Projeto e Análise de Algoritmos II (MC558)
Reduções entre problemas

Prof. Dr. Ruben Interian

Resumo

Resumo

Revisão do conteúdo

- As reduções podem ser usadas para encontrar **cotas superiores** (e, como veremos hoje, **cotas inferiores**) de problemas.
- As reduções permitem comparar a **complexidade de problemas**.
  (Pergunta: A **não é "mais difícil"** do que B?)

Objetivo

Hoje veremos mais exemplos de reduções, em particular mostrando como podemos **encontrar cotas inferiores** de problemas.

Resumo

Reduções entre problemas

- **Algoritmo ótimo**: Um algoritmo A é ótimo para um problema P quando sua complexidade coincide com uma cota inferior de P, ou seja, **todo algoritmo** B que resolve P possui no mínimo a complexidade de A: $f_B(n) = \Omega(f_A(n))$.

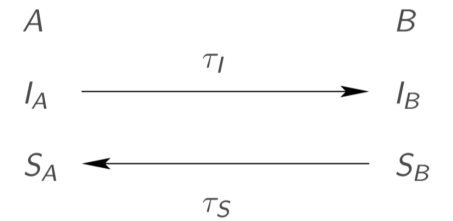
Reduções entre problemas

- **Algoritmo ótimo**: Um algoritmo A é ótimo para um problema P quando sua complexidade coincide com uma cota inferior de P, ou seja, **todo algoritmo** B que resolve P possui no mínimo a complexidade de A: $f_B(n) = \Omega(f_A(n))$.
- **Complexidade de um problema**: complexidade de um algoritmo ótimo que resolve o problema.

Reduções entre problemas

- **Algoritmo ótimo**: Um algoritmo A é ótimo para um problema P quando sua complexidade coincide com uma cota inferior de P, ou seja, **todo algoritmo** B que resolve P possui no mínimo a complexidade de A: $f_B(n) = \Omega(f_A(n))$.
- **Complexidade de um problema**: complexidade de um algoritmo ótimo que resolve o problema.
- **Analisar um problema** A: deterinar a complexidade de A, ou seja, analisar o tempo de execução (de pior caso) de um algoritmo ótimo para A.

Reduções entre problemas



$\text{Alg}_A\ (I_A)$

1: $I_B \leftarrow \tau_I(I_A)$
2: $S_B \leftarrow \text{Alg}_B(I_B)$
3: $S_A \leftarrow \tau_S(S_B)$
devolva S_A

Se a redução é "rápida" (e.g., linear, e Alg_B precisa ser no mínimo linear):

A **não é "mais difícil"** do que B: $f_A(n) = O(f_B(n))$

=

B é **pelo menos "tão difícil"** quanto A: $f_B(n) = \Omega(f_A(n))$.

Reduções entre problemas

Detalhe importante:

Para mostrar que existe uma redução de A para B, não precisamos mostrar que existe um algoritmo para resolver o problema B. Apenas precisamos mostrar quais são os algoritmos τ_I e τ_S!

Reduções entre problemas

Detalhe importante:

Para mostrar que existe uma redução de A para B, não precisamos mostrar que existe um algoritmo para resolver o problema B. Apenas precisamos mostrar quais são os algoritmos τ_I e τ_S!

Reduzir o problema A para o problema B não significa resolver A!

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Problema da Multiplicação de Matrizes Quadradas (MMQ)

Entrada: duas matrizes quadradas A e B de ordem n.

Objetivo: calcular o produto $A \times B$.

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Problema da Multiplicação de Matrizes Simétricas (MMS)

Entrada: duas matrizes simétricas quadradas A e B de ordem n.

Objetivo: calcular o produto $A \times B$.

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Problema da Multiplicação de Matrizes Simétricas (MMS)

Entrada: duas matrizes simétricas quadradas A e B de ordem n.

Objetivo: calcular o produto $A \times B$.

Observações:

- **MMS** é um caso particular de **MMQ**: a redução **MMS** $\propto$ **MMQ** é imediata. Portanto, **MMQ** é pelo menos tão difícil quanto **MMS**.

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Problema da Multiplicação de Matrizes Simétricas (MMS)

Entrada: duas matrizes simétricas quadradas A e B de ordem n.

Objetivo: calcular o produto $A \times B$.

Observações:

- **MMS** é um caso particular de **MMQ**: a redução **MMS** $\propto$ **MMQ** é imediata. Portanto, **MMQ** é pelo menos tão difícil quanto **MMS**.
- Muito menos óbvio é: Será que **MMS** é **pelo menos tão difícil quanto MMQ**?

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Redução: **MMQ** $\propto$ **MMS**

- Instância de **MMQ**: $I_{MMQ} = (A, B, n)$.

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Redução: **MMQ** $\propto$ **MMS**

- Instância de **MMQ**: $I_{MMQ} = (A, B, n)$.
- τ_I constrói a instância de **MMS**: $I_{MMS} = (A', B', 2n)$ onde

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

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Redução: **MMQ** $\propto$ **MMS**

- Instância de **MMQ**: $I_{MMQ} = (A, B, n)$.
- τ_I constrói a instância de **MMS**: $I_{MMS} = (A', B', 2n)$ onde

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

Veja que A' e B' são simétricas, e τ_I custa $O(n^2)$.

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Redução: **MMQ** $\propto$ **MMS**

- Instância de **MMQ**: $I_{MMQ} = (A, B, n)$.
- τ_I constrói a instância de **MMS**: $I_{MMS} = (A', B', 2n)$ onde

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

  Veja que A' e B' são simétricas, e τ_I custa $O(n^2)$.

- A solução de **MMS** é:

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

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

O algoritmo τ_S pode ser implementado da seguinte forma:

$\tau_S\ (P', 2n)$

1: $P = 0_{n,n}$
2: $\triangleright$ copia uma parte da saída de **MMS** (correspondente à AB) para P:
3: **para cada** $i \leftarrow 1, \ldots, n$ **faça**
4: **para cada** $j \leftarrow 1, \ldots, n$ **faça**
5: $p_{ij} \leftarrow p'_{ij}$
devolva P

O algoritmo τ_S é **linear** no tamanho da entrada ($O(n^2)$).

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Existe uma redução **linear** no tamanho da entrada, de **MMQ** para **MMS**.

Consequências:

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Existe uma redução **linear** no tamanho da entrada, de **MMQ** para **MMS**.

Consequências:

- Já sabiamos que **MMS** $\propto_{n^2}$ **MMQ**, agora sabemos que **MMQ** $\propto_{n^2}$ **MMS**.

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Existe uma redução **linear** no tamanho da entrada, de **MMQ** para **MMS**.

Consequências:

- Já sabiamos que **MMS** $\propto_{n^2}$ **MMQ**, agora sabemos que **MMQ** $\propto_{n^2}$ **MMS**.
- Ambos os problemas são, de certa forma, equivalentes: um algoritmo para **MMQ** implica em um algoritmo da mesma complexidade para **MMS**, e vice-versa.

Reduções entre problemas: Exemplo MMQ $\propto$ MMS

Existe uma redução **linear** no tamanho da entrada, de **MMQ** para **MMS**.

Consequências:

- Já sabiamos que **MMS** $\propto_{n^2}$ **MMQ**, agora sabemos que **MMQ** $\propto_{n^2}$ **MMS**.
- Ambos os problemas são, de certa forma, equivalentes: um algoritmo para **MMQ** implica em um algoritmo da mesma complexidade para **MMS**, e vice-versa.
- As cotas superiores de **MMQ** e **MMS** são iguais.

Reduções entre problemas: cotas inferiores

Como podemos **encontrar cotas inferiores** de problemas usando reduções?

Reduções entre problemas: cotas inferiores

$\text{Alg}_A\ (I_A)$

1: $I_B \leftarrow \tau_I(I_A)$
2: $S_B \leftarrow \text{Alg}_B(I_B)$
3: $S_A \leftarrow \tau_S(S_B)$
devolva S_A

Vamos supor que:

- O problema A tem cota inferior $f(n)$.

Reduções entre problemas: cotas inferiores

$\text{Alg}_A\ (I_A)$

1: $I_B \leftarrow \tau_I(I_A)$
2: $S_B \leftarrow \text{Alg}_B(I_B)$
3: $S_A \leftarrow \tau_S(S_B)$
devolva S_A

Vamos supor que:

- O problema A tem cota inferior $f(n)$.
- Alg_B resolve B em tempo $g(n)$.

Reduções entre problemas: cotas inferiores

$\text{Alg}_A\ (I_A)$

1: $I_B \leftarrow \tau_I(I_A)$
2: $S_B \leftarrow \text{Alg}_B(I_B)$
3: $S_A \leftarrow \tau_S(S_B)$
devolva S_A

Vamos supor que:

- O problema A tem cota inferior $f(n)$.
- Alg_B resolve B em tempo $g(n)$.
- A redução é $o(f(n))$.

Reduções entre problemas: cotas inferiores

$\text{Alg}_A\ (I_A)$

1: $I_B \leftarrow \tau_I(I_A)$
2: $S_B \leftarrow \text{Alg}_B(I_B)$
3: $S_A \leftarrow \tau_S(S_B)$
devolva S_A

Vamos supor que:

- O problema A tem cota inferior $f(n)$.
- Alg_B resolve B em tempo $g(n)$.
- A redução é $o(f(n))$.

Alg_A é um algoritmo para A cuja complexidade é:

$$g(n) + o(f(n)) \geq f(n) \Rightarrow$$

Reduções entre problemas: cotas inferiores

$\text{Alg}_A\ (I_A)$

1: $I_B \leftarrow \tau_I(I_A)$
2: $S_B \leftarrow \text{Alg}_B(I_B)$
3: $S_A \leftarrow \tau_S(S_B)$
devolva S_A

Vamos supor que:

- O problema A tem cota inferior $f(n)$.
- Alg_B resolve B em tempo $g(n)$.
- A redução é $o(f(n))$.

Alg_A é um algoritmo para A cuja complexidade é:

$$g(n) + o(f(n)) \geq f(n) \Rightarrow$$

$$g(n) \geq f(n) - o(f(n))$$

Reduções entre problemas: cotas inferiores

$\text{Alg}_A\ (I_A)$

1: $I_B \leftarrow \tau_I(I_A)$
2: $S_B \leftarrow \text{Alg}_B(I_B)$
3: $S_A \leftarrow \tau_S(S_B)$
devolva S_A

Vamos supor que:

- O problema A tem cota inferior $f(n)$.
- Alg_B resolve B em tempo $g(n)$.
- A redução é $o(f(n))$.

Alg_A é um algoritmo para A cuja complexidade é:

$$g(n) + o(f(n)) \geq f(n) \Rightarrow$$

$$g(n) \geq f(n) - o(f(n))$$

Conclusão: $g(n)$ é $\Omega(f(n))$. Temos cota inferior para B!

Reduções entre problemas: cotas inferiores

Considere dois problemas A e B. Suponha que:

1. $f(n)$ é cota inferior para A.
2. $A \propto B$ em tempo $o(f(n))$.

Então $f(n)$ é cota inferior para B.

Reduções entre problemas: cotas inferiores

Considere dois problemas A e B. Suponha que:

1. $f(n)$ é cota inferior para A.
2. $A \propto B$ em tempo $o(f(n))$.

Então $f(n)$ é cota inferior para B.

A complexidade e a cota inferior estão definidas a partir de um modelo de computação. Vamos supor que o modelo de computação para A e B é o mesmo: **operações usadas** (permitidas) **para resolver A e B precisam ser as mesmas.**

Reduções entre problemas: Exemplo ORD $\propto$ EC

Problema da Ordenação (ORD)

Entrada: sequência $X = (x_1, x_2, \ldots, x_n)$ de n elementos comparáveis.

Objetivo: encontrar uma permutação de X cujos elementos estejam ordenados.

Observações:

- Podemos comparar dois elementos usando uma instrução de tempo constante.
- Esse problema tem cota inferior $\Omega(n \log n)$.

Reduções entre problemas: Exemplo ORD $\propto$ EC

Problema da Envoltória Convexa (EC)

Entrada: conjunto $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ de n pontos no plano.

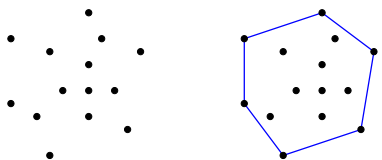
Objetivo: encontrar o menor polígono convexo que contém os n pontos.

Reduções entre problemas: Exemplo ORD $\propto$ EC

Problema da Envoltória Convexa (EC)

Entrada: conjunto $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ de n pontos no plano.

Objetivo: encontrar o menor polígono convexo que contém os n pontos.



Observações:

- A saída é uma sequência anti-horária de vértices do polígono.
- Problema clássico de Geometria Computacional.

Reduções entre problemas: Exemplo ORD $\propto$ EC

Redução: **ORD** $\propto$ **EC**

- Instância de **ORD**: $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
- τ_I constrói a instância de **EC**:

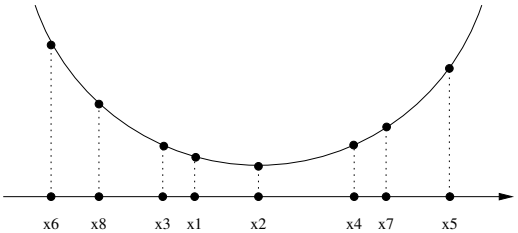
$$I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}.$$

Reduções entre problemas: Exemplo ORD $\propto$ EC

Redução: **ORD** $\propto$ **EC**

- Instância de **ORD**: $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
- τ_I constrói a instância de **EC**:

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



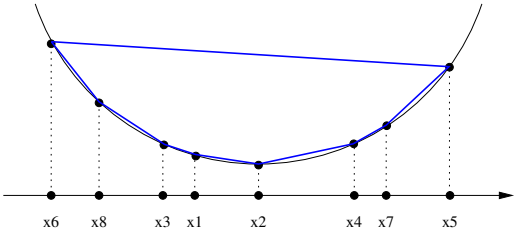
Claramente τ_I custa um tempo $O(n)$.

Reduções entre problemas: Exemplo ORD $\propto$ EC

- S_{EC}, a solução de I_{EC}, é uma ordem cíclica de pontos.

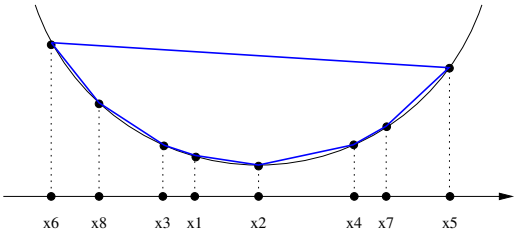
Reduções entre problemas: Exemplo ORD $\propto$ EC

- S_{EC}, a solução de I_{EC}, é uma ordem cíclica de pontos.



Reduções entre problemas: Exemplo ORD $\propto$ EC

- S_{EC}, a solução de I_{EC}, é uma ordem cíclica de pontos.



- τ_S:
  (I) determinar o ponto que tem **menor abcissa**, e
  (II) listar as abcissas dos pontos na mesma ordem cíclica anti-horária de S_{EC}.
- Temos uma ordenação de $I_{ORD} = (x_1, x_2, \ldots, x_n)$.
- Claramente τ_S custa um tempo $O(n)$.

Reduções entre problemas: Exemplo ORD $\propto$ EC

Segue que $\Omega(n \log n)$ é uma cota inferior para **EC**.

Reduções entre problemas: Exemplo ORD $\propto$ EC

Segue que $\Omega(n \log n)$ é uma cota inferior para **EC**.

***EC** pode ser resolvido em $O(n \log n)$ por um algoritmo que usa a estratégia de divisão-e-conquista. Como $\Omega(n \log n)$ é uma cota inferior, esse algoritmo é **ótimo**.

Reduções entre problemas: Exemplo UE $\propto$ PMP

Problema da Unicidade de Elementos (UE)

Entrada: sequência de elementos comparáveis de comprimento n.

$$X = (x_1, x_2, \ldots, x_n).$$

Objetivo: decidir se todos os elementos de X são **distintos**.

Observações:

- No **modelo de árvore de decisão**, **UE** tem cota inferior $\Omega(n \log n)$. A prova é semelhante à prova da cota inferior da **ORD**.

Reduções entre problemas: Exemplo UE $\propto$ PMP

Problema da Unicidade de Elementos (UE)

Entrada: sequência de elementos comparáveis de comprimento n.

$$X = (x_1, x_2, \ldots, x_n).$$

Objetivo: decidir se todos os elementos de X são **distintos**.

Observações:

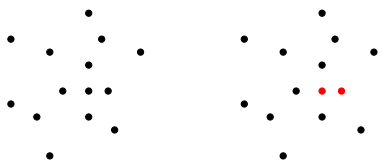
- No **modelo de árvore de decisão**, **UE** tem cota inferior $\Omega(n \log n)$. A prova é semelhante à prova da cota inferior da **ORD**.
- O problema pode ser resolvido em tempo $O(n \log n)$. **(Como?)**

Reduções entre problemas: Exemplo UE $\propto$ PMP

Problema Problema do Par Mais Próximo (PMP)

Entrada: coleção $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ de n pontos no plano.

Objetivo: encontrar **dois pontos** que estejam a menor distância.



Observações:

- Problema clássico de Geometria Computacional.

Reduções entre problemas: Exemplo UE $\propto$ PMP

Redução: **UE** $\propto$ **PMP**

- Instância de **UE**: $I_{UE} = (x_1, x_2, \ldots, x_n)$.
- τ_I constrói a instância de **PMP**:

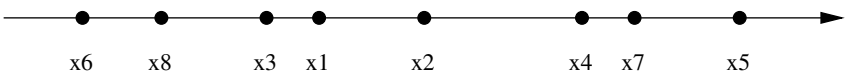
$$I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}.$$

Reduções entre problemas: Exemplo UE $\propto$ PMP

Redução: **UE** $\propto$ **PMP**

- Instância de **UE**: $I_{UE} = (x_1, x_2, \ldots, x_n)$.
- τ_I constrói a instância de **PMP**:

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



- Claramente, τ_I custa $O(n)$.

Reduções entre problemas: Exemplo UE $\propto$ PMP

- A solução de I_{PMP} é um par de pontos $(x_i, 0), (x_j, 0)$.

Reduções entre problemas: Exemplo UE $\propto$ PMP

- A solução de I_{PMP} é um par de pontos $(x_i, 0), (x_j, 0)$.
- τ_S apenas verifica se a distância entre os dois pontos é **zero**:
  (I) Se $d = 0$, então a resposta de I_{UE} é NÃO.
  (II) Se $d > 0$, então a resposta de I_{UE} é SIM.
- Claramente, τ_S custa $O(1)$.

Reduções entre problemas: Exemplo UE $\propto$ PMP

- A solução de I_{PMP} é um par de pontos $(x_i, 0), (x_j, 0)$.
- τ_S apenas verifica se a distância entre os dois pontos é **zero**:
  (I) Se $d = 0$, então a resposta de I_{UE} é NÃO.
  (II) Se $d > 0$, então a resposta de I_{UE} é SIM.
- Claramente, τ_S custa $O(1)$.

Segue que $\Omega(n \log n)$ é uma cota inferior para **PMP**.

Reduções entre problemas: Reduções de Turing

Reduções de Turing

Reduções entre problemas: Reduções de Turing

Em uma **redução de Turing**, é permitido usar várias vezes o hipotético algoritmo Alg_B.

Reduções entre problemas: Reduções de Turing

Em uma **redução de Turing**, é permitido usar várias vezes o hipotético algoritmo Alg_B.

- Ou seja, podemos reduzir A para B executando várias vezes Alg_B.
- Um exemplo clássico é o problema de determinar caminhos mínimos entre todos os pares de vértices de um grafo direcionado (problema A):
  - A pode ser resolvido com $|V|$ execuções do algoritmo que resolve o problema de determinar caminhos mínimos a partir de um vértice origem s (problema B).

Reduções entre problemas: Reduções de Turing

Em uma **redução de Turing**, é permitido usar várias vezes o hipotético algoritmo Alg_B.

- Ou seja, podemos reduzir A para B executando várias vezes Alg_B.
- Um exemplo clássico é o problema de determinar caminhos mínimos entre todos os pares de vértices de um grafo direcionado (problema A):
  - A pode ser resolvido com $|V|$ execuções do algoritmo que resolve o problema de determinar caminhos mínimos a partir de um vértice origem s (problema B).
- Existem, ainda, as reduções de Karp, que iremos usar para estudar classes de complexidade de problemas. Elas exigem que Alg_B seja executado apenas uma vez, e que a complexidade da redução seja polinomial (entre outras restrições).

Reduções entre problemas

Perguntas

Sejam P_1 e P_2 dois problemas tais que $P_1 \propto_n P_2$ e suponha que P_1 tem cota inferior $\Omega(n \log n)$, onde n é um parâmetro que mede o tamanho da entrada do problema P_1. Quais das seguintes afirmações são verdadeiras? Justifique as suas respostas.

1. Todo algoritmo que resolve P_1 pode ser usado para resolver P_2.

Reduções entre problemas

Perguntas

Sejam P_1 e P_2 dois problemas tais que $P_1 \propto_n P_2$ e suponha que P_1 tem cota inferior $\Omega(n \log n)$, onde n é um parâmetro que mede o tamanho da entrada do problema P_1. Quais das seguintes afirmações são verdadeiras? Justifique as suas respostas.

1. Todo algoritmo que resolve P_1 pode ser usado para resolver P_2.
2. Todo algoritmo que resolve P_2 pode ser usado para resolver P_1.

Reduções entre problemas

Perguntas

Sejam P_1 e P_2 dois problemas tais que $P_1 \propto_n P_2$ e suponha que P_1 tem cota inferior $\Omega(n \log n)$, onde n é um parâmetro que mede o tamanho da entrada do problema P_1. Quais das seguintes afirmações são verdadeiras? Justifique as suas respostas.

1. Todo algoritmo que resolve P_1 pode ser usado para resolver P_2.
2. Todo algoritmo que resolve P_2 pode ser usado para resolver P_1.
3. $\Omega(n \log n)$ é cota inferior para P_2.

Reduções entre problemas

Perguntas

Sejam P_1 e P_2 dois problemas tais que $P_1 \propto_n P_2$ e suponha que P_1 tem cota inferior $\Omega(n \log n)$, onde n é um parâmetro que mede o tamanho da entrada do problema P_1. Quais das seguintes afirmações são verdadeiras? Justifique as suas respostas.

1. Todo algoritmo que resolve P_1 pode ser usado para resolver P_2.
2. Todo algoritmo que resolve P_2 pode ser usado para resolver P_1.
3. $\Omega(n \log n)$ é cota inferior para P_2.
4. O problema P_2 pode ser resolvido em tempo $O(n \log n)$.

Resumo

Síntese

- Podemos **encontrar cotas inferiores** de problemas usando reduções.
- Vimos algumas reduções envolvendo problemas de Geometria Computacional.

Material bibliográfico e exercícios

U. Manber. Introduction to Algorithms. – **Cap. 10**

Exercícios: ver exercícios no final do Capítulo 10.

Dúvidas

Dúvidas?