do algoritmo? (b) Mostre que para todo n existe um conjunto de n segmentos para o qual este máximo é atingido.

Ex. 5.31: (a) Prove que o número de eventos na agenda $\mathcal{E}$, em qualquer momento, é sempre menor ou igual a $n + k$. (b) Será que para todo n e todo $k \leq n(n-1)/2$ existe um conjunto s de segmentos para o qual esse máximo é atingido?

Ex. 5.32: Seja α_i o número de itens na lista $\mathcal{A}$, no momento em que o i-ésimo item é inserido na mesma. Mostre que, para todo n, existe um conjunto S de n segmentos tal que $\alpha_i = \Omega(n)$ para $\Omega(n)$ valores diferentes de i.

Ex. 5.33: Seja ε_j o tamanho da agenda $\mathcal{E}$, no momento em que o j-ésimo evento é inserido na mesma. Mostre que, para todo n, existe um conjunto S de n segmentos tal que $\varepsilon_j = \Omega(n)$ para pelo menos $\Omega(n^2)$ valores diferentes de j.

5.8 Varredura da esfera

A idéia central do algoritmo de Bentley-Ottmann é transformar um problema bidimensional estático num problema unidimensional dinâmico, em que a cada momento nos preocupamos apenas com o que acontece ao longo de uma linha que varre o plano todo.

Podemos usar essa mesma técnica para calcular eficientemente as intersecções entre as arestas de mapas em geral. Para isso, entretanto, precisamos modificar vários detalhes do algoritmo.

Porquê na esfera?

Em primeiro lugar, muitos mapas de interesse prático são mapas sobre a esfera $\mathbb{S}^{\nvDash}$ (ou no plano projetivo orientado $\mathbb{T}^{\nvDash}$). Mesmo mapas restritos ao plano cartesiano $\mathbb{R}^{\nvDash}$ freqüentemente tem arestas infinitas ou semi-infinitas; a maneira mais simples de tratar essas arestas é supor que seus extremos são pontos no infinito.

Meridianos e pólos de varredura

Varrer a esfera $\mathbb{S}^{\nvDash}$ é mais complicado do que varrer o plano $\mathbb{R}^{\nvDash}$. Em primeiro lugar, em vez de uma reta de varredura, temos que usar um *meridiano*: um arco de círculo máximo com extremos antipodais, girando em torno do eixo que passa pelos mesmos. Veja a figura 5.15. Vamos chamar os extremos do meridiano de *pólo norte* ($\mathcal{N}$) e *pólo sul* ($\mathcal{S} = \neg\mathcal{N}$); sendo que o meridiano gira em sentido positivo em torno de $\mathcal{N}$, e negativo em torno de $\mathcal{S}$.

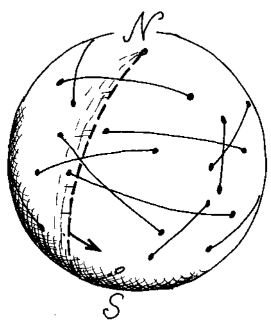


Figura 5.15: Varrendo a esfera com um meridiano.

Reta suporte do meridiano

Cada meridiano m pode ser identificado com sua *reta suporte* $\mathcal{N} \vee p = p \vee \mathcal{S}$, onde p é qualquer ponto do meridiano. Note que, segundo esta definição, o trajeto de $\mathcal{N}$ para $\mathcal{S}$ ao longo do meridiano gira no sentido positivo em torno do lado positivo de sua reta suporte.

Meridiano inicial e longitude

Outra dificuldade a enfrentar na varredura da esfera é que a ordem em que os eventos são encontrados pelo meridiano de varredura não é simplesmente a ordem de abscissa crescente. Na verdade, os meridianos com mesmo pólo são ordenados circularmente, e não linearmente. Portanto, para ordenar os eventos, precisamos escolher um *meridiano inicial* m^*, onde a varredura começa. Podemos então classificar cada evento pela *longitude* do meridiano que passa por ele, medida a partir de m^* no sentido de rotação positivo em torno do pólo norte.

Para fins de ilustração, vamos supor que $\mathcal{N}$ é o ponto $[0, 0, 1]$; e que m^* é o meridiano que contém o ponto $[0, -1, 0]$, cuja reta suporte é

$\Omega = \langle 0,0,1 \rangle$. Entretanto, vale a pena frisar que $\mathcal{N}$ pode ser qualquer ponto da esfera, e m^* pode ser qualquer arco com extremos $\mathcal{N}$ e $\neg\mathcal{N}$.

Ex. 5.34: (a) Descreva a aparência de um meridiano genérico no modelo plano de $\mathbb{T}^{\nvDash}$, supondo que $\mathcal{N} = [0,0,1]$. (b) Idem, para $\mathcal{N} = [0,1,0]$. (c) Idem, para $\mathcal{N} = [1,0,0]$.

Leste, oeste, norte, sul

Se a longitude de um ponto p for menor que a de outro ponto q, diremos que *p está a oeste de q*, e *q está a leste de q*. Se p e q tiverem a mesma longitude, e um percurso de $\mathcal{N}$ para $\mathcal{S}$ ao longo do meridiano encontrar p antes de q, diremos que *p está ao norte de q*, e *q está ao sul de p*. (Não precisaremos definir a ordem norte-sul para pontos com longitudes diferentes.) Note que estes conceitos não estão definidos quando p ou q coincidem com os pólos da varredura.

Ex. 5.35: Dê um algoritmo para decidir se um ponto p está a oeste de um ponto q. (Dica: compare a ordenação por longitude com a angular usada na seção 4.3.4.)

Ex. 5.36: Dê um algoritmo para decidir se p está ao norte de q, sabendo-se que ambos estão no mesmo meridiano.

Começando pelo meio

Uma terceira dificuldade da varredura esférica é que, existem mapas tais que qualquer meridiano, com qualquer pólo, cruza uma ou mais arestas. Ou seja, não existe em geral uma posição “limpa” para o meridiano de varredura. Portanto, qualquer algoritmo de varredura precisa começar “pelo meio”, com uma lista de segmentos ativos não-vazia. Em geral, o algoritmo precisa deixar em aberto alguns detalhes do resultado, que dependem das partes não varridas desses segmentos; e, ao fim da varredura, precisa voltar atrás e completar esses detalhes.

Corte e costura

Uma maneira relativamente clara de descrever (e implementar) estas modificações é imaginar a esfera cortada ao longo do meridiano inicial m^*. Topologicamente, esse corte transforma a esfera $\mathbb{S}^{\nvDash}$ no plano cartesiano $\mathbb{R}^{\nvDash}$, e a varredura de $\mathbb{S}^{\nvDash}$ por um meridiano na varredura de $\mathbb{R}^{\nvDash}$ por uma reta — tal como no algoritmo de Bentley e Ottmann. O problema de “começar no meio” passa a ser então que cada aresta que cruza m^* fica dividida em pelo menos duas arestas; e o mesmo acontece com as faces. Para corrigir esse problema, quando a varredura termina precisamos "costurar" de volta as duas margens do mapa, juntando os pedaços de cada aresta ou face que cruza o meridiano de referência. A figura 5.16 ilustra este processo.

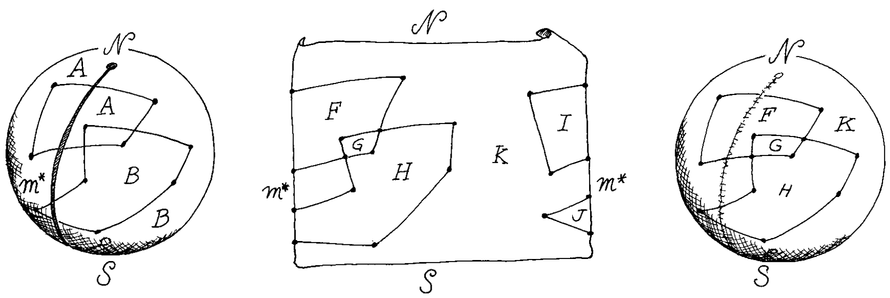


Figura 5.16: Corte e costura de mapas na esfera.

5.8.1 Arestas curvas

Arestas curvas

Além disso, precisamos levar em conta que muitos mapas de interesse prático tem arestas curvas. Na varredura de tais mapas, precisamos lidar com o fato que um meridiano de varredura pode encontrar a mesma arestas várias vezes, e que a lista de arestas ativas pode mudar não apenas em vértices e cruzamentos, mas também quando o meridiano fica tangente a uma aresta. Veja a figura 5.17.

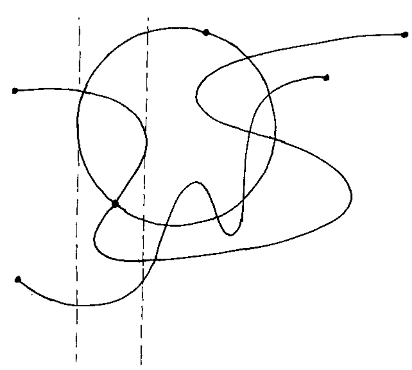


Figura 5.17: Varrendo um mapa com arestas curvas.

Arestas monotônicas

Este problema não ocorre se toda aresta for *monotônica em longitude*: isto é, se cada meridiano cruzar cada aresta em no máximo um ponto. Em geral, podemos resolver este problema com um pré-processamento que divide toda aresta em trechos monotônicos, introduzindo novos vértices nos pontos da mesma em que a longitude é localmente mínima ou máxima.

Para que esta solução seja viável, é preciso que haja apenas um número finito de pontos estacionários, e que eles sejam determináveis algoritmicamente. Estas duas condições são satisfeitas no caso de curvas algébricas implícitas ou paramétricas, e em particular no caso de círculos, seções cônicas em geral, e curvas de Bèzier.

Cruzamentos múltiplos

Outro detalhe a ser levado em conta é que duas arestas curvas podem se cruzar mais de uma vez. Felizmente, pode-se provar que a intersecção de dois arcos de curvas algébricas consiste um número finito de arcos e pontos isolados, que podem ser determinados algoritmicamente.

Ex. 5.37: Dê um algoritmo para decompor um círculo de centro (finito) $p = [w, x, y]$ e raio r em um ou mais arcos monotônicos em longitude, em relação a um pólo $\mathcal{N}$ dado, separados por vértices isolados. Suponha que círculo não passa por $\mathcal{N}$.

5.8.2 Casos degenerados

Casos degenerados

Além das dificuldades causadas pela topologia esférica e pelas arestas curvas, temos que nos preocupar também com certas "situações degeneradas", que foram excluídas por hipótese na nossa descrição do algoritmo de Bentley e Ottmann, mas que não podem ser ignoradas na superposição de mapas.

Múltiplas arestas por vértice

Por exemplo, quase todo mapa tem duas ou mais arestas incidindo no mesmo vértice. Essa é uma característica essencial do mapa, que determina sua topologia; portanto, não é viável exigir que a cada extremo de cada aresta tenha uma latitude diferente. Isso significa que, quando o meridiano de varredura passa sobre um vértice, temos que processar num único evento todas as arestas que incidem nesse vértice.

Vértices sobre arestas

Pode também acontecer que, na sobreposição de mapas A e B, um vértice do mapa A esteja no meio de uma aresta de B, ou vice-versa. Para não complicar o algoritmo principal, vamos supor que os mapas foram previamente refinados de modo a eliminar estas situações.

Mais precisamente, vamos supor que toda aresta de A que continha um vértice v de B foi dividida em duas arestas, separadas por um novo vértice v' com mesmas coordenadas que v; e analogamente para o mapa B.

Com este refinamento inicial, todo vértice da superposição é a intersecção de um vértice e uma face, ou de dois vértices, ou de duas arestas.

Arestas sobrepostas

Pode também acontecer que a intersecção de duas arestas (uma de A e outra de B) contenha não apenas pontos discretos, mas também arcos inteiros. Se as curvas são algébricas, e os mapas foram refinados como descrito acima, prova-se que neste caso as duas arestas são idênticas, e portanto iguais à sua intersecção. Este fato simplifica bastante o tratamento destes casos.

5.8.3 Escolha dos pólos

Escolha dos pólos

Note que as situações degeneradas descritas até aqui são intrínsecas ao problema: elas dizem respeito apenas aos dados e ao resultado desejado. Além dessas, há várias outras situações degeneradas que envolvem os pólos de varredura e o meridiano inicial, e portanto são internas ao algoritmo. Por exemplo, podemos ter eventos (vértices ou cruzamentos) no meridiano inicial, ou múltiplos eventos com a mesma longitude, ou arestas passando pelos pólos, etc.

perturbação virtual

Felizmente, estas situações podem ser evitadas por meio de "perturbações virtuais" na posição do pólo e do meridiano inicial. A idéia é supor que o pólo norte não está exatamente em $[0, 0, 1]$, mas sim num ponto $[\varepsilon^2, \varepsilon, 1]$, onde ε é um *infinitésimo*: um "número" que, por definição, é maior que zero, mas menor que qualquer real positivo.

Uma conseqüência imediata desta perturbação infinitésima é que *os pólos $\mathcal{N}$ e $\mathcal{S}$ não coincidem com nenhum vértice do mapa* — na verdade, não coincidem com nenhum ponto "ordinário" de $\mathbb{T}^{\not\models}$!

Longitude perturbada

Mais ainda, a perturbação garante que *cada ponto de $\mathbb{T}^{\not\models}$ está num meridiano distinto*. Para verificar esta afirmação, vamos tomar dois pontos quaisquer $p_1 = [w_1, x_1, y_1]$ e $p_2 = [w_2, x_2, y_2]$ de $\mathbb{T}^{\not\models}$, e calcular a orientação do triângulo $p_1 p_2 \mathcal{N}$:

$$
\begin{aligned}
&\Delta(\mathcal{N}, p_1, p_2) && (5.3)\\
&= \operatorname{sgn}\begin{vmatrix} \varepsilon^2 & \varepsilon & 1 \\ w_1 & x_1 & y_1 \\ w_2 & x_2 & y_2 \end{vmatrix} \\
&= \operatorname{sgn}\left(\begin{vmatrix} 0 & 0 & 1 \\ w_1 & x_1 & y_1 \\ w_2 & x_2 & y_2 \end{vmatrix} + \varepsilon\begin{vmatrix} 0 & 1 & 0 \\ w_1 & x_1 & y_1 \\ w_2 & x_2 & y_2 \end{vmatrix} + \varepsilon^2\begin{vmatrix} 1 & 0 & 0 \\ w_1 & x_1 & y_1 \\ w_2 & x_2 & y_2 \end{vmatrix}\right) && (5.4)\\
&= \operatorname{sgn}\left(\begin{vmatrix} w_1 & x_1 \\ w_2 & x_2 \end{vmatrix} - \varepsilon\begin{vmatrix} w_1 & y_1 \\ w_2 & y_2 \end{vmatrix} + \varepsilon^2\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}\right) && (5.5)
\end{aligned}
$$

Observe que o argumento de sgn na fórmula (5.5) é um polinômio $a_0 + a_1\varepsilon + a_2\varepsilon^2$ sobre o infinitésimo ε, cujos coeficientes são números reais ordinários. Se o primeiro coeficiente a_0 for diferente se zero, seu valor absoluto é por definição maior que qualquer múltiplo de ε ou ε^2; e portanto o sinal da fórmula toda é o sinal de a_0. Pelo mesmo argumento, se a_0 for zero, mas a_1 for diferente de zero, o sinal da fórmula será o sinal de a_1. Se ambos forem zero, vale o sinal de a_2. Temos portanto o seguinte algoritmo:

Algoritmo 5.3 **($\Delta_{\mathcal{N}}(p_1, p_2)$)** *Dados dois pontos* $p_1 = [w_1, x_1, y_1]$ *e* $p_2 = [w_2, x_2, y_2]$*, devolve* $\Delta(\mathcal{N}, p_1, p_2)$*, onde* $\mathcal{N} = [\varepsilon^2, \varepsilon, 1]$.

1. $s \leftarrow \begin{vmatrix} w_1 & x_1 \\ w_2 & x_2 \end{vmatrix}$; se $s \neq 0$ então devolva $\operatorname{sgn} s$;
2. $s \leftarrow \begin{vmatrix} w_1 & y_1 \\ w_2 & y_2 \end{vmatrix}$; se $s \neq 0$ então devolva $-\operatorname{sgn} s$;
3. $s \leftarrow \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}$; devolva $\operatorname{sgn} s$.

Note que $\Delta_{\mathcal{N}}(p_1, p_2)$ só é zero se os três determinantes 2×2 acima forem nulos. Nesse caso, pode-se concluir que os dois vetores (w_1, x_1, y_1) e (w_2, x_2, y_2) são múltiplos um do outro; ou seja, os pontos p_1 e p_2 são coincidentes (iguais ou antipodais). Como pontos antipodais sempre estão em meridianos opostos, concluímos que p_1 e p_2 estão no mesmo meridiano se e somente se $p_1 = p_2$.

Perturbação do meridiano inicial

A técnica de perturbação infinitésima também pode ser usada para garantir que o meridiano de referência m^* não passa por nenhum vértice ou cruzamento dos dois mapas. Para tanto, basta tomar m^* na reta $\langle 1+\varepsilon^2, \varepsilon^3, -\varepsilon^2\rangle$. Pode-se verificar que esta "reta perturbada" passa por $\mathcal{N}$ e $\mathcal{S}$, mas não passa por nenhum ponto "ordinário" de $\mathbb{T}^{\not\models}$.

Os exercícios a seguir supõem estas perturbações infinitésimas; isto é, $\mathcal{N} = [\varepsilon^2, \varepsilon, 1]$ e $m^* = \langle 1 + \varepsilon^2, \varepsilon^3, -\varepsilon^2\rangle$.

Ex. 5.38: Dê um algoritmo que, dados dois pontos p e q de $\mathbb{T}^{\not\models}$, devolve $+1$ se p está a leste de q, -1 se p está a oeste de q, e 0 se $p = q$.

Ex. 5.39: Dê um algoritmo que, dados dois pontos p e q de $\mathbb{T}^{\not\models}$, distintos e não antipodais, decide se p é o extremo oeste ou o extremo leste do segmento pq.

Ex. 5.40: Dê um algoritmo que, dados um ponto p de $\mathbb{T}^{\not\models}$, e um segmento qr que cruza o meridiano de p, determina se p está ao sul ou ao norte de qr.

Ex. 5.41: Dê um algoritmo, que, dados dois segmentos pq e pr, dos quais p é o extremo oeste, devolve $+1$ se pq está mais ao norte que pr, -1 se está mais ao sul, e 0 se os dois segmentos são colineares.

Ex. 5.42: Dê um algoritmo para decidir se um segmento pq cruza o meridiano inicial m^*.

Ex. 5.43: Dê um algoritmo, que, dados dois segmentos disjuntos pq e rs que cruzam o meridiano inicial m^*, determina qual deles cruza o segmento mais ao norte.

5.9 Apêndice: Conceitos básicos de topologia

Esta seção é uma recapitulação sucinta das definições mais de topologia de conjuntos usadas neste capítulo. Para maiores detalhes, recomendamos consultar um livros texto da área, como por exemplo [Lim76].

5.9.1 Espaço topológico; abertos e fechados

Espaço e conjunto aberto

Um *espaço topológico* é uma tripla $(X, \mathcal{A})$, onde X é um conjunto qualquer (os *pontos*), e $\mathcal{A}$ é uma coleção de subconjuntos de X (os *abertos*), satisfazendo os seguintes axiomas:

1. o conjunto X e o conjunto vazio são abertos;
2. a união de qualquer coleção de subconjuntos abertos é um aberto;
3. a intersecção de uma coleção *finita* de subconjuntos abertos é um aberto.

Topologia

Nessas condições, dizemos que a coleção $\mathcal{A}$ *determina uma estrutura topológica* sobre o conjunto X, ou que *é uma topologia para* X.

No resto deste apêndice, vamos supor que $(X, \mathcal{A})$ é um espaço topológico.

5.9.2 Subconjuntos fechados

Conjunto fechado

Por definição, um subconjunto Z de X é *fechado* se seu complemento $X \setminus Z$ é aberto. Portanto, X e $\{\}$ são fechados; a intersecção de qualquer família de subconjuntos fechados é fechada; e a união *finita* de subconjuntos fechados é fechada.

Em geral, num espaço $(X, \mathcal{A})$ existem subconjuntos de pontos que não são nem abertos nem fechados.

5.9.3 Sub-espaços

Sub-espaço

Dizemos que um espaço $(Z, \mathcal{B})$ é um *sub-espaço* de $(X, \mathcal{A})$ se e somente se $Z \subseteq X$, e

$$\mathcal{B} = \{\, A \cap Z \;:\; A \in \mathcal{A} \,\}$$

Neste caso, dizemos que $\mathcal{B}$ é a *topologia de Z induzida por $\mathcal{A}$*.

5.9.4 Produto cartesiano

Produto cartesiano

O *produto cartesiano* de dois espaços topológicos $(X, \mathcal{A})$ e $(Y, \mathcal{B})$ é o espaço cujos pontos são os pares $X \times Y$, e cujos abertos são todas as uniões (finitas e infinitas) de produtos $A \times B$ com $a \in \mathcal{A}$ e $B \in \mathcal{B}$.

5.9.5 Topologia natural de $\mathbb{R}^n$

Intervalo aberto

Um *intervalo aberto* de números reais é um conjunto da forma

$$\{\, x \;:\; a < x < b \,\}$$

onde a e b são reais com $a < b$.

Topologia natural de $\mathbb{R}^n$

Por definição, na *topologia natural* da reta $\mathbb{R}$ um subconjunto é aberto se e somente se ele é a união (possivelmente infinita) de intervalos abertos.

Para o espaço $\mathbb{R}^n$, com $n > 1$, a *topologia natural* é a definida pela regra do produto cartesiano acima. Ou seja, um subconjunto de $\mathbb{R}^n$ é aberto se e somente se ele é a união (possivelmente infinita) de produtos de n intervalos abertos de $\mathbb{R}$.

Ex. 5.44: Se c é um ponto do espaço cartesiano $\mathbb{R}^n$, e r um real positivo, a *bola (aberta) com centro c e raio r* é o conjunto de pontos

$$\{\, p \in \mathbb{R}^n \;:\; |p - c| < r \,\}$$

onde $|x|$ é o comprimento (euclidiano) do vetor x. Prove que, do ponto de vista topológico, bolas redondas são tão boas quanto bolas quadradas; ou seja,

(a) toda bola aberta é um subconjunto aberto de $\mathbb{R}^n$.

(b) na topologia natural de $\mathbb{R}^n$, um subconjunto é aberto se e somente se ele é a união (possivelmente infinita) de bolas abertas.

Se Z é um subconjunto de $\mathbb{R}^{\ltimes}$, a *topologia natural* de Z é a induzida sobre Z pela topologia natural de $\mathbb{R}^{\ltimes}$.

Topologia natural da esfera

Em particular, a topologia natural da esfera $\mathbb{S}^{\nvDash}$ é a induzida na mesma pela topologia natural de $\mathbb{R}^{|\nvDash}$. Identificando-se os pontos de $\mathbb{S}^{\nvDash}$ com os do plano projetivo orientado $\mathbb{T}^{\nvDash}$, conforme descrito na seção 2.3.2, esta mesma regra define *topologia natural* para $\mathbb{T}^{\nvDash}$.

Sempre que mencionarmos um subconjunto de pontos de $\mathbb{R}^{\ltimes}$, $\mathbb{S}^{\ltimes}$, ou $\mathbb{T}^{\ltimes}$ como um espaço topológico, sem especificar uma topologia, estaremos supondo a topologia natural.

5.9.6 Vizinhança

Vizinhança

Uma *vizinhança* de um ponto $x \in X$ é qualquer subconjunto aberto de X que inclui x. Por extensão, uma vizinhança de um conjunto $Z \subseteq X$ é qualquer subconjunto aberto de X que contém Z.

Na topologia natural de $\mathbb{R}^{\ltimes}$, toda bola com centro x é uma vizinhança de x.

5.9.7 Ponto de acumulação

Ponto de acumulação

Diremos que um elemento $p \in X$ é um *ponto de acumulação* de um conjunto $Z \subseteq X$ se e somente se qualquer vizinhança de p contém um número infinito de pontos de Z.

Convergência e limite

Diremos que uma seqüência infinita de pontos $p_1, p_2, \ldots$ de um espaço X *converge* para um conjunto de pontos $Z \subseteq X$ se qualquer vizinhança de Z contém todos os pontos da seqüência, exceto por um subconjunto finito. Se uma seqüência converge para um conjunto com um único ponto, dizemos que esse ponto é *o limite* da seqüência.

Ex. 5.45: Prove que $p \in X$ é um ponto de acumulação de $Z \subseteq X$, no espaço $(X, \mathcal{A})$, se e somente se p for um ponto de acumulação de Z em algum sub-espaço $(Y, \mathcal{B})$ de $(X, \mathcal{A})$ tal que $p \in Y$ e $Z \in \mathcal{B}$.

Ex. 5.46: Prove que $p \in X$ é um ponto de acumulação de Z se e somente se ele é o limite de uma seqüência infinita de pontos de Z.

5.9.8 Fecho, interior, e fronteira

Se $Z \subseteq X$, definimos o *interior de* Z *(em* X*)* como sendo a união de todos os subconjuntos abertos de X contidos em Z.

Ex. 5.47: Prove que o interior de Z em X é um subconjunto aberto de X.

Ex. 5.48: Prove que o interior de Z em X contém qualquer subconjunto aberto de X contido em Z.

Ex. 5.49: Prove que um ponto $x \in X$ pertence ao interior de Z em X se e somente se existe alguma vizinhança de x em X que está inteiramente contida em Z.

Definimos também o *fecho de* Z *(em* X*)* como sendo a intersecção de todos os subconjuntos fechados de X que contém Z.

Ex. 5.50: Prove que o fecho de Z em X está contido em qualquer subconjunto fechado de X que contém Z.

Ex. 5.51: Prove que o fecho de Z em X consiste de todos os elementos de X que são pontos de acumulação de Z.

A *fronteira de* Z *(em* X*)* é a diferença entre o fecho de Z e o interior de Z.

Ex. 5.52: Prove que $x \in X$ pertence à fronteira de Z em X se e somente se toda vizinhança de x em X contém algum ponto de Z e algum ponto de $X \backslash Z$.

5.9.9 Espaço quociente

Espaço quociente

Se $\equiv$ é uma relação de equivalência sobre os pontos de um espaço $(X, \mathcal{A})$, definimos o *espaço quociente* $(X, \mathcal{A})/\equiv$ como sendo $(Y, \mathcal{B})$, onde Y é o conjunto $X/\equiv$ das classes de equivalência de X por $\equiv$; e $\mathcal{B}$ é a família de todos os conjuntos $A/\equiv$, para todo $A \in \mathcal{A}$ que é união de classes de $X/\equiv$.

Colagem de polígonos

Espaços quocientes aparecem naturalmente quando uma superfície é descrita por meio da colagem de faces. Nesse caso, o espaço $(X, \mathcal{A})$ original é a união de polígonos regulares isolados, um para cada face. A relação de equivalência define a maneira como as beiradas dessas faces devem ser coladas. Isto é, se o ponto x numa aresta de um polígono deve ser colado com o ponto y em outra aresta (de outro polígono, ou do mesmo), então por definição $x \equiv y$; exceto por este caso, $x \equiv y$ se e somente se $x = y$.

Plano projetivo

Outro exemplo clássico de espaço quociente é o plano projetivo não orientado, que pode ser definido como o quociente da esfera $\mathbb{S}^{\nvDash}$ (com a topologia natural) pela relação de equivalência que identifica cada ponto da mesma com seu antípoda.

5.9.10 Separabilidade

Pontos separáveis

Dois pontos de um espaço topológico são *separáveis* se existe um conjunto aberto que contém um mas não o outro; caso contrário eles são *inseparáveis*.

Ex. 5.53: Mostre que inseparabilidade é uma relação de equivalência.

Do ponto de vista topológico, dois pontos inseparáveis são essencialmente indistinguíveis

Espaço separável

Um espaço é *separável* se todos os pares de pontos são separáveis. Não estaremos perdendo muita coisa interessante se restringirmos nossa atenção a espaços separáveis; pois o quociente de qualquer espaço por sua relação de inseparabilidade é um espaço separável, cujos abertos correspondem aos abertos do espaço original de maneira biunívoca e compatível com $\subseteq$.

5.9.11 Continuidade

Função contínua

Definimos uma *função contínua* de um espaço topológico $(X, \mathcal{A})$ para outro espaço topológico $(Y, \mathcal{B})$ como sendo uma função de X para Y tal que a imagem inversa de todo subconjunto fechado de Y é um subconjunto fechado de X.

Ex. 5.54: Seja f uma função contínua de $(X, \mathcal{A})$ para $(Y, \mathcal{B})$. Mostre que

(a) se $x \in X$ é ponto de acumulação de $Z \subseteq X$ segundo $\mathcal{A}$, então $f(x)$ é um ponto de acumulação de $f(Z)$ segundo $\mathcal{B}$.

(b) se $x \in X$ é limite da seqüência $x_1, x_2, \ldots$ de pontos de X, segundo $\mathcal{A}$, então $f(x)$ é o limite da seqüência $f(x_1), f(x_2), \ldots$ segundo $\mathcal{B}$.

5.9.12 Equivalência topológica

Homeomorfismo

Equivalência topológica

Um dos conceitos fundamentais da topologia é o de *homeomorfismo*: uma bijeção entre dois espaços é um homeomorfismo se e somente se ela leva conjuntos abertos para conjuntos abertos. Dizemos que dois espaços topológicos são *homeomorfos*, ou *equivalentes*, se e somente se existe um homeomorfismo entre eles.

Decorre da definição que uma bijeção entre dois espaços é um homeomorfismo se e somente se ela é contínua, e sua inversa também é contínua.

Ex. 5.55: Mostre que a reta $\mathbb{R}$ é equivalente ao segmento aberto $(0, 1)$.

Ex. 5.56: Mostre que o plano $\mathbb{R}^2$ é equivalente ao quadrado aberto $(0, 1) \times (0, 1)$.

Ex. 5.57: Mostre que o plano $\mathbb{R}^2$ é equivalente ao conjunto

$$\left\{ (x, y) \in \mathbb{R}^2 \ : \ x^2 + y^2 < 1 \right\}$$

Propriedade topológica

Conjunto conexo

Uma *propriedade topológica* é uma propriedade de pontos e subconjuntos de um espaço X que pode ser ser definida apenas em termos de conjuntos abertos de X (e das operações da teoria de conjuntos). Um exemplo é o fato de um conjunto de pontos ser fechado: que significa, por definição, que o complemento dele é aberto. Outro exemplo é o fato de um espaço X ser *conexo*, que significa que não existe nenhum subconjunto próprio e não-vazio de X que é ao mesmo tempo fechado e aberto. Ainda outro exemplo é fato de um conjunto de pontos Z num espaço X *separar* dois pontos $u, v \in X \setminus Z$, significando que não existe nenhum sub-espaço de $X \setminus Z$ que seja conexo e contenha u e v.

Portanto, um homeomorfismo f preserva todas as propriedades topológicas de pontos e subconjuntos de pontos. Em particular, se um espaço é conexo, qualquer espaço homeomorfo a ele também será conexo; se Z separa u e v, então $f(Z)$ separa $f(u)$ de $f(v)$. E assim por diante.

Ex. 5.58: Considere os sub-espaços do $\mathbb{R}^{\nVDash}$ definidos abaixo. Descreva cada um em palavras. Quais deles são superfícies? Quais são superfícies compactas? (Justifique as respostas.)

(a) $\{\,(x, y, z) \;:\; x^2 + y^2 + z^2 \in \{1, 2\}\,\}$.

(b) $\{\,(x, y, z) \;:\; x^2 + y^2 + z^2 \in \mathbb{N}\setminus\{\nvdash\}\,\}$.

(c) $\{\,(x, y, z) \;:\; \max\{|x|, |y|, |z|\} = 1\,\}$.

(d) $\{\,(x, y, z) \;:\; x^2 + y^2 \leq 1 \wedge z = 0\,\}$.

(e) $\{\,(x, y, z) \;:\; x^2 + y^2 < 1 \wedge z = 0\,\}$.

Capítulo 6

Problemas de proximidade

6.1 O problema do correio

Considere o seguinte *problema do correio*. Seja S um conjunto fixo de n pontos do plano $\mathbb{R}^{\nvDash}$, que chamaremos de *sítios*. Dado um ponto p genérico, qual a maneira mais eficiente de determinar o sítio de S mais próximo a p? (Veja a figura 6.1.)

problema do correio

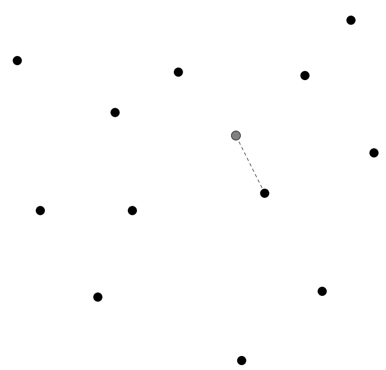


Figura 6.1: O problema do correio.

Este problema aparece em muitos contextos práticos, onde os os sítios podem ser fábricas, depósitos, lojas, postos do correio, cidades, estações meteorológicas, etc.. Em computação gráfica, em particular, os sítios podem ser posições de objetos na tela, e p a posição do cursor gráfico que está sendo usado para selecionar um desses objetos. Ou, então, os sítios são pontos onde uma função $f(x, y)$ foi amostrada, e p um ponto onde a mesma precisa ser interpolada.

A solução trivial para este problema é calcular as n distâncias $\mathrm{dist}(s, p)$ para todo $s \in S$, encontrar o mínimo desses números, e devolver o sítio s correspondente. Este algoritmo obviamente requer $\Theta(n)$ operações.

É possível melhorar este tempo? Para uma única consulta, a resposta é não: qualquer algoritmo correto precisa levar em conta todos os sítios, e portanto exige tempo pelo menos proporcional a n — isto é, $\Omega(n)$.

Índice geométrico

Entretanto, suponha que queremos resolver este problema para um grande número de pontos p, mantendo o conjunto S de sítios fixo. Neste caso, vale a pena investir algum tempo inicialmente, construindo a partir do conjunto S algum tipo de *índice geométrico* (por analogia com o índice alfabético de um livro). Com um índice adequado, é possível responder a cada consulta muito mais rapidamente do que com o algoritmo trivial. Supondo-se um número suficiente de consultas, o tempo economizado nas mesmas acaba compensando o tempo gasto na construção do índice.

6.2 O diagrama de Voronoi

Para tornar a idéia de índice geométrico mais palpável, considere o problema análogo em uma dimensão. Neste caso, os sítios — elementos de S — são pontos na reta $\mathbb{R}$, isto é, números reais. Uma consulta consiste em determinar qual o elemento de S que está mais próximo de um número real p dado.

6.2.1 Voronoi na reta

Para cada sítio $s \in S$, considere o conjunto R_s dos pontos p da reta $\mathbb{R}$ tais que o sítio mais próximo a p é s. Obviamente, determinar o sítio mais próximo a p equivale a determinar qual das regiões R_s contém p.

Para entender as regiões R_s, suponha que os sítios de S estão ordenados em ordem crescente: $s_1 < s_2 < \cdots < s_n$. Nestas condições, não é difícil ver que cada região R_{s_i} é um intervalo limitado pelos pontos v_i e v_{i+1}, onde v_i é o ponto eqüidistante entre os sítios consecutivos s_{i-1} e s_i, isto é, $v_i = \frac{1}{2}(s_{i-1} + s_i)$.

Para enquadrar as regiões extremas R_{s_1} e R_{s_n} nesta descrição, é necessário supor que $v_1 = -\infty$ e $v_{n+1} = +\infty$. Veja a figura 6.2.

Figura 6.2: Voronoi na reta.

Voronoi na reta

A coleção das regiões R_s é por definição o *diagrama de Voronoi* (DVOR) do conjunto S, assim chamado em homenagem ao matemático russo G. Voronoi.[1].

Se tivermos que responder a muitas consultas para um mesmo conjunto S, é vantajoso calcular primeiro o diagrama de Voronoi de S. Para tanto, basta ordenar os sítios em ordem crescente, e calcular os pontos separadores $v_1, v_2, \ldots, v_{n+1}$, chamados de *vértices de Voronoi*, conforme descrito acima. Feito isso, para cada ponto de consulta p, basta determinar os vértices consecutivos v_i e v_{i+1} tais que $v_i < p < v_{i+1}$, e devolver o sítio s_i.

A vantagem desta estratégia é que o intervalo (v_i, v_{i+1}) que contém o ponto p pode ser determinado rapidamente — em tempo $O(\log n)$ — por busca binária.

Custo do índice e da consulta

Por outro lado, a ordenação dos sítios requer apenas $O(n \log n)$ operações, e o cálculo dos vértices com apenas $O(n)$ operações. Portanto, o tempo gasto na construção do diagrama de Voronoi é apenas $O(n \log n)$. Note-se que este trabalho precisa ser feito uma única vez; portanto, o custo total para pré-processar os n sítios, e responder a m consultas sobre os mesmos, é

$$O(n \log n) + mO(\log n) = O((n+m)\log n)$$

[1] Os nomes *diagrama de Thiessen* e *tesselação de Dirichlet* também são às vezes usados para este conceito.

Em contraste, o algoritmo trivial — comparar cada ponto p com cada sítio — levaria tempo $O(mn)$ para atender essas mesmas consultas.

Para tornar estas análises mais concretas, suponha por exemplo $n = m = 1000$. O algoritmo trivial tem que calcular e comparar 1.000.000 distâncias entre pontos e sítios. Em contraste, o algoritmo baseado no diagrama de Voronoi, descrito acima, executaria apenas $c \times 2000 \times \log 1000 \approx c \times 20.000$ operações, onde c é uma constante pequena.

6.2.2 Voronoi no plano

Voronoi no plano

O conceito de diagrama de Voronoi, definido acima para a reta, pode ser também aplicado ao plano (ou a qualquer espaço métrico).

Especificamente, dado um conjunto S de sítios em $\mathbb{R}^{\not\models}$, o diagrama de Voronoi de S é uma coleção de regiões R_s do plano, uma para cada sítio s, tais que um ponto p está na região R_s se e somente o sítio mais próximo a p é s. Veja a figura 6.3

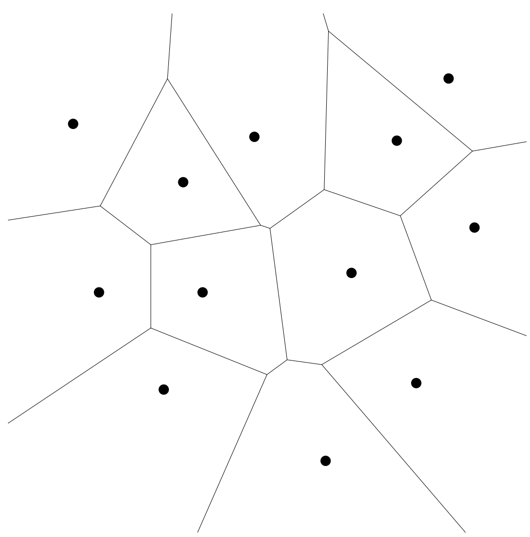


Figura 6.3: O diagrama de Voronoi no plano.

Regiões são convexas

Qual é a forma da região R_s? Considere um outro sítio qualquer t distinto de s. Um ponto p do plano está mais próximo de s do que de t se e somente se p está no semi-plano H_{st}, que contém s, e que é limitado pela reta mediatriz m_{st} do segmento st. Veja a figura 6.4.

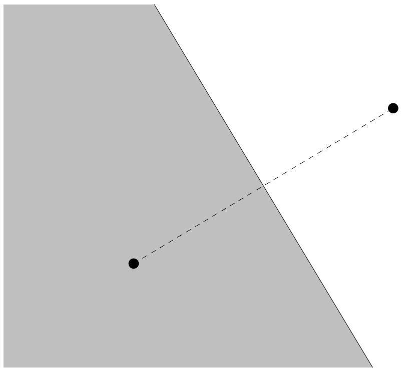


Figura 6.4: A região pontilhada está mais próxima de s do que de t.

Portanto, a região R_s é a intersecção de todos esses semiplanos, ou seja

$$R_s = \bigcap_{t \in S \setminus s} H_{st}$$

Conclui-se imediatamente desta definição que R_s é um polígono convexo que contém o sítio correspondente s.

É fácil de ver que um ponto p pertence à fronteira entre duas regiões R_s e R_t se e somente se p é eqüidistante de s e de t, e estes são os dois sítios de S mais próximos a p. O mesmo vale para os pontos na fronteira de três ou mais regiões, que são os vértices dos polígonos.

6.2.3 Complexidade do Voronoi

Como vimos acima, o Voronoi de n sítios na reta pode ser descrito de maneira compacta pela lista de seus $n-1$ vértices.

Complexidade do Voronoi

Qual a complexidade do diagrama de Voronoi no plano? À primeira vista, parece ser muito grande. Enquanto que na reta $\mathbb{R}$ cada região de Voronoi tem no máximo dois vizinhos, no plano $\mathbb{R}^2$ uma única região pode ter até $n-1$ lados. Se isso pudesse acontecer com todas as n regiões, teríamos $\Theta(n^2)$ arestas, o que impediria o uso do diagrama de Voronoi em aplicações práticas.

Felizmente, esta hipótese pessimista não é possível. Uma vez que o diagrama de Voronoi é um grafo planar, ele satisfaz a relação de Euler

$$\langle \text{vértices} \rangle - \langle \text{arestas} \rangle + \langle \text{faces} \rangle = 2$$

Usando esta identidade, e um pouco de álgebra, pode-se provar que o diagrama de Voronoi de $n \geq 3$ sítios tem no máximo $3n - 6$ arestas e $2n - 4$ vértices. Ou seja, a complexidade do diagrama é apenas proporcional ao número de sítios; e não quadrática, como temíamos.

6.2.4 Usando o diagrama

Usando o diagrama de Voronoi

Suponha que construímos o diagrama de Voronoi $\{ R_s : s \in S \}$ dos sítios dados. Portanto, para encontrar o sítio mais próximo a um ponto dado p basta determinar a região R_s do diagrama que contém p.

Esta estratégia não compensa se tivermos que testar todas as regiões R_s, uma por uma. (Isso certamente levaria mais tempo do que comparar as distâncias entre p e os sítios!) No caso unidimensional, uma simples busca binária permitia resolver este problema em tempo $O(\log n)$. Será que existe ago equivalente à busca binária em duas dimensões?

Felizmente, a resposta é *sim*. Os detalhes são complicados demais para discutir neste curso, mas a conclusão é que, mesmo em duas dimensões, a região R_s que contém o ponto p pode ser encontrada em tempo $O(\log n)$.

Mais adiante veremos que o diagrama de Voronoi de n sítios pode ser calculado em tempo $O(n \log n)$. Portanto, o custo total para responder a m consultas sobre n sítios no plano é o mesmo do caso unidimensional, ou seja $O((m + n) \log n)$.

6.3 O diagrama de Delaunay

Arestas do Voronoi

Seja S um conjunto de sítios no plano. Diremos que dois sítios s, t de S são *vizinhos de Voronoi* se as respectivas regiões R_s, R_t do diagrama de Voronoi tem um lado em comum. Lidando cada par de sítios vizinhos com um segmento de reta, obtemos o *diagrama de Delaunay* (DDEL) de S. Veja a figura 6.5.

Diagrama de Delaunay

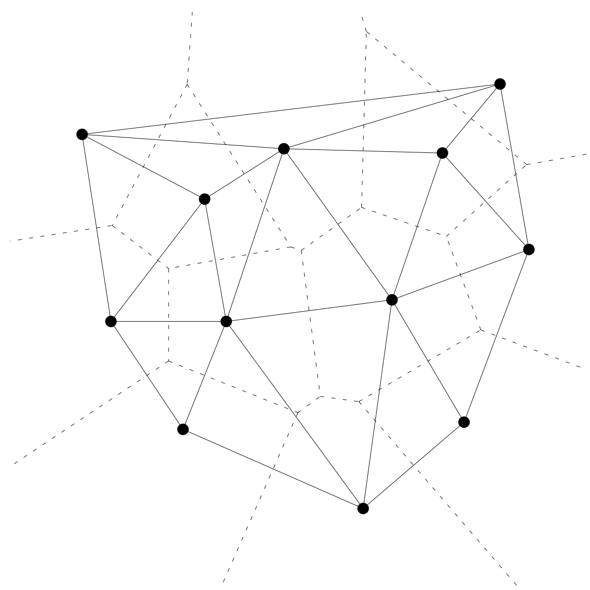


Figura 6.5: O diagrama de Delaunay.

6.3.1 Propriedades fundamentais

Sejam s e t os dois extremos de uma aresta do diagrama de Delaunay de S. Pela definição do DDEL, as regiões de Voronoi R_s e R_t devem ter um lado e em comum. Seja c um ponto qualquer no interior de e (excluindo seus extremos). Pela definição do DVOR, c está situado à mesma distância r dos dois sítios s e t; e a distância de c a qualquer outro sítio u é estritamente maior que r.

Se considerarmos o círculo de centro c e raio r, podemos concluir o seguinte fato:

Arestas do Delaunay: círculo vazio

Propriedade 6.1 (Círculo Vazio) *Dois sítios são adjacentes no diagrama de Delaunay se e somente se existe algum círculo que passa por eles, e não contém ou passa por nenhum outro sítio do diagrama.*

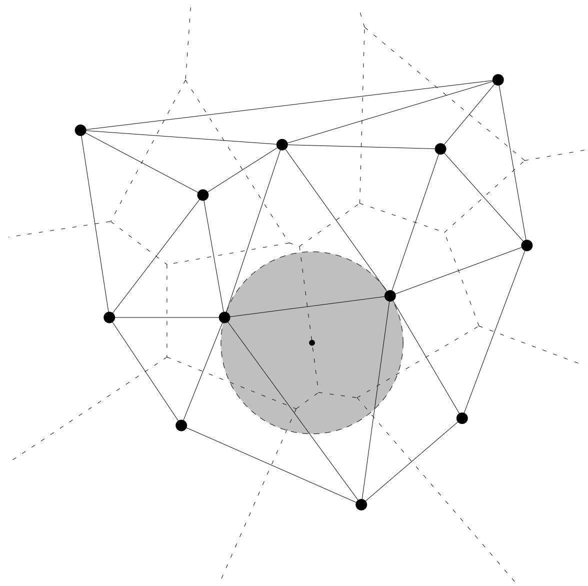


Figura 6.6: A propriedade do Círculo Vazio.

A partir desta propriedade fundamental, podemos derivar facilmente muitas outras propriedades importantes do DDEL.

Por exemplo, podemos provar que as arestas do DDEL não se cruzam ou tocam, a não ser nos seus extremos. Isto significa que os diagramas de Voronoi e de Delaunay são duais, não apenas como grafos, mas como diagramas. Ou seja, todo vértice, aresta, ou face de um deles corresponde respectivamente a uma face, aresta, ou vértice do outro, preservando as relações de incidência entre esses elementos.

Vértices do Voronoi

Em particular, se v é um vértice do diagrama de Voronoi, e $R_{s_1}, R_{s_2}, \ldots, R_{s_k}$ são as regiões incidentes a v, en ordem anti-horária, então v deve ser eqüidistante dos sítios $s_1, s_2, \ldots, s_k$, e qualquer outro sítio está mais afastado de v do que eles. Se considerarmos o círculo com centro v que passa por esses k sítios, concluímos que

Propriedade 6.2 (Círculo Circunscrito) *Três ou mais sítios definem uma face do diagrama de Delaunay se e somente se existe algum círculo que passa por eles, e não contém ou passa por nenhum outro sítio do diagrama. O centro desse círculo é o vértice do diagrama de Voronoi que corresponde a essa face.*

Faces do Delaunay: círculo circunscrito

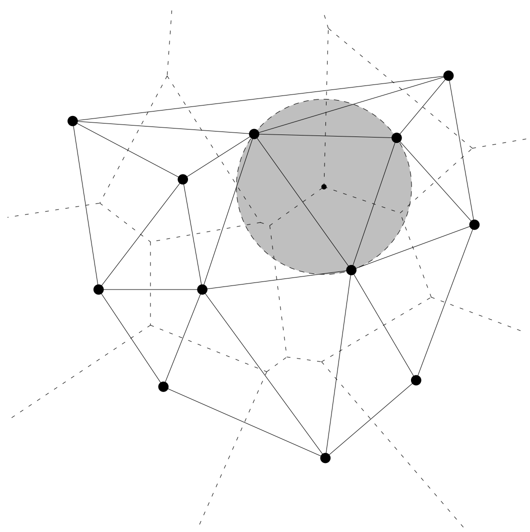


Figura 6.7: Faces do diagrama de Delaunay.

Portanto, toda face do DDEL é um polígono simples inscrito em algum círculo, e portanto um polígono convexo.

6.3.2 Triangulação de Delaunay

Triangulação de Delaunay

Se os sítios são gerados aleatoriamente, ou são sujeitos a erros de medida aleatórios, a probabilidade de quatro ou mais sítios pertencerem a um mesmo círculo é nula. Nessas condições, as faces do DDEL são invariavelmente triângulos, e o DDEL é também chamado de *triangulação de Delaunay* (TDEL).

Mesmo nos casos em que o DDEL tem faces com mais de três vértices, ele pode ser facilmente transformado numa triangulação; basta acrescentar $k-3$ diagonais (não cruzadas) a cada face com $k>3$ lados.

6.3.3 Eqüiangularidade

As triangulações de Delaunay, obtidas como descrito acima, são bastante usadas interpolação de dados, modelagem de sólidos, cálculo de elementos finitos, e outras áreas da análise numérica.

Ângulos uniformes

A razão para essa popularidade é que os erros de aproximação cometidos nessas áreas dependem da forma dos triângulos usados, sendo em geral tanto maiores quanto menos eqüiláteros forem os triângulos. Acontece que, dentre todas as triangulações do plano com um conjunto fixo S de vértices, a TDEL é a que produz os triângulos mais eqüiláteros possíveis.

Em particular, considere duas faces adjacentes quaisquer stu e tsv de uma triangulação de Delaunay T, tais que o quadrilátero $usvt$ seja convexo. A propriedade 6.2 diz que o sítio u tem que estar fora do círculo tsv, e v tem que estar fora do círculo stu. Pode-se deduzir daí que a soma dos ângulos $t\hat{u}s$ e $s\hat{v}t$ é menor do que 180°. Por outro lado, sabemos que a soma dos quatro ângulos internos de um quadrilátero é 360°; portanto, a soma dos ângulos $u\hat{s}v$ e $v\hat{t}u$ é maior do que 180°.

Compare agora a triangulação de Delaunay T com a triangulação alternativa T' dos mesmos sítios, obtida de T eliminando-se a aresta st e acrescentando-se a aresta uv. Ou seja, T e T' diferem apenas na maneira de cortar o quadrilátero $usvt$ em dois triângulos. Com um pouco de álgebra e geometria, é possível mostrar que os dois triângulos usados por T são mais eqüiláteros que os dois usados por T', se medirmos a "eqüilateralidade" de um triângulo pelo tamanho do seu menor ângulo interno.

Aliás, esta propriedade local é suficiente para caracterizar as triangulações de Delaunay, dentre todas as triangulações do mesmo conjunto de sítios:

Ângulos do quadrilátero

Propriedade 6.3 (Eqüiangularidade) *Seja T uma triangulação de um conjunto de sítios S, cujo contorno é a envoltória convexa de S. Nesse caso, T é uma triangulação de Delaunay se e somente se o vértice u está fora do círculo tsv, para todos os pares de triângulos adjacentes stu e tsv tais que o quadrilátero $usvt$ é convexo.*

Note que que o custo de testar um quadrilátero $usvt$ é constante, e número de quadriláteros a testar é limitado pelo número de arestas st

da triangulação. Portanto, esta propriedade nos permite testar se uma triangulação de n sítios é Delaunay em tempo $O(n)$; enquanto que um teste ingênuo, baseado na propriedade 6.1, levaria tempo $\Omega(n^2)$.

6.3.4 Troca de arestas

Delaunay por troca de arestas

Esta caracterização também fornece um algoritmo extremamente simples para construir uma TDEL de um conjunto de sítios S. Basta obter uma triangulação qualquer de S, e verificar se todas suas arestas st satisfazem a propriedade 6.3. Caso a propriedade falhe para algum quadrilátero $usvt$, removemos a aresta st e acrescentamos a aresta uv; e continuamos repetindo os testes e trocas até que todas as arestas estejam consistente.

Note que a troca de st por uv invalida quaisquer testes anteriores das arestas us, sv, vt, e tu; portanto, após a troca, essas arestas devem ser testadas de novo.

Pode-se mostrar que este processo termina depois de no máximo $\Theta(n^2)$ trocas, no pior caso. Como veremos, se o problema é construir o DDEL completo, há algoritmos muito mais rápidos. Mas este algoritmo é uma maneira excelente de recalcular o diagrama depois de uma mudança local no conjunto de sítios, em particular depois da adição de um sítio. Neste caso, o algoritmo acima leva tempo mínimo, proporcional ao número de arestas da triangulação que realmente precisam ser alteradas.

6.4 Construção do Delaunay

Os diagramas de Delaunay e de Voronoi são basicamente equivalentes: Se soubermos como construir um, podemos obter outro com apenas $O(n)$ de esforço.

Isto posto, é em geral aconselhável trabalhar com o diagrama de Delaunay, enquanto for possível. A razão principal é que os vértices do DDEL são os próprios sítios de entrada, e portanto finitos. Em contraste, os vértices do DVOR podem estar no infinito, e tem que ser calculados por métodos numéricos (resolução de um sistema linear de dimensão 2×2), sendo portanto sujeitos a erros de arredondamento e *overflow*.

6.4.1 Poliedro de Delaunay

Delaunay: linearização dos cálculos

A construção do diagrama de Delaunay (ou de Voronoi) a partir da definição parece exigir o cálculo de expressões não-lineares complicadas, como a construção do círculo passando por três pontos dados. Entretanto, por uma transformação adequada dos dados, podemos substituir todas essas operações por simples cálculos de determinantes 3×3 e 4×4.

Projeção no parabolóide

A transformação consiste em substituir todo sítio s com coordenadas (x, y) pelo ponto s^* do $\mathbb{R}^3$ com coordenadas (x, y, x^2+y^2). Geometricamente, isso corresponde a projetar sítio verticalmente, de baixo para cima, até a superfície Λ do parabolóide definido pela equação $z = x^2+y^2$. Portanto, esta transformação substitui o conjunto de sítios originais S por um conjunto S^* de pontos no parabolóide Λ. Veja a figura 6.8

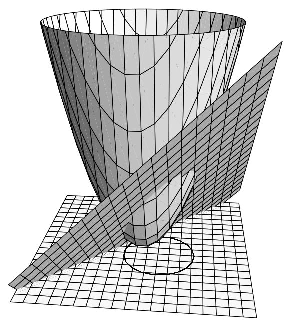


Figura 6.8: Transformação do parabolóide.

Constata-se que esta transformação também associa a todo círculo C de $\mathbb{R}^2$ um plano C^* de $\mathbb{R}^3$, tal que C passa por um sítio s se e somente se o plano C^* passa pelo ponto s^*. Portanto, o cálculo do círculo qua passa por três pontos de $\mathbb{R}^2$ corresponde ao cálculo do plano que passa por três pontos de $\mathbb{R}^3$.

Mais ainda, prova-se que um sítio s está dentro do círculo C se e somente se o ponto s^* está abaixo do plano C^*. Combinando este fato com a propriedade do círculo vazio, concluímos que dois sítios s, t de S são ligados por uma aresta do DDEL se e somente se existe um plano no $\mathbb{R}^3$ que passa por suas imagens s^* e t^*, e que que passa por baixo de todos os demais pontos de S^*.

Por sua vez, esta é a condição para que o segmento s^*t^* seja uma aresta da envoltória convexa H^* de S^*, o menor poliedro convexo que contém S^*. Mais exatamente, é a condição para que s^*t^* seja um aresta *da parte inferior* de H^*; isto é, da parte que estaria na sombra se o sol estivesse em $(0, 0, +\infty)$.

A correspondência acima descrita fornece um algoritmo para a construção do diagrama de Delaunay. Basta calcular as imagens S^* dos sítios dados, calcular a envoltória convexa de S^*, e projetar de volta sua parte inferior sobre o plano (x, y).

Portanto, vamos deixar de lado os diagrama de Delaunay por algum tempo, e considerar o problema da envoltória convexa em três dimensões.

6.4.2 Envoltória convexa

Envoltória convexa no espaço

Seja então S um conjunto de pontos (sítios) no espaço $\mathbb{R}^3$. A envoltória convexa de S pode ser eficientemente calculada pelo método de divisão e conquista. Primeiro, dividimos o conjunto S em duas partes U e V, com aproximadamente o mesmo número de pontos, por algum plano de $\mathbb{R}^3$. (Por exemplo, podemos ordenar os pontos pela sua coordenada x, e a dividir a lista resultante ao meio.) Em seguida, calculamos as envoltórias convexas H_U e H_V desses dois conjuntos, recursivamente. Finalmente, determinamos o menor poliedro convexo H que contem os dois poliedros H_U e H_V. Veja a figura 6.9.

A recursão acima termina quando o conjunto de pontos S tem apenas um elemento; nesse caso, a envoltória convexa se reduz a esse ponto.

6.4.3 Juntando envoltórias

Juntando envoltórias convexas

Intuitivamente, a última etapa do algoritmo acima — calcular a envoltória convexa única H de H_U e H_V — equivale a envolver os dois poliedros por um balão elástico. O poliedro H consiste das duas metades “externas” de H_U e H_V, ligadas uma “manga” constituída de faces novas (geralmente triângulos).

Toda face da “manga” (geralmente, um triângulo) tem pelo menos um vértice em cada um dos dois poliedros H_U e H_V. Cada face é colada na face seguinte por exatamente uma aresta, de modo a formar um anel de faces.

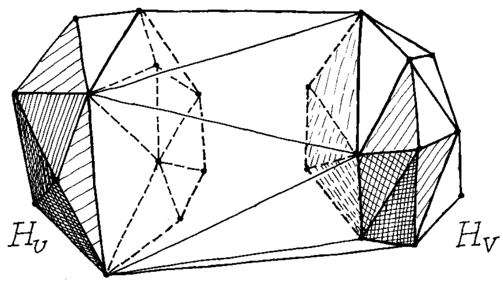


Figura 6.9: Envoltória convexa no $\mathbb{R}^3$.

Calculando a "manga"

O algoritmo para determinar essas faces simula o processo de embrulhar os dois poliedros com uma folha de papel. Mais precisamente, ele simula o movimento de um plano P que "rola" em torno de H_U e H_V, sempre tocando em pelo menos um vértice de cada poliedro, e sempre mantendo os dois poliedros no mesmo semi-espaço de P.

Plano tangente comum

Portanto, o primeiro passo é encontrar um plano P nessas condições. Isto é, P toca em exatamente dois pontos u_0, v_0 de S, um em cada poliedro, e deixa todos os outros pontos de S no mesmo semi-espaço de P.

Rolando o plano

Feito isso, o próximo passo é rodar o plano P em torno da reta u_0v_0, até que este encoste em algum outro ponto de S. Pode-se provar que o primeiro ponto w_0 atingido pelo plano é vizinho de u_0 em H_U, ou de v_0 em H_V; o que permite determiná-lo sem muito esforço.

Uma vez encontrado o vértice $w0$, temos a primeira face da "manga", ou seja, o triângulo $u_0v_0w_0$. Para prosseguir a busca, devemos substituir um dos u_0 ou $v0$ por w_0, de forma a obter outro par de vértices u_1, v_1, um em cada poliedro.

Com isto, estamos novamente em condições de repetir o passo anterior. Ou seja, podemos agora rodar o plano P em torno da reta u_1v_1, ainda no mesmo sentido, até atingir outro ponto w_1 de S. A segunda face da "manga" será então $u_1v_1w_1$.

Substituindo um dos dois vértices u_1, v_1 pelo vértice w_1, obtemos a reta-pivô seguinte, u_2v_2; e assim por diante. Este processo termina quando o plano rolante P volta à posição de partida, isto é, quando o novo par de pivôs u_k, v_k é igual ao par inicial u_0, v_0.

Corte e costura

Nesse ponto, tudo o que resta a fazer é recortar os dois poliedros ao longo dos caminhos $u_0u_1 \ldots uk$ e $v_1v_2 \ldots vk$, descartar as metades que estão voltadas para dentro, e juntar as outras duas metades às faces da "manga."

Análise de custo

Não é difícil provar que o processo de cálculo da "manga" e junção dos dois poliedros, como descrito acima, termina em tempo $O(n)$, onde n é o número total de vértices dos dois poliedros. Portanto, o custo total $t(n)$ para construir a envoltória convexa de n pontos de $\mathbb{R}^3$, pelo algoritmo recursivo da seção 6.4.2, é dado pela recorrência

$$t(n) = 2t(n/2) + O(n)$$

cuja solução é $t(n) = O(n \log n)$.

6.4.4 A bolha ambulante

Recapitulando, nas seções anteriores vimos um algoritmo de custo $O(n \log n)$ para o cálculo do diagrama de Delaunay de n sítios no plano $\mathbb{R}^2$. O algoritmo consiste em transformar os sítios dados em pontos do espaço $\mathbb{R}^3$, calcular a envoltória convexa destes pontos por divisão-e-conquista, e projetar a parte inferior do poliedro resultante no $\mathbb{R}^2$.

Trabalhando em duas dimensões

Em vez de levantar os sítios para o espaço $\mathbb{R}^3$, podemos "projetar" o algoritmo de envoltória convexa sobre o plano $\mathbb{R}^2$. O resultado é um algoritmo de divisão-e-conquista que constrói o DDEL trabalhando diretamente com os sítios dados.

Primeiro, dividimos o conjunto S em duas partes U e V, com aproximadamente o mesmo número de pontos, por alguma reta m de $\mathbb{R}^2$. (Por exemplo, podemos ordenar os pontos pela sua coordenada x, e a dividir a lista resultante ao meio.) Em seguida, construímos o diagrama de Delaunay D_U e D_V para cada parte, recursivamente. Finalmente, juntamos esses dois diagramas num único diagrama D para o conjunto S todo.

Vimos na seção 6.4.3 que a envoltória comum de dois poliedros convexos pode ser determinada com o auxílio de um "plano rolante", que gira em torno dos dois poliedros, mantendo sempre contato com pelo menos um vértice de cada um. As arestas novas da envoltória comum são os pares de vértices usados como pivôs no decorrer dessa rolagem.

A bolha ambulante

Traduzindo esses conceitos para diagramas de Delaunay no $\mathbb{R}^2$, concluímos que a junção dos dois diagramas parciais D_U e D_V pode ser feita com o auxílio de uma *bolha ambulante*: um círculo que percorre o espaço entre os dois diagramas, mantendo sempre contato com pelo menos um sítio de cada um, e sem nunca incluir sítios em seu interior. Veja a figura 6.10.

Os pares de sítios tocados pela bolha definem as arestas que precisam ser acrescentadas entre D_U e D_V para para formar o diagrama de S. Nesse processo, é necessário remover as arestas de D_U e D_V que são cortadas pelas novas arestas.

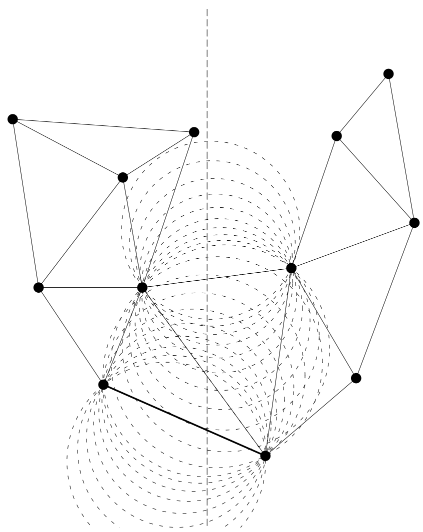


Figura 6.10: O algoritmo da bolha ambulante.

6.5 Aplicações do DDEL

Aplicações do diagrama de Delaunay

O diagrama de Delaunay, e triangulações do mesmo, tem inúmeros usos práticos, além de serem um caminho natural para o cálculo do diagrama de Voronoi. Já mencionamos seu uso em análise numérica e modelagem. Nesta seção, estudaremos o uso do DDEL como base para outros algoritmos geométricos

6.5.1 Diagrama de Gabriel

Problema da reconstrução de contornos

Em visão por computador e reconhecimento de padrões, um problema comum é reconstruir o contorno de uma figura plana a partir de uma coleção S de pontos isolados (*sítios*) na fronteira da mesma.

Em geral, a parte mais delicada deste problema é determinar a ordem dos pontos dados ao longo do contorno; ou seja, quando é que um par de sítios deve ser ligado por uma aresta.

Este problema não tem solução bem definida, pois em geral existe um número enorme de maneiras de ligar os pontos de S de modo a formar um contorno sem cruzamentos; e não existe um critério óbvio e universal para preferir uma delas às outras.

Há muitas heurísticas que podem ser usadas para restringir o número de contornos em consideração. Uma classe popular de heurísticas começam construindo algum grafo planar G, cujos vértices são os pontos dados, e cujas arestas são segmentos de reta que não se cruzam. Daí em diante, são considerados apenas contornos que são subgrafos de G; isso garante automaticamente que as arestas desses contornos não se cruzam.

Diagrama de Gabriel

Um grafo que pode ser usado para esse fim é o *diagrama!de Gabriel*, no qual dois sítios s, t são ligados por uma aresta se e somente se o ângulo sut é agudo, para qualquer outro sítio v de S.

É fácil ver que esta condição equivale a dizer que o círculo cujo diâmetro é o segmento st não contém nenhum outro sítio de S. Veja a figura 6.11.

Relação entre DGAB *e* DDEL

A partir desta definição, concluímos imediatamente que toda aresta do diagrama de Gabriel é também uma aresta do diagrama de Delaunay. Por sua vez, isso implica que o diagrama de Gabriel é um grafo planar; e, portanto, tem apenas $O(n)$ arestas.

Na verdade, pode-se verificar que uma aresta (s, t) do DDEL está no grafo de Gabriel se e somente se o segmento st cruza a aresta do diagrama de Voronoi que separa as regiões de s e t. Esta observação nos dá um algoritmo para calcular o grafo de Gabriel a partir do diagrama de Delaunay, em tempo $O(n)$.

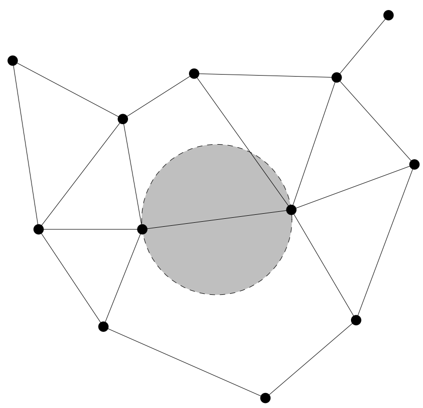


Figura 6.11: O diagrama de Gabriel.

6.5.2 Grafo de vizinhança relativa

Grafo de vizinhança relativa

Outro grafo bastante usado em reconhecimento de padrões (mais ainda do que o diagrama de Gabriel) é o *grafo de vizinhança relativa* (GVREL). Por definição, dois sítios s, t são ligados por uma aresta do GVREL quando nenhum outro sítio está mais próximo de ambos do que eles estão próximos entre si. Formalmente, se $\mathrm{dist}(s,t) = d$, então s e t são vizinhos se e somente se, para qualquer outro sítio u, tivermos $\mathrm{dist}(u,s) > d$ ou $\mathrm{dist}(u,t) > d$.

Uma definição equivalente, talvez mais fácil de visualizar, é a seguinte. Considere a região L_{st} do plano que é a intersecção dos discos com centros nos dois sítios e raio igual à distância d entre eles. Os dois sítios s, t estão ligados por uma aresta no GVREL se e somente se a lente L_{st} não contém nenhum outro sítio de S. Veja a figura 6.12.

Relação entre DGAB *e* GVREL

Note que a lente L_{st} contém o disco D_{st} cujo diâmetro é o segmento st. Portanto, toda aresta do GVREL é também pertence aos diagramas de Gabriel e de Delaunay. Ou seja, o grafo de vizinhança relativa é um subgrafo (próprio) desses dois. Segue-se daí que of GVREL também é planar e tem apenas $O(n)$ arestas.

Na verdade, uma vez construído o diagrama de Delaunay, é possível calcular o GVREL a partir dele, em tempo $O(n)$.

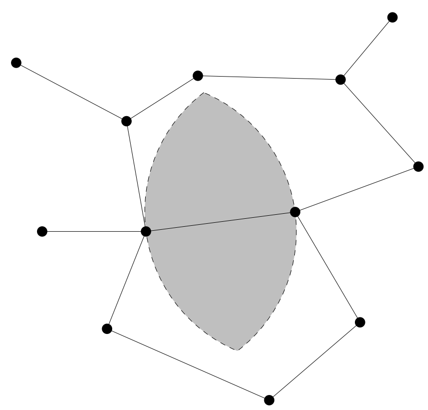


Figura 6.12: O grafo de vizinhança relativa.

Apesar de ter relativamente poucas arestas, o grafo de vizinhança relativa é sempre conexo. Para mostrar este fato, basta mostrar que, para qualquer subconjunto U de S, próprio e não vazio, existe pelo menos uma aresta que liga algum sítio de U a algum sítio de $V = S\backslash U$. Dentre todos os pares de sítios em $U \times V$, seja (u, v) um par tal que $\operatorname{dist}(u, v) = d$ é mínima. Com esta escolha, para qualquer outro sítio w em S, temos que $\operatorname{dist}(w, v) > d$ se w está em U, e $\operatorname{dist}(u, w) > d$ se w está em V. Mas isto significa que u e v estão ligados por uma aresta, pela definição de GVREL. Concluímos portanto que o GVREL é conexo.

6.5.3 Árvore conectora mínima

Conector de comprimento mínimo

Suponha que queremos ligar um certo número de centrais telefônicas por por cabos, de modo que seja possível telefonar de qualquer estação para qualquer outra. Qual é a maneira de fazer isso, gastando a menor quantidade possível de cabo?

Uma formulação mais abstrata desse problema é a seguinte: dado um conjunto S de sítios no plano, determinar um grafo cujos vértices são esses sítios e cujas arestas são curvas no plano, tal que a soma dos comprimentos das arestas seja mínima.

Árvore conectora mínima

É fácil provar que o grafo que satisfaz estas condições deve ser conexo e sem ciclos — ou seja, uma árvore; e que suas arestas são segmentos de retas que não se cruzam. Essa árvore é a *árvore conectora mínima* (ACMIN) do conjunto S. Veja a figura 6.13.

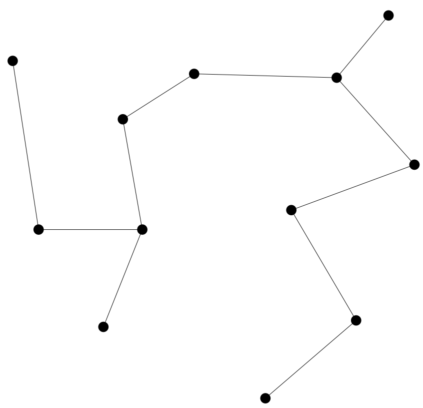


Figura 6.13: A árvore conectora mínima.

Note que a definição não nos permite usar, como vértice da ACMIN, nenhum outro ponto do plano além dos sítios dados. Por exemplo, considere três sítios formando um triângulo eqüilátero de lado 1, como na figura 6.14. Neste caso, a árvore à esquerda, com comprimento total $\sqrt{3}$, não é considerada válida. As únicas respostas válidas são as três árvores à direita, de comprimento 2.

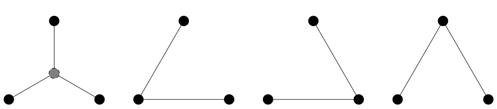


Figura 6.14: Arvores conectoras mínimas.

Vértices de Steiner

A exclusão do uso de vértices adicionais (*vértices de Steiner*) pode aumentar em até 50% o comprimento total do grafo. Entretanto, essa regra é justificável em muitas aplicações. Além do que, se sem essa regra o problema fica muito mais difícil.

Supondo que vértices de Steiner sejam proibidos, podemos reduzir a construção da ACMIN ao seguinte problema da teoria de grafos. Seja G o grafo cujos vértices são os sítios, e cujas arestas são todos os pares de sítios distintos (isto é, seja G o grafo completo com vértices S.) Suponha que cada aresta (u, v) de G tem um v“custo” proporcional a $\operatorname{dist}(u, v)$. Determine um subgrafo de G que use todos os vértices, seja uma árvore, e tenha custo total mínimo.

Este é um problema clássico da teoria de grafos, que pode ser resolvido em tempo $O(m \log m)$, onde m é o número de arestas do grafo. O problema é que m neste caso é $\binom{n}{2} = \Theta(n^2)$. É possível melhorar este tempo?

Relação entre ACMIN *e* GVREL

Felizmente, pode-se provar que a árvore conectora mínima é um subgrafo do grafo de vizinhança relativa, e portanto do diagrama de Delaunay. Portanto, em vez do grafo completo G acima, podemos usar o diagrama de Delaunay. Como o DDEL tem apenas $O(n)$ arestas, o custo de calcular a ACMIN por este caminho é apenas $O(n \log n)$, incluindo a construção do DDEL.

6.5.4 Vizinho mais próximo

Problema do vizinho mais próximo

Considere o seguinte problema: dado um conjunto S de pontos em $\mathbb{R}^{\not\models}$, encontrar para cada $s \in S$ o ponto $t \in S$ mais próximo a s. (Em outras palavras, considere o problema do sítio mais próximo, onde os pontos de consulta p são os próprios sítios de S.)

O algoritmo trivial — para cada sítio s, compare todos os sítios t — tem custo $\Theta(n^2)$. Entretanto, é fácil mostrar que, se t é o sítio mais próximo ao sítio s, então s e t devem ser adjacentes na árvore conectora de custo mínimo, e portanto no diagrama de Delaunay.

Portanto, basta construir DDEL(S) e, para cada sítio s, examinar apenas os sítios t que são adjacentes a s no mesmo. Como há apenas $O(n)$ arestas, o custo total deste método é $O(n \log n) + O(n) = O(n \log n)$.

Bibliografia

[AHU74] A. V. Aho, J. E. Hopcroft, e J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1974.

[Avi80] D. Avis. Comments on a lower bound for convex hull determination. *Inform. Process. Lett.*, 11:126, 1980.

[Baa88] S. Baase. *Computer Algorithms.* Addison-Wesley, 1988(?).

[BO79] J. L. Bentley e T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28(9):643–647, setembro de 1979.

[BO83] M. Ben-Or. Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pp. 80–86, 1983.

[CLR90] T. H. Cormen, C. E. Leiserson, e R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[Gra72] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, 1:132–133, 1972.

[Lee83] D. T. Lee. On finding the convex hull of a simple polygon. *Internat. J. Comput. Inform. Sci.*, 12:87–98, 1983.

[Lim76] Elon Lages Lima. *Elementos de topologia geral.* Livros técnicos e científicos, Rio de Janeiro, RJ, 1976.

[Man89] U. Manber. *Algorithms: A Creative Approach.* Addison-Wesley, Reading, MA, 1989.

[PS85] F. P. Preparata e M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.

[Sed83] R. Sedgewick. *Algorithms.* Addison-Wesley, Reading, MA, 1st edição, 1983.

[ST83] D. D. Sleator e R. E. Tarjan. Self-adjusting binary trees. In *Proc. 15th ACM Symp. on Theory of Computing*, pp. 235–245, 1983.

[vE80] P. van Emde Boas. On the $\Omega(n \log n)$ lower-bound for convex hull and maximal vector determination. *Inform. Process. Lett.*, 10:132–136, 1980.

Índice