

Projeto e Análise de Algoritmos*

Redução entre problemas

Segundo Semestre de 2019

*Criado por C. de Souza, C. da Silva, O. Lee, F. Miyazawa et al.

A maior parte deste conjunto de slides foi inicialmente preparada por Cid Carvalho de Souza e Cândida Nunes da Silva para cursos de Análise de Algoritmos. Além desse material, diversos conteúdos foram adicionados ou incorporados por outros professores, em especial por Orlando Lee e por Flávio Keidi Miyazawa. Os slides usados nessa disciplina são uma junção dos materiais didáticos gentilmente cedidos por esses professores e contêm algumas modificações, que podem ter introduzido erros.

O conjunto de slides de cada unidade do curso será disponibilizado como guia de estudos e deve ser usado unicamente para revisar as aulas. Para estudar e praticar, leia o livro-texto indicado e resolva os exercícios sugeridos.

Lehilton

Agradecimentos (Cid e Cândida)

- ▶ Várias pessoas contribuíram **direta ou indiretamente** com a preparação deste material.
- ▶ Algumas destas pessoas cederam gentilmente seus arquivos digitais enquanto outras cederam gentilmente o seu tempo fazendo correções e dando sugestões.
- ▶ Uma lista destes “colaboradores” (**em ordem alfabética**) é dada abaixo:
 - ▶ Célia Picinin de Mello
 - ▶ Flávio Keidi Miyazawa
 - ▶ José Coelho de Pina
 - ▶ Orlando Lee
 - ▶ Paulo Feofiloff
 - ▶ Pedro Rezende
 - ▶ Ricardo Dahab
 - ▶ Zanoni Dias

Cotas para um problema

Formalizando problema

Como formalizar um problema **genericamente**?

Definição (Problema computacional)

Um problema computacional é uma **relação** $P \subseteq \mathcal{I} \times \mathcal{S}$, onde

- ▶ \mathcal{I} é o conjunto de entradas e
- ▶ \mathcal{S} é o conjunto de saídas.

Algoritmo para um problema

Definição

*Dizemos que um algoritmo **ALG resolve** um problema $P = (\mathcal{I}, \mathcal{P})$ se para toda entrada $I \in \mathcal{I}$, ele devolve uma saída $S \in \mathcal{S}$ tal que $(I, S) \in P$.*

- ▶ escrevemos $I \in P$ para representar uma entrada
- ▶ escrevemos $A(I)$ para representar a saída do algoritmo
- ▶ denotamos por n o “tamanho” de I
- ▶ normalmente n é o número de bits de I

Revisitando a complexidade de um algoritmo

Seja **ALG** um algoritmo para um problema P e n um parâmetro.

Notação O :

- ▶ Se o algoritmo leva tempo **no máximo** $f(n)$ para toda entrada de tamanho n , então dizemos que **ALG** executa em tempo $O(f(n))$.

Notação Ω :

- ▶ Se o algoritmo leva tempo **pelo menos** $g(n)$ para alguma entrada de tamanho n , então dizemos que **ALG** executa em tempo $\Omega(g(n))$.

Cotas superior e inferior de um problema

Seja P um problema e seja n um parâmetro:

Definição (Cota superior)

Uma função $f(n)$ é chamada de cota superior para P se *existe algum algoritmo* que resolve P em tempo $O(f(n))$.

Definição (Cota inferior)

Uma função $g(n)$ é chamada de cota inferior para P se *todo algoritmo* que resolve P leva tempo $\Omega(f(n))$.

Algoritmo ótimo

Um algoritmo ALG é **ótimo** para um problema P se:

1. ALG resolve P em tempo $O(f(n))$ e
2. $f(n)$ é uma cota inferior de P .

- ▶ **HEAP-SORT** e **MERGE-SORT** são ótimos para ordenação:
 - ▶ eles têm complexidade $O(n \lg n)$
 - ▶ ordenação tem cota inferior $\Omega(n \lg n)$
- ▶ **BUSCA-BINARIA** é ótimo para busca em vetor ordenado:
 - ▶ tem complexidade $O(\lg n)$
 - ▶ qualquer algoritmo leva tempo $\Omega(\lg n)$

Comparando problemas

Como comparamos dois algoritmos para **um único problema**?

- ▶ comparamos a complexidade de cada algoritmo
- ▶ descobrimos se um algoritmo é mais “rápido” que outro

E se quisermos comparar **dois problemas** A e B ?

- ▶ queremos descobrir se então A é mais “fácil” do que B
- ▶ podemos comparar as cotas de cada algoritmo

Exemplo: Achar o máximo é mais **fácil** que ordenar um vetor!

- ▶ máximo tem cota superior $O(n)$
- ▶ ordenação tem cota inferior $\Omega(n \lg n)$

Reduções

Uma analogia

Um certo editor de literatura internacional é especialista em publicar livros em português. Ele conta com um time de tradutores, entre os quais:

- ▶ Cook, responsável por traduzir de **inglês para português**.
- ▶ Levin, responsável por traduzir de **russo para inglês**.

Em uma edição especial, ele gostaria de publicar Crime e Castigo; como então traduzir de **russo para português**?

- ▶ Em geral, lidamos com problemas bem conhecidos
- ▶ Mas eventualmente, topamos com problemas novos

Pergunta: como relacionar esses problemas?

Redução

Problema A:

- ▶ Instância: I_A
- ▶ Solução: S_A

Problema B:

- ▶ Instância: I_B
- ▶ Solução: S_B

Definição

Uma **redução** do problema A ao problema B é um par de sub-rotinas τ_I e τ_S tais que:

- ▶ τ_I transforma instância I_A de A em uma instância I_B de B
- ▶ τ_S transforma solução S_B de I_B em uma solução S_A de I_A .

Redução como um algoritmo

Como podemos resolver o problema A ?

1. Suponha que existe um algoritmo ALG_B para o problema B .
2. Podemos usar ALG_B como uma **caixa-preta**

```
REDUÇÃO( $I_A$ )  
1   $I_B \leftarrow \tau_I(I_A)$   
2   $S_B \leftarrow ALG_B(I_B)$   
3   $S_A \leftarrow \tau_S(I_A, S_B)$   
4  devolva  $S_A$ 
```

Em outras palavras:

- ▶ se sei resolver B , então também sei resolver A !
- ▶ A é mais “fácil” que B
- ▶ denotamos $A \preceq B$

Problema da Alocação de Centros (AC)

Entrada:

- ▶ um grafo bipartido conexo $G = ((X \cup Y), E)$ e
- ▶ uma função de pesos nas arestas $\omega : E \rightarrow \mathbb{R}_+$

Saída:

- ▶ alocar cada vértice v em X a um vértice $\phi[v]$ em Y tal que o peso $\omega(v, \phi[v])$ seja **mínimo**

Problema do Caminho Mínimo (CM)

Entrada:

- ▶ grafo direcionado acíclico $G = (V, E)$
- ▶ função de peso $\omega : E \rightarrow \mathbb{R}_+$ nas arestas
- ▶ origem s .

Saída:

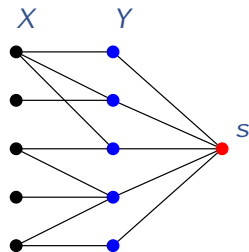
- ▶ vetor d com $d[v] = \text{dist}(s, v)$ para $v \in V$
- ▶ vetor π definindo uma **arvore de caminhos mínimos**

Reduzindo: transformação da entrada

Recebemos uma **entrada** do problema de origem AC:

$\tau_I(G, \omega)$

- 1 $G' \leftarrow G, \omega' \leftarrow \omega$
- 2 Adicione um novo vértice s a G' .
- 3 **para cada** $v \in Y$ faça
- 4 Adicione aresta (s, v) a G' .
- 5 $\omega'(s, v) \leftarrow 0$
- 6 **devolva** (G', ω', s)

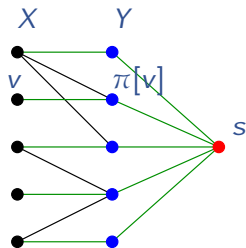


Tempo: $O(Y)$

Reduzindo: transformação da saída

Também recebemos uma **solução** do problema de destino CM:

$\tau_S(G, \omega, d, \pi)$
1 para cada $v \in X$ faça
2 $\phi[v] \leftarrow \pi[v]$
3 devolva ϕ



Tempo: $O(X)$

Reduzindo: $AC \preceq CM$

Seja ALG_{CM} um algoritmo para Caminho Mínimo.

- ▶ ALG_{CM} poderia ser DIJKSTRA, BELLMAN-FORD...
- ▶ Pode ser que **não conheçamos** um algoritmo para o problema de destino!

```
REDUÇÃO-AC-CM( $G, \omega$ )  
1  ( $G', \omega', s$ )  $\leftarrow \tau_I(G, \omega)$   
2  ( $d, \pi$ )  $\leftarrow ALG_{CM}(G', \omega', s)$   
3   $\phi \leftarrow \tau_S(G, \omega, d, \pi)$   
4  devolva  $\phi$ 
```

Tempo total: [tempo da redução] + [tempo de ALG_{CM}]

Tempo da redução

Quanto tempo gastamos só com a redução?

- ▶ não contamos o tempo do algoritmo para o problema B
- ▶ a **complexidade de uma redução** $f(n)$ é a soma dos tempos das transformações τ_I e τ_S
- ▶ escrevemos $A \leq_{f(n)} B$

```
REDUÇÃO-AC-CM( $G, \omega$ )  
1 ( $G', \omega', s$ )  $\leftarrow \tau_I(G, \omega)$   
2 ( $d, \pi$ )  $\leftarrow \text{ALG}_{\text{CM}}(G', \omega', s)$   
3  $\phi \leftarrow \tau_S(G, \omega, d, \pi)$   
4 devolva  $\phi$ 
```

Nesse caso: $AC \leq_{|X|+|Y|} CM$

Parênteses: reduções polinomiais

Queremos construir algoritmos rápidos. Mas o que é “rápido”?

- ▶ normalmente, dizemos que um algoritmo é rápido se ele executa em **tempo polinomial**
- ▶ daí, queremos reduções de tempo polinomial
- ▶ nesse caso, escrevemos $A \leq_{\text{poli}} B$

Qual a consequência de $A \leq_{\text{poli}} B$?

1. se B tem algoritmo de tempo polinomial, então A também
 2. se A **não** tem algoritmo de tempo polinomial, tampouco B
- ▶ Isso é útil para distinguir problemas fáceis de difíceis!
 - ▶ Mas é assunto para depois...

Exemplos de reduções

Exemplos de reduções

- ▶ Sistema Linear (LS) para Sistema Linear Simétrico (SLS)

Sistema Linear (LS)

Entrada:

- ▶ matriz M de dimensões $n \times n$ com determinante **não** nulo
- ▶ vetor b de dimensão n

Saída:

- ▶ vetor x de dimensão n que satisfaz o seguinte sistema linear:

$$Mx = b$$

Sistema Linear Simétrico (SLS)

Entrada:

- ▶ matriz M **simétrica** de dimensões $n \times n$ com determinante **não** nulo
- ▶ vetor b de dimensão n

Saída:

- ▶ vetor x de dimensão n que satisfaz o seguinte sistema linear:

$$Mx = b$$

SLS é um caso particular de LS

- ▶ então trivialmente $SLS \preceq LS$
- ▶ será que LS é estritamente mais difícil?

A resposta é **não!**

- ▶ Iremos reduzir LS para SLS.
- ▶ Isso é, $LS \preceq SLS$.

Um fato simples

Lema

Um vetor x é solução de $Mx = b$ se, e só se, x é solução de $M^T Mx = M^T b$.

Demonstração:

- (\Rightarrow)
- ▶ multiplicamos $Mx = b$ por M^T
 - ▶ obtemos $M^T Mx = M^T b$
- (\Leftarrow)
- ▶ M^T tem determinante não nulo
 - ▶ então M^T tem inversa Z
 - ▶ multiplicando $M^T Mx = M^T b$ por Z
 - ▶ obtemos $Mx = b$

Observe que $M' = M^T M$ é uma matriz simétrica!

REDUÇÃO-LS-SLS(M, b)

- 1 $M' \leftarrow M^T M$
- 2 $b' \leftarrow M^T b$
- 3 $x \leftarrow \text{ALG}_{\text{SLS}}(M', b')$
- 4 devolva x

Concluimos que de fato LS \preceq SLS.

Exemplos de reduções

- ▶ Casamento Cíclico de Strings para Casamento de Strings

Problema do casamento cíclico de strings (CSM)

Entrada:

- ▶ alfabeto Σ
- ▶ cadeia $A = a_0a_1 \dots a_{n-1}$ com n símbolos
- ▶ cadeia $B = b_0b_1 \dots b_{n-1}$ com n símbolos

Saída:

- ▶ decidir se B é um deslocamento cíclico de A
- ▶ se sim, então o número k de letras deslocadas

Exemplo:

- ▶ Entrada: $A = acgtact$ e $B = gtactac$
- ▶ Saída: TRUE, $k = 2$

Problema de destino

Problema do casamento de strings (SM)

Entrada:

- ▶ alfabeto Σ
- ▶ cadeia $A = a_0a_1 \dots a_{n-1}$ com n símbolos
- ▶ cadeia $B = b_0b_1 \dots b_{m-1}$ com m símbolos

Saída:

- ▶ decidir se B é **subcadeia** de A
- ▶ se sim, qual índice k da primeira ocorrência de B em A

Exemplo:

- ▶ Entrada: $A = acgttacccgtacccg$ e $B = tac$
- ▶ Saída: TRUE, $k = 4$

Observação: o problema SM pode ser resolvido em tempo $O(n + m)$ pelo algoritmo KMP de Knuth, Morris and Pratt (1977).

REDUÇÃO-CSM-SM(A, B, n)

- 1 $A' \leftarrow AA$ \triangleright concatena duas cópias de A
- 2 $B' \leftarrow B$
- 3 $n' \leftarrow 2n$
- 4 $m' \leftarrow n$
- 5 devolva $ALG_{SM}(A', n', B', m')$

- ▶ Tempo da redução: $O(n)$
- ▶ Correção: basta mostrar que se k é a solução de SM, então k também é solução de CSM.

Exemplo:

- ▶ $I_{CSM} = (acgtact, gtactac, 7)$
- ▶ $I_{SM} = (acgtactacgtact, 14, gtactac, 7)$
- ▶ $S_{SM} = S_{CSM} = (TRUE, 2)$

Exemplos de reduções



Existência de Triângulo para Multiplicação de Matrizes Quadradas

Problema de origem

Problema da existência de triângulo (PET)

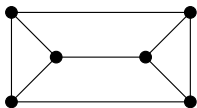
Entrada:

- ▶ grafo conexo $G = (V, E)$ sem laços com $n = |V|$ e $m = |E|$

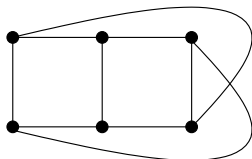
Saída:

- ▶ decidir se G contém um triângulo

Exemplo:



SIM



NÃO

Observações sobre o PET

Alguns algoritmos conhecidos:

- ▶ um algoritmo trivial de tempo $O(n^3)$:
 - ▶ verificar todas as triplas de vértices
- ▶ um algoritmo $O(mn)$ que
 - ▶ é muito bom se o grafo é **esparso**

Vamos supor que o grafo é denso:

- ▶ G será representado por uma matriz de adjacência A

$$a_{ij} = \begin{cases} 1 & \text{se } (i,j) \in E \\ 0 & \text{se } (i,j) \notin E \end{cases}$$

Lema

Seja $A^2 = A \times A$, ou seja, $a_{ij}^2 = \sum_{k=1}^n a_{ik} a_{kj}$.

Então $a_{ij}^2 > 0$ se e somente se existe caminho de tamanho 2 saindo de i e chegando em j .

Demonstração:

- (\Rightarrow)
- ▶ se $a_{ij}^2 > 0$, então algum termo $a_{ik} a_{kj}$ é positivo
 - ▶ segue que $a_{ik} = 1$ e $a_{kj} = 1$
 - ▶ ou seja, há arestas (i, k) e (k, j)
- (\Leftarrow)
- ▶ seja um caminho (i, k, j) de i até j
 - ▶ então $a_{ik} = 1$ e $a_{kj} = 1$
 - ▶ daí $a_{ik} a_{kj} > 0$ e, portanto, $a_{ij}^2 > 0$

Problema da Multiplicação de Matrizes Quadradas (MMQ)

Entrada:

- ▶ matriz quadrada A de ordem n .
- ▶ matriz quadrada B de ordem n .

Saída:

- ▶ produto $P = A \times B$.

Observações:

- ▶ há um algoritmo óbvio de complexidade $O(n^3)$
- ▶ MMQ pode ser resolvida mais rapidamente
 - ▶ em tempo $O(n^{2,807})$ pelo algoritmo de Strassen (1969)
 - ▶ em tempo $O(n^{2,3728639})$ pelo de François Le Gall (2014)

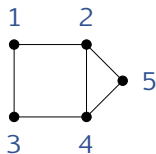
Observe que só existe triângulo com aresta (i, j) se

1. existir caminho de tamanho 2 de i a j
2. existir aresta (i, j)

```

REDUÇÃO-PET-MMQ( $A, n$ )
1   $A^2 \leftarrow \text{ALG}_{\text{MMQ}}(A, A, n)$ 
2  para  $i = 1$  até  $n$  faça
3      para  $j = 1$  até  $n$  faça
4          se  $a_{ij}^2 > 0$  e  $a_{ij} = 1$  então
5              devolva SIM
6  devolva NÃO
    
```

- ▶ Tempo da redução: $O(n^2)$
- ▶ Tempo total: $O(n^{2,3728639})$


 $A(G)$

	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	1	1
3	1	0	0	1	0
4	0	1	1	0	1
5	0	1	0	1	0

 $A^2 = A(G) \times A(G)$

	1	2	3	4	5
1	2	0	0	2	1
2	0	3	2	1	1
3	0	2	2	0	1
4	2	1	0	3	1
5	1	1	1	1	2

Exemplos de reduções



Multiplicação de Matrizes para Multiplicação de Matrizes Simétricas

Considerando o caso particular

Considere um caso particular de MMQ:

Multiplicação de Matrizes Simétricas (MMS)

Entrada:

- ▶ matriz **simétrica** quadrada A de ordem m .
- ▶ matriz **simétrica** quadrada B de ordem m .

Saída:

- ▶ produto $P = A \times B$.

Claro que $MMS \leq_{m^2} MMQ$.

- ▶ Portanto, MMQ é pelo menos tão difícil quanto MMS
- ▶ Será que MMS também é pelo menos tão difícil quanto MMQ?

MMQ \leq_n MMS

Reduzindo MMQ \leq_{n^2} MMS:

1. Considere uma instância de MMQ, $I_{MMQ} = (A, B, n)$.
2. Construa uma instância de MMS, $I_{MMS} = (A', B', 2n)$, em que

$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad \text{e} \quad B' = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}.$$

3. A solução de MMS é:

$$P' = A'B' = \begin{bmatrix} AB & 0 \\ 0 & A^TB^T \end{bmatrix}.$$

4. Devolva o primeiro bloco da matriz P' .

Tempo da redução: $O(n^2)$.

- ▶ construir I_{MMQ} leva tempo $O(n^2)$
- ▶ copiar o bloco e P' leva tempo $O(n^2)$

Interpretando os fatos

- ▶ suponha que exista um algoritmo para MMS de tempo $O(T(m))$ para algum polinômio $T(m)$
 - ▶ lembre que m é a ordem das matrizes A' e B'
- ▶ quão pequeno pode ser $T(m)$?
 - ▶ é claro que $T(m) = \Omega(m^2)$, pois é preciso ler a entrada
 - ▶ será que pode ser mais rápido que $o(m^{2,3728639})$?

Para responder isso, usamos a redução $\text{MMQ} \leq_{n^2} \text{MMS}$:

- ▶ ela implica em um algoritmo de tempo total $O(T(m) + n^2)$
- ▶ como $m = 2n$, tempo é $O(T(2n) + n^2) = O(T(n) + n^2)$
- ▶ como $T(n)$ domina n^2 , o tempo é simplesmente $O(T(n))$

Ou seja:

⇒ algoritmo $T(m)$ para MMS implica algoritmo $T(n)$ para MMQ!

Reduções para obtenção de cota inferior

Uma redução $A \leq_{f(n)} B$

REDUÇÃO(I_A)

- 1 $I_B \leftarrow \tau_I(I_A)$
- 2 $S_B \leftarrow \text{ALG}_B(I_B)$
- 3 $S_A \leftarrow \tau_S(I_A, S_B)$
- 4 devolva S_A

Suponha que

- ▶ A tem cota inferior $h(n)$
- ▶ ALG_B resolve B em tempo $g(n)$
- ▶ a redução gasta tempo $f(n) \leq \frac{h(n)}{2}$

REDUCAO é um algoritmo para A de tempo

$$f(n) + g(n) \geq h(n) \Rightarrow g(n) \geq h(n) - f(n) \geq h(n) - \frac{h(n)}{2} = \frac{h(n)}{2}$$

Conclusão: $g(n) \geq \Omega(h(n))$

Teorema

Considere dois problemas A e B e suponha que

1. $h(n)$ é cota inferior para A e
2. $A \leq_{f(n)} B$,
3. $f(n) = o(h(n))$.

Então $h(n)$ é cota inferior para B .

Observação:

- ▶ a cota inferior depende do **modelo de computação**
- ▶ supomos o mesmo modelo para ambos problemas

Reduções para obtenção de cota inferior

- ▶ Ordenação para Envoltória Convexa

Problema de origem

Problema da Ordenação (ORD)

Entrada:

- ▶ sequência $X = (x_1, x_2, \dots, x_n)$ de n elementos comparáveis

Saída:

- ▶ permutação X cujos elementos estejam ordenados

Observações:

- ▶ só podemos comparar dois elementos por uma sub-rotina caixa-preta de tempo constante (chamada **oráculo**)
- ▶ esse problema tem cota inferior $\Omega(n \lg n)$

Problema de destino

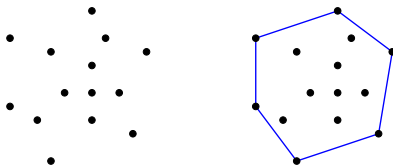
Problema da Envoltória Convexa (EC)

Entrada:

- ▶ conjunto $\{(x_1, y_1), \dots, (x_n, y_n)\}$ de n pontos no plano

Saída:

- ▶ menor polígono convexo que contém os n pontos



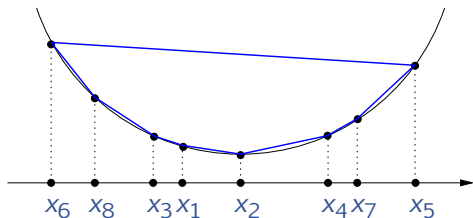
Observações:

- ▶ os vértices são representados em ordem anti-horária
- ▶ problema clássico de **Geometria Computacional**
- ▶ pode ser resolvido em tempo $O(n \lg n)$

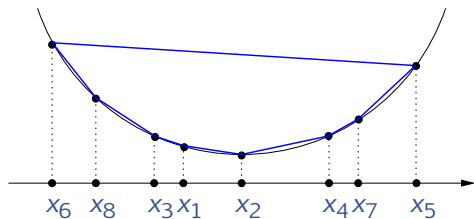
Reduzindo ORD \leq_n EC:

1. Considere uma instância de $I_{ORD} = (x_1, x_2, \dots, x_n)$.
2. Construa instância

$$I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \dots, (x_n, x_n^2)\}.$$



3. Resolva I_{EC} e obtenha solução S_{EC} , que é uma lista **cíclica** dos vértices do polígono.



4. Determine índice i de S_{EC} do ponto com menor abcissa.
5. Liste os todos os índices a partir de i .
- ▶ O tempo da redução é $O(n)$.
 - ▶ Portanto $\Omega(n \lg n)$ também é **cota inferior** para EC.

Reduções para obtenção de cota inferior

- ▶ Unicidade de Elementos para Ponto Mais Próximo em 2D

Problema da Unicidade de Elementos (UE)

Entrada:

- ▶ sequência $X = (x_1, x_2, \dots, x_n)$ de n elementos comparáveis

Saída:

- ▶ decidir se os elementos são **todos** distintos

Observações:

- ▶ só podemos comparar dois elementos por uma sub-rotina caixa-preta de tempo constante (chamada **oráculo**)
- ▶ esse problema tem cota inferior $\Omega(n \lg n)$
- ▶ o problema pode ser resolvido em tempo $O(n \lg n)$. (Como?)

Problema de destino

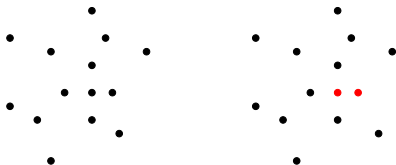
Problema do Par Mais Próximo (PMP)

Entrada:

- ▶ coleção $\{(x_1, y_1), \dots, (x_n, y_n)\}$ de n pontos no plano

Saída:

- ▶ par de pontos i e j que estejam a menor distância.



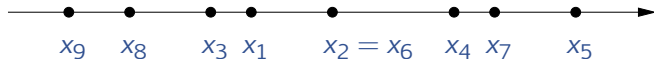
Observação:

- ▶ pode ser resolvido em tempo $O(n \lg n)$

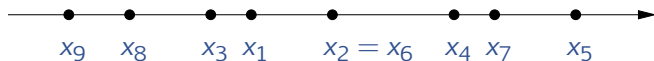
Reduzindo UE \leq_n PMP:

1. Considere instância $I_{UE} = (x_1, x_2, \dots, x_n)$.
2. Construa instância

$$I_{PMP} = \{(x_1, 0), (x_2, 0), \dots, (x_n, 0)\}.$$



3. Resolva I_{PMP} e obtenha par de pontos $(x_i, 0), (x_j, 0)$.
4. Calcule a **distância** d entre os pontos:
 - (a) Se $d = 0$, então devolva NÃO.
 - (b) Se $d > 0$, devolva SIM.



- ▶ O tempo da redução é $O(n)$.
- ▶ Portanto $\Omega(n \lg n)$ também é **cota inferior** para PMP.

Reduções para obtenção de cota inferior

- ▶ 3-Soma para Colinearidade

Problema de origem

Problema da 3-Soma (3SUM)

Entrada:

- ▶ sequência $X = (x_1, x_2, \dots, x_n)$ de n reais

Saída:

- ▶ determinar se existem índices distintos i, j e k tais que:

$$x_i + x_j + x_k = 0$$

Exemplo:

- ▶ instância $X = (\underline{4}, -6, \underline{1}, 8, 7, \underline{-5})$
- ▶ solução $i = 1, j = 3$ e $k = 6$

Observações:

- ▶ pode ser resolvido em $O(n^2)$ (Como?)
- ▶ acreditava-se que $\Omega(n^2)$ era **cota inferior**
- ▶ pode ser resolvido em $o(n^2)$ (Grønlund e Pettie, 2014)
- ▶ ainda se acredita que não dá pra fazer melhor que $n^{2-\Omega(1)}$

Problema de destino

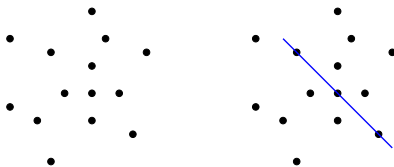
Problema da Colinearidade Não Horizontal (COL)

Entrada:

- ▶ conjunto $\{(x_1, y_1), \dots, (x_n, y_n)\}$ de n pontos no plano

Saída:

- ▶ determinar se três pontos estão em alguma **reta não horizontal**



Observações:

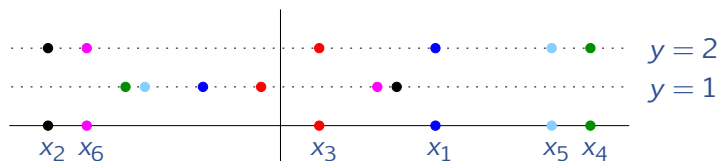
- ▶ pode ser resolvido em tempo $O(n^2)$
- ▶ acredita-se que $\Omega(n^2)$ é **cota inferior**

3SUM \leq_n COL

Reduzindo 3SUM \leq_n COL:

1. Considere instância $I_{3SUM} = (x_1, x_2, \dots, x_n)$.
2. Construa instância

$$I_{COL} = \{(x_i, 0), (-x_i/2, 1), (x_i, 2) : i = 1, 2, \dots, n\}.$$



Exemplo: $X = (4, -6, 1, 8, 7, -5)$

3. Resolva I_{COL} e obtenha S_{COL} .
4. Se a resposta S_{COL} for SIM, responda SIM;
mas se a resposta S_{COL} for NÃO, responda NÃO.

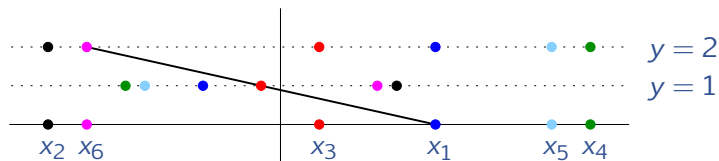
3SUM \leq_n COL (cont)

- ▶ A solução de I_{COL} (se houver) é uma tripla de pontos colineares. Claramente, cada um desses pontos deve estar em um dos eixos horizontais. Ou seja, tem a forma:

$$(x_i, 0), (-x_j/2, 1), (x_k, 2).$$

Portanto, $x_i + x_j + x_k = 0$.

- ▶ De modo análogo, se $x_i + x_j + x_k = 0$, então os pontos citados são colineares.



Exemplo: $X = (4, -6, 1, 8, 7, -5)$

- ▶ É claro que $\Omega(n)$ é uma cota inferior para 3SUM.
- ▶ E se houver cota inferior $\Omega(h(n))$ maior para 3SUM?
 - ▶ a redução gasta tempo $f(n) = O(n)$
 - ▶ nesse caso, $f(n) = o(h(n))$
 - ▶ então $\Omega(h(n))$ seria cota inferior para COL
- ▶ Mas só conhecemos a cota trivial $\Omega(n)$ para 3SUM.

Exercício

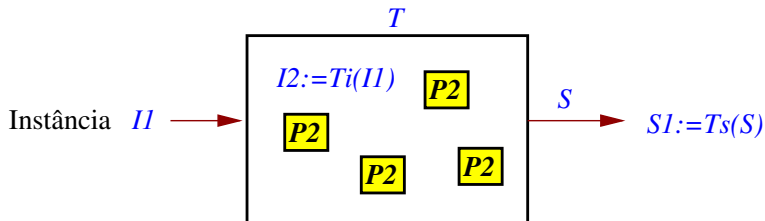
O problema **3SUMplus** consiste em: dados uma sequência $X = (x_1, x_2, \dots, x_n)$ de reais e um valor real b , determinar se existem três índices distintos i, j e k tais que $x_i + x_j + x_k = b$.

1. Mostre que $3SUM \leq_n 3SUMplus$.
2. Mostre que $3SUMplus \leq_n 3SUM$.
3. Suponha que o Professor Sabit Udo descobriu uma **cota inferior** de $\Omega(n^{1,9})$ para **3SUMplus**. Nesse caso, quais das afirmações abaixo são verdadeiras?
 - (i) Não existe algoritmo $O(n^{1,5})$ para **3SUMplus**.
 - (ii) Não existe algoritmo $O(n^{1,5})$ para **3SUM**.
 - (iii) Existe um algoritmo $O(n^{1,9})$ para **3SUMplus**.
 - (iv) Existe um algoritmo $O(n^{1,9})$ para **3SUM**.

Outros exemplos de reduções

Redução de Turing

Existem reduções de P_1 para P_2 , fazendo várias aplicações de P_2 .



Exemplo de Redução de Turing

Problema de Multiplicação de Inteiros

Dados inteiros a e b , calcular $a \cdot b$.

Problema do Quadrado

Dados inteiro x , calcular x^2 .

Proposição: Se pudermos fazer número constante de somas, subtrações e divisão por 2 então podemos reduzir Multiplicação de Inteiros para Quadrado.

Prova: Note que

$$a \cdot b = \frac{(a + b)^2 - a^2 - b^2}{2}$$

Sistema de Representantes Distintos

Dada coleção de conjuntos S_1, \dots, S_k temos que $R = \{r_1, \dots, r_k\}$ é um Sistema de Representantes Distintos (SRD) se $r_i \in S_i$ para todo $i = 1, \dots, k$.

Problema do Sistema de Representantes Distintos

Dada coleção de conjuntos S_1, \dots, S_k , encontrar um SRD.

Sistema de Representantes Distintos

Considere os seguintes conjuntos:

Ecológicos: Ana, Alberto

Ruralistas: João, Alberto

Feministas: Ana, Maria

Então, {Ana, João, Maria} formam um SRD.

Não há SRD para a coleção abaixo:

▶ $S_1 = \{1, 2\}$

▶ $S_2 = \{3, 4\}$

▶ $S_3 = \{3, 4\}$

▶ $S_4 = \{1, 2, 4\}$

▶ $S_5 = \{2, 4\}$

Teorema de Hall

S_1, \dots, S_k tem um SRD se e somente se

$$|\{S_{i_1} \cup S_{i_2} \cup \dots \cup S_{i_m}\}| \geq m$$

para $\{i_1, \dots, i_m\} \subseteq \{1, 2, 3, \dots, k\}$ e $1 \leq m \leq k$.

Isso é, qualquer subcoleção de m conjuntos tem pelo menos m itens distintos.

- ▶ Podemos usar o Teorema de Hall, de maneira direta, para testar todas possíveis subcoleções de conjuntos.
- ▶ Mas o número de possíveis subcoleções é $O(2^k)$.

Sistema de Representantes Distintos

Problema do Emparelhamento Máximo - EM

Dado grafo bipartido $G = (X, Y, E)$, onde X e Y são conjuntos de vértices e E é o conjunto de arestas, encontrar um emparelhamento (conjunto de arestas sem extremos em comum) de cardinalidade máxima.

Proposição: Problema do SRD \leq Problema do EM.

Dada coleção S_1, \dots, S_k , instância de um SRD, sobre conjunto A ,

defina grafo $G = (X, Y, E)$ onde

- ▶ $X = S_1 \cup S_2 \cup \dots \cup S_k$
- ▶ $Y = \{1, 2, \dots, k\}$
- ▶ $E = \{(a, j) \mid a \in S_j \text{ e } 1 \leq j \leq k\}$.

Note que o SRD tem solução sse G tem emparelhamento de tamanho k . □

As seguintes operações são possíveis em strings:

- ▶ Inserção de um caracter
- ▶ Remoção de um caracter
- ▶ Troca de um caracter por outro

Problema de Edição de String - ES

Dadas strings A e B , transformar A em B com o menor número de operações.

Edição de String

Se $A = babb$ e $B = bbc$, transformamos A em B com duas operações:

$babb$



Remover a

bbb



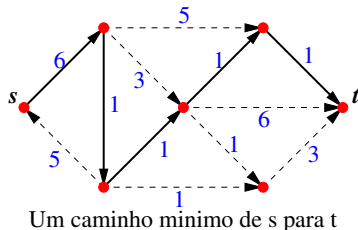
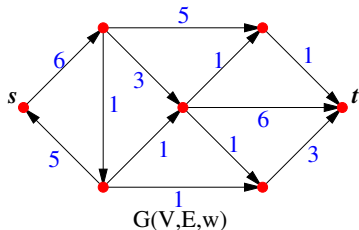
Trocar último b por c

bbc

Exercício: O Problema de Edição de String pode ser resolvido por programação dinâmica.

Problema do Caminho Mínimo em Grafo Orientado

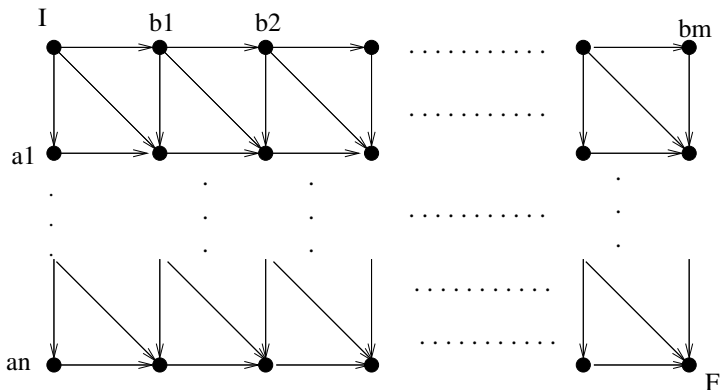
Dado grafo orientado $G(V,E)$, onde cada aresta ij possui custo $c_{ij} > 0$, e vértices s e t , encontrar um caminho de custo total mínimo de s a t em G .



Edição de String

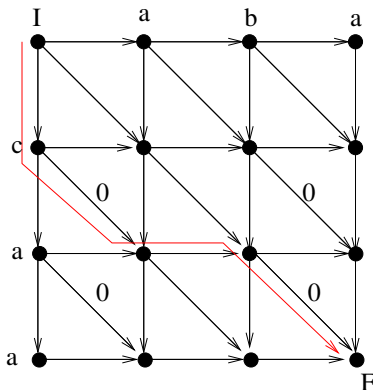
Proposição: Problema de Edição de String \leq Problema do Caminho Mínimo em grafos orientados.

Prova: Dadas strings $A = a_1 a_2 \dots a_n$ e $B = b_1 b_2 \dots b_m$ (instância do Problema ES) construímos um grafo da seguinte forma:



- ▶ Arestas horizontais correspondem a inserção de um caractere e possuem custo 1.
- ▶ Arestas verticais correspondem a remoção de um caractere e possuem custo 1.
- ▶ Arestas diagonais correspondem a uma troca e tem custo 1 caso os caracteres sejam diferentes, e 0 caso sejam iguais.
- ▶ O problema é encontrar um caminho mínimo do vértice I ao F .

Dados strings $A = caa$ e $B = aba$ construa grafo G :



custo de edição é $2 = 1 + 0 + 1 + 0$

Temos que mostrar que um caminho mínimo em G de I até F corresponde a uma edição mínima.

- ▶ Dado uma edição mínima, a partir do caracter vazio temos 4 opções (inserir, remover, trocar/match) que correspondem as arestas no grafo.
- ▶ Uma edição de strings corresponde a um caminho e este por sua vez corresponde a uma edição.
- ▶ Portanto um caminho mínimo será uma edição mínima.

Código de Huffman: Compressão de dados

Relembrando:

- ▶ Codificação para compressão de dados
- ▶ Cada caractere deve ter representação binária única
- ▶ Há codificações com número de bits fixo e variável
- ▶ Codificação Prefixa: Codificação de um caracter não é prefixo de outro

- ▶ Exemplo de codificação prefixa:

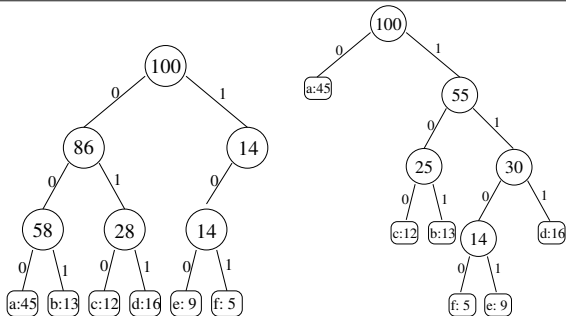
<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
0	101	100	111	1101	1100

- ▶ Codificação prefixa pode ser representada por uma árvore de Huffman
- ▶ Caracteres são representados nas folhas

Exemplo

Considere 100.000 caracteres com frequências e codificações:

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
Frequência da letra	45	13	12	16	9	5
Codificação usando 3 bits	000	001	010	011	100	101
Codificação de tamanho variável	0	101	100	111	1101	110



Codificação Fixa: $100.000 \times 3 = 300.000$ bits

Codificação Variável: $\sum_{s \in A} \text{freq}(s) \cdot 100.000 \cdot |\text{codigo}(s)| = 224.000$
bits

Código de Huffman

Dado texto com caracteres C_1, C_2, \dots, C_n ,

- ▶ cada caractere C_i tem frequência f_i ,
- ▶ obter código S_i para C_i , onde $s_i = |S_i|$ e

- ▶ minimizamos $c = \sum_{i=1}^n s_i f_i$

- ▶ Os códigos devem satisfazer a restrição de prefixo.
- ▶ Construa árvore de Huffman correspondente a codificação

Teorema: Existe um algoritmo que calcula a árvore ótima para um conjunto de caracteres C_1, C_2, \dots, C_n com frequências f_1, f_2, \dots, f_n em tempo $O(n \log n)$.

Código de Huffman

Faremos uma redução do problema de ordenação de inteiros para o código de Huffman.

Proposicao: Sejam C_1, C_2, \dots, C_n caracteres com frequências f_1, f_2, \dots, f_n onde

$$\sum_{i=1}^{j-1} f_i < f_j \text{ para cada } j = 2, \dots, n.$$

Então a árvore de Huffman ótima terá C_n como folha no nível 1, C_{n-1} como folha no nível 2, e assim sucessivamente com C_1 e C_2 no nível $n - 1$.

Código de Huffman

Prova: Seja T^* uma árvore ótima para uma instância como a enunciada. Suponha por absurdo que em T^* o caractere C_n não esteja no nível 1. Adicione uma nova raiz em T^* retirando C_n do seu lugar e colocando este como filho da nova raiz. T^* também é adicionada a esta nova raiz.

O custo da nova árvore diminuiu de pelo menos f_n e aumentou de

$$\sum_{i=1}^{n-1} f_i.$$

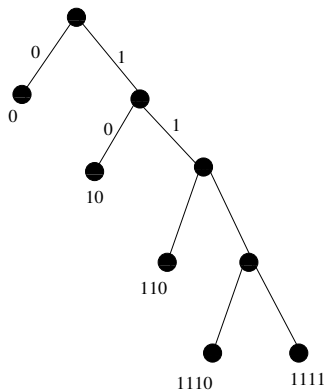
Portanto T^* não era ótima o que é um absurdo. Podemos repetir o mesmo raciocínio para os demais caracteres

C_{n-1}, C_{n-2}, \dots



Código de Huffman

- ▶ Instâncias baseadas no último teorema terão uma árvore de Huffman com a seguinte forma:



Tentativa:

- ▶ Considere uma instância para o problema de ordenação com números x_1, x_2, \dots, x_n com uma diferença grande de valores entre eles:

$$\sum_{i=1}^{j-1} x_i < x_j \text{ para } j = 2, \dots, n$$

- ▶ Crie uma instância para o problema de códigos de Huffman com caracteres C_i e frequência x_i para $i = 1, \dots, n$.
- ▶ Baseado no teorema, a solução será uma árvore onde podemos facilmente ordenar os números x_1, \dots, x_n .

O que está faltando?

Caso geral:

- ▶ Ao fazer a redução $P_1 \leq P_2$ devemos considerar uma instância genérica de P_1 .
- ▶ Podemos salvar a nossa redução considerando uma entrada x_1, \dots, x_n qualquer para o problema de ordenação.
- ▶ Neste caso usamos frequências $2^{x_1}, 2^{x_2}, \dots, 2^{x_n}$.
- ▶ Note que para qualquer valor j temos

$$\sum_{i=1}^{j-1} 2^i < 2^j$$

- ▶ Portanto a instância criada gera uma árvore como a desejada.