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 comparar a **complexidade de problemas** (A **não é "mais difícil"** do que B?), assim como encontrar novos algoritmos para problemas.
- Podemos **encontrar cotas inferiores** de problemas usando reduções.

Objetivo

- Hoje veremos mais exemplos de reduções, em particular algumas reduções envolvendo grafos, e introduziremos um novo tipo de reduções de problemas.

Resumo

Reduções entre problemas: Exemplo ES $\propto$ PCM

Problema de Edição de String (ES)

Entrada: duas strings, A e B.

Objetivo: encontrar a menor sequência de operações para transformar A em B.

Reduções entre problemas: Exemplo ES $\propto$ PCM

Problema de Edição de String (ES)

Entrada: duas strings, A e B.

Objetivo: encontrar a menor sequência de operações para transformar A em B.

Operações sobre strings:

- inserção de um caractere,
- remoção de um caractere,
- troca de um caractere por outro.

Reduções entre problemas: Exemplo ES $\propto$ PCM

Exemplo: Considere as strings $A = babb$ e $B = bbc$.

- podemos transformar A em B:

$$babb$$

Reduções entre problemas: Exemplo ES $\propto$ PCM

Exemplo: Considere as strings $A = babb$ e $B = bbc$.

- podemos transformar A em B:

$$babb \Downarrow \text{ remova } a \quad bbb$$

Reduções entre problemas: Exemplo ES $\propto$ PCM

Exemplo: Considere as strings $A = babb$ e $B = bbc$.

- podemos transformar A em B:

$$babb$$
$$\Downarrow \quad \text{remova } a$$
$$bbb$$
$$\Downarrow \quad \text{troque o } 3^{ro}\ b \text{ pelo } c$$
$$bbc$$

Reduções entre problemas: Exemplo ES $\propto$ PCM

Exemplo: Considere as strings $A = babb$ e $B = bbc$.

- podemos transformar A em B:

$$babb$$
$$\Downarrow \quad \text{remova } a$$
$$bbb$$
$$\Downarrow \quad \text{troque o } 3^{ro}\ b \text{ pelo } c$$
$$bbc$$

- Realizamos duas operações ao todo.

Reduções entre problemas: Exemplo ES $\propto$ PCM

Problema do Caminho Mínimo (PCM)

Entrada: grafo direcionado $G(V, E)$, custos $w_{ij} \geq 0$ para cada arco $(i, j) \in E$, vértices s e t.

Objetivo: encontrar o caminho mínimo de s a t em G.

Reduções entre problemas: Exemplo ES $\propto$ PCM

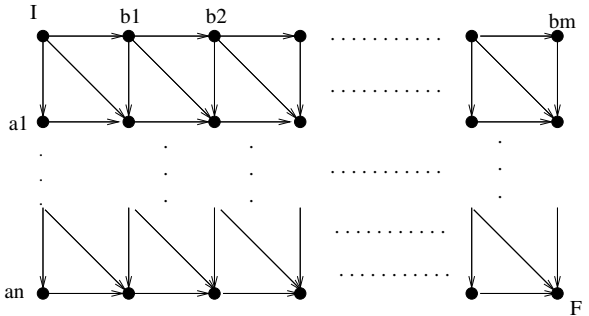
Redução: **ES** $\propto$ **PCM**

- Instância de **ES**: duas strings $A = a_1 a_2 \ldots a_n$ e $B = b_1 b_2 \ldots b_m$.

Reduções entre problemas: Exemplo ES $\propto$ PCM

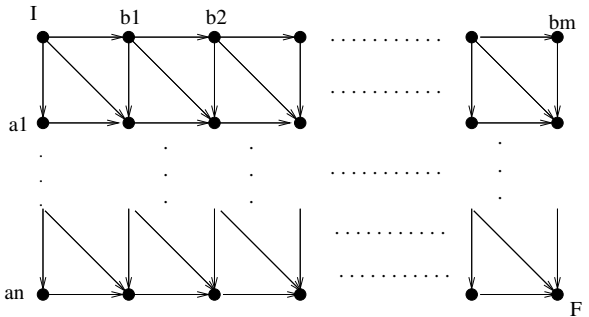
Redução: **ES** $\propto$ **PCM**

- Instância de **ES**: duas strings $A = a_1 a_2 \ldots a_n$ e $B = b_1 b_2 \ldots b_m$.
- τ_I constrói a instância de **PCM**, um grafo direcionado G com $|V| = (n+1)(m+1)$:



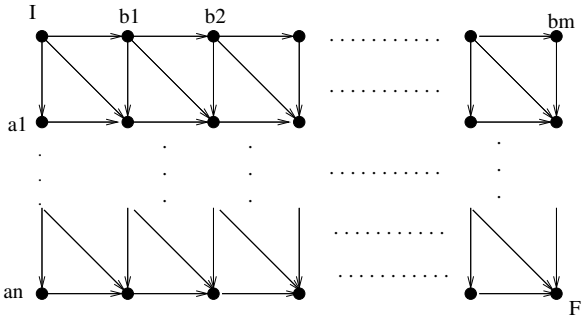
Reduções entre problemas: Exemplo ES $\propto$ PCM

- No grafo G:
  - Arestas horizontais correspondem a inserção de um caractere (custo 1).
  - Arestas verticais correspondem a remoção de um caractere (custo 1).
  - Arestas diagonais correspondem a uma troca (custo 1 se os caracteres são diferentes, 0 se os caracteres são iguais).



Reduções entre problemas: Exemplo ES $\propto$ PCM

- Alg_B encontra um caminho mínimo de I até F no grafo obtido.
- τ_S devolve a sequência de operações usada.
- Exemplo: $A = abcy$ e $B = yabc$.



Reduções entre problemas: Exemplo ES $\propto$ PCM

- O algoritmo trivial para **Edição de String** é exponencial, com complexidade 3^k onde $k = \max(n, m)$.

Reduções entre problemas: Exemplo ES $\propto$ PCM

- O algoritmo trivial para **Edição de String** é exponencial, com complexidade 3^k onde $k = \max(n, m)$.
- Qual é a complexidade do algoritmo que resolve **Edição de String** usando essa redução para o **Problema do Caminho Mínimo**?

Reduções entre problemas: Exemplo ES $\propto$ PCM

- O algoritmo trivial para **Edição de String** é exponencial, com complexidade 3^k onde $k = \max(n, m)$.
- Qual é a complexidade do algoritmo que resolve **Edição de String** usando essa redução para o **Problema do Caminho Mínimo**?

  $O(V + E)$ usando **DagShortestPaths**

Reduções entre problemas: Exemplo ES $\propto$ PCM

- O algoritmo trivial para **Edição de String** é exponencial, com complexidade 3^k onde $k = \max(n, m)$.
- Qual é a complexidade do algoritmo que resolve **Edição de String** usando essa redução para o **Problema do Caminho Mínimo**?

  $O(V + E)$ usando **DagShortestPaths**

  $O(nm + nm)$

Reduções entre problemas: Exemplo ES $\propto$ PCM

- O algoritmo trivial para **Edição de String** é exponencial, com complexidade 3^k onde $k = \max(n, m)$.
- Qual é a complexidade do algoritmo que resolve **Edição de String** usando essa redução para o **Problema do Caminho Mínimo**?

  $O(V + E)$ usando **DagShortestPaths**
  $O(nm + nm)$
  $O(nm)$.

Reduções entre problemas: Exemplo CLI $\propto$ IND

Problema do clique (CLI)

Entrada: grafo não-direcionado $G = (V, E)$.

Objetivo: encontrar a maior clique em G, ou seja, o maior conjunto de vértices mutuamente adjacentes.

Reduções entre problemas: Exemplo CLI $\propto$ IND

Problema do clique (CLI)

Entrada: grafo não-direcionado $G = (V, E)$.

Objetivo: encontrar a maior clique em G, ou seja, o maior conjunto de vértices mutuamente adjacentes.

Observações:

- Redes sociais: clique é um grupo de pessoas em que todas se conhecem.

Reduções entre problemas: Exemplo CLI $\propto$ IND

Problema do clique (CLI)

Entrada: grafo não-direcionado $G = (V, E)$.

Objetivo: encontrar a maior clique em G, ou seja, o maior conjunto de vértices mutuamente adjacentes.

Observações:

- Redes sociais: clique é um grupo de pessoas em que todas se conhecem.
- Redes funcionais de genes: clique é um grupo de genes cujas funções (funcionalidades) estão relacionadas.

Reduções entre problemas: Exemplo CLI $\propto$ IND

Problema do conjunto independente (IND)

Entrada: grafo não-direcionado $G = (V, E)$.

Objetivo: encontrar o maior conjunto de vértices $S \subseteq V$ tal que não existem dois vértices adjacentes contidos em S.

Reduções entre problemas: Exemplo CLI $\propto$ IND

Redução: **CLI** $\propto$ **IND**

- Instância de **CLI**: grafo $G = (V, E)$.

Reduções entre problemas: Exemplo CLI $\propto$ IND

Redução: **CLI** $\propto$ **IND**

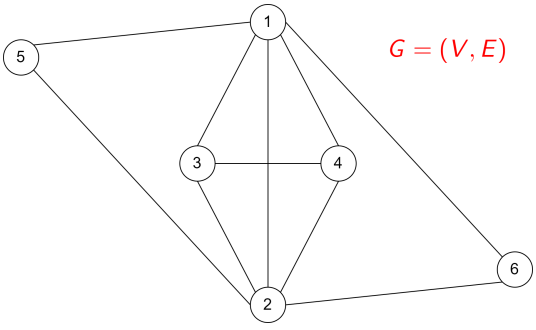
- Instância de **CLI**: grafo $G = (V, E)$.
- τ_I constrói a instância de **IND**: o grafo complementar $\overline{G} = (V, \overline{E})$.
  - **Lembrando**: as arestas em $\overline{G}$ são as "inversas": $e \in \overline{E}$ se $e \notin E$; $e \notin \overline{E}$ se $e \in E$.

Reduções entre problemas: Exemplo CLI $\propto$ IND

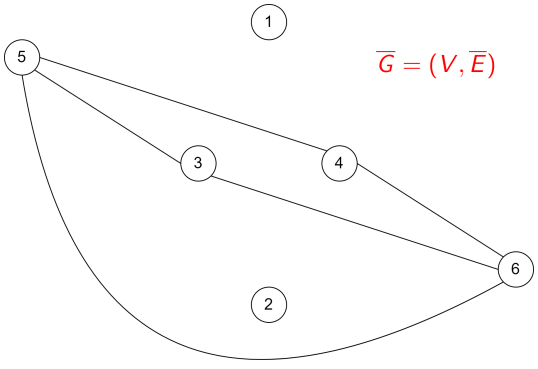
Redução: **CLI** $\propto$ **IND**

- Instância de **CLI**: grafo $G = (V, E)$.
- τ_I constrói a instância de **IND**: o grafo complementar $\overline{G} = (V, \overline{E})$.
  - **Lembrando**: as arestas em $\overline{G}$ são as "inversas": $e \in \overline{E}$ se $e \notin E$; $e \notin \overline{E}$ se $e \in E$.
- τ_S devolve o mesmo conjunto de vértices S devolvido por algum Alg_{IND}.

Reduções entre problemas: Exemplo CLI $\propto$ IND



$G = (V, E)$



$\overline{G} = (V, \overline{E})$

Reduções entre problemas: Exemplo CLI $\propto$ IND

Observações importantes:

- Veja que não apenas **CLI** $\propto$ **IND**, mas também **IND** $\propto$ **CLI**.

Reduções entre problemas: Exemplo CLI $\propto$ IND

Observações importantes:

- Veja que não apenas **CLI** $\propto$ **IND**, mas também **IND** $\propto$ **CLI**.
- A redução **não é linear**. (Por quê?)

Reduções entre problemas: Exemplo CLI $\propto$ IND

Observações importantes:

- Veja que não apenas **CLI** $\propto$ **IND**, mas também **IND** $\propto$ **CLI**.
- A redução **não é linear**. (Por quê?)
- Intuitivamente, quando faria sentido usar essa redução para resolver **CLI**?

Reduções entre problemas: Exemplo CLI $\propto$ IND

Observações importantes:

- Veja que não apenas **CLI** $\propto$ **IND**, mas também **IND** $\propto$ **CLI**.
- A redução **não é linear**. (Por quê?)
- Intuitivamente, quando faria sentido usar essa redução para resolver **CLI**?
- O problema do conjunto independente (**IND**), em geral, não possui algoritmos eficientes conhecidos. Existem diversos algoritmos para esse problema, mas nenhum deles é assintoticamente polinomial.

Reduções entre problemas: Exemplo CLI $\propto$ IND

Observações importantes:

- Veja que não apenas **CLI** $\propto$ **IND**, mas também **IND** $\propto$ **CLI**.
- A redução **não é linear**. (Por quê?)
- Intuitivamente, quando faria sentido usar essa redução para resolver **CLI**?
- O problema do conjunto independente (**IND**), em geral, não possui algoritmos eficientes conhecidos. Existem diversos algoritmos para esse problema, mas nenhum deles é assintoticamente polinomial.
- Veja que a nossa redução é "rápida" (**polinomial**), comparada aos algoritmos para o problema, que não são rápidos (consomem um **tempo super-polinomial**).

Reduções entre problemas: Exemplo CLI $\propto$ IND

Observações importantes:

- Veja que não apenas **CLI** $\propto$ **IND**, mas também **IND** $\propto$ **CLI**.
- A redução **não é linear**. (Por quê?)
- Intuitivamente, quando faria sentido usar essa redução para resolver **CLI**?
- O problema do conjunto independente (**IND**), em geral, não possui algoritmos eficientes conhecidos. Existem diversos algoritmos para esse problema, mas nenhum deles é assintoticamente polinomial.
- Veja que a nossa redução é "rápida" (**polinomial**), comparada aos algoritmos para o problema, que não são rápidos (consomem um **tempo super-polinomial**).
- Isso significa que, **se houvesse um algoritmo eficiente** (polinomial) para Conjunto Independente, **haveria também um algoritmo eficiente** para Clique.

Reduções entre problemas: Exemplo CLI $\propto$ IND

Observações importantes:

- Veja que não apenas **CLI** $\propto$ **IND**, mas também **IND** $\propto$ **CLI**.
- A redução **não é linear**. (Por quê?)
- Intuitivamente, quando faria sentido usar essa redução para resolver **CLI**?
- O problema do conjunto independente (**IND**), em geral, não possui algoritmos eficientes conhecidos. Existem diversos algoritmos para esse problema, mas nenhum deles é assintoticamente polinomial.
- Veja que a nossa redução é "rápida" (**polinomial**), comparada aos algoritmos para o problema, que não são rápidos (consomem um **tempo super-polinomial**).
- Isso significa que, **se houvesse um algoritmo eficiente** (polinomial) para Conjunto Independente, **haveria também um algoritmo eficiente** para Clique.
- **E vice-versa!**

Reduções entre problemas: Exemplo CLI $\propto$ IND

Exemplo de **questão** (e exemplo de **resposta correta** à questão)

Questão. Encontre uma redução de complexidade polinomial de **CLI** para **IND**.

Reduções entre problemas: Exemplo CLI $\propto$ IND

Exemplo de **questão** (e exemplo de **resposta correta** à questão)

Questão. Encontre uma redução de complexidade polinomial de **CLI** para **IND**.

Resposta. Precisamos encontrar uma redução, ou seja, dois algoritmos, τ_I e τ_S.
O primeiro algoritmo, τ_I, transforma uma instância de **CLI**, um grafo $G = (V, E)$, em uma instância do **IND**, construindo o grafo complementar $\overline{G} = (V, \overline{E})$:

Algorithm $\tau_I\ (G)$

1: $\overline{E} \leftarrow \emptyset$
2: **para cada** $v \in V$ **faça**
3: **para cada** $w \in V \setminus (Adj[v] \cup \{v\})$ **faça**
4: $\overline{E}.add(v, w)$
devolva $\overline{G} = (V, \overline{E})$

Reduções entre problemas: Exemplo CLI $\propto$ IND

O algoritmo τ_I é polinomial, pois percorre cada vértice (linha 2), criando arestas para vértices não incidentes a ele (linhas 3-5), no pior caso em tempo $O(n \cdot n) = O(n^2)$.

Reduções entre problemas: Exemplo CLI $\propto$ IND

O algoritmo τ_I é polinomial, pois percorre cada vértice (linha 2), criando arestas para vértices não incidentes a ele (linhas 3-5), no pior caso em tempo $O(n \cdot n) = O(n^2)$.

O segundo algoritmo, τ_S, simplesmente retorna a saída do **IND**, um conjunto independente S de vértices de $\overline{G}$, que também é uma clique em G:

Algorithm τ_S (S)

1: **devolva** S

Reduções entre problemas: Exemplo CLI $\propto$ IND

O algoritmo τ_I é polinomial, pois percorre cada vértice (linha 2), criando arestas para vértices não incidentes a ele (linhas 3-5), no pior caso em tempo $O(n \cdot n) = O(n^2)$.

O segundo algoritmo, τ_S, simplesmente retorna a saída do **IND**, um conjunto independente S de vértices de $\overline{G}$, que também é uma clique em G:

Algorithm τ_S (S)

1: **devolva** S

A redução é **válida**. Se encontrarmos um conjunto independente S em $\overline{G}$, S será uma clique em G, pois $e \notin \overline{E}$ se $e \in E$, e haverá arestas entre todos os pares de vértices de S em G. A clique S é a maior possível em G, pois se houvesse outra clique S' ainda maior, **IND** teria retornado S', um conjunto independente maior em $\overline{G}$.

Resumo

Reduções entre problemas: Novo problema

Novo problema:

$$
\begin{aligned}
\min \quad & c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \\
\text{sujeito a} \quad & a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n \leq b_1 \\
& a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n \leq b_2 \\
& \ldots \\
& a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n \leq b_m \\
& \\
& x_j \geq 0 \text{ para } j = 1, \ldots, n
\end{aligned}
$$

Reduções entre problemas: Novo problema

Novo problema:

$$\min \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeito a } \sum_{j=1}^{n} a_{ij} x_j \leq b_i \text{ para } i = 1, \ldots, m$$

$$x_j \geq 0 \text{ para } j = 1, \ldots, n$$

Reduções entre problemas: Novo problema

Novo problema:

$$\begin{aligned} &\min c^T x \\ &\text{sujeito a } Ax \leq b,\ x \geq 0 \end{aligned}$$

Reduções entre problemas: Novo problema

Novo problema:

$$\begin{array}{c} \min\ c^T x \\ \text{sujeito a } Ax \leq b,\ x \geq 0 \end{array}$$

$x \in \mathbb{R}^n$ ou $x \in \mathbb{Z}^n$

Reduções entre problemas: Novo problema

Novo problema:

$$\begin{array}{c} \min\ c^T x \\ \text{sujeito a } Ax \leq b,\ x \geq 0 \end{array}$$

$x \in \mathbb{R}^n$ ou $x \in \mathbb{Z}^n$

O que é a entrada deste problema? O que é a saída?

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Problema do conjunto dominante (MDS)

Entrada: grafo não-direcionado $G = (V, E)$.

Objetivo: encontrar o menor conjunto dominante em G, ou seja, o menor conjunto de vértices S tal que cada vértice $v \notin S$ é adjacente a pelo menos um vértice em S.

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Problema do conjunto dominante (MDS)

Entrada: grafo não-direcionado $G = (V, E)$.

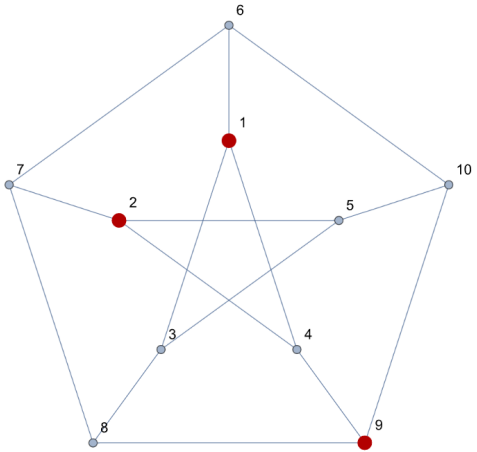
Objetivo: encontrar o menor conjunto dominante em G, ou seja, o menor conjunto de vértices S tal que cada vértice $v \notin S$ é adjacente a pelo menos um vértice em S.

Observações:

- Em uma rede social, um conjunto dominante é um grupo de usuários capazes de atingir a (se comunicar com) todos os indivíduos da rede.
- Usado em estratégias de marketing viral por empresas.

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Exemplo de instância do problema do conjunto dominante



Reduções entre problemas: Exemplo MDS $\propto$ ILP

Problema de Programação Linear Inteira (ILP)

Entrada: matriz $A_{m \times n}$, vetor $b \in \mathbb{R}^m$, vetor $c \in \mathbb{R}^n$.

Objetivo: encontrar valores para as variáveis inteiras não negativas $x_1, x_2, \ldots, x_n = x$ que minimizem a soma $\sum_{j=1}^{n} c_j x_j$, sujeito às restrições $Ax \leq b$.

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Problema de Programação Linear Inteira (ILP)

Entrada: matriz $A_{m \times n}$, vetor $b \in \mathbb{R}^m$, vetor $c \in \mathbb{R}^n$.

Objetivo: encontrar valores para as variáveis inteiras não negativas $x_1, x_2, \ldots, x_n = x$ que minimizem a soma $\sum_{j=1}^{n} c_j x_j$, sujeito às restrições $Ax \leq b$.

Observação: matematicamente, o problema é formulado assim:

$$\min c^T x$$
$$\text{sujeito a } Ax \leq b, x \geq 0$$

$$\min \sum_{j=1}^{n} c_j x_j$$
$$\text{sujeito a } \sum_{j=1}^{n} a_{ij} x_j \leq b_i \text{ para } i = 1, \ldots, m$$
$$x_j \geq 0 \text{ para } j = 1, \ldots, n$$

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Redução: **MDS** $\propto$ **ILP**

- Instância de **MDS**: grafo $G = (V, E)$.

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Redução: **MDS** $\propto$ **ILP**

- Instância de **MDS**: grafo $G = (V, E)$.
- τ_I constrói a instância de **ILP**:
  - Para cada vértice $j \in V$ teremos uma variável binária x_j, indicando se $j \in$ **MDS**;

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Redução: **MDS** $\propto$ **ILP**

- Instância de **MDS**: grafo $G = (V, E)$.
- τ_I constrói a instância de **ILP**:
  - Para cada vértice $j \in V$ teremos uma variável binária x_j, indicando se $j \in$ **MDS**;
  - Para cada vértice $j \in V$ cria uma restrição $\sum_{i \in Adj[j] \cup \{j\}} x_i \geq 1$.

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Redução: **MDS** $\propto$ **ILP**

- Instância de **MDS**: grafo $G = (V, E)$.
- τ_I constrói a instância de **ILP**:
  - Para cada vértice $j \in V$ teremos uma variável binária x_j, indicando se $j \in$ **MDS**;
  - Para cada vértice $j \in V$ cria uma restrição $\sum_{i \in Adj[j] \cup \{j\}} x_i \geq 1$.
  - Essa restrição significa que ou j, ou um dos seus adjacentes, precisa estar no **MDS**.

Reduções entre problemas: Exemplo MDS $\propto$ ILP

- Formulação completa do problema **ILP**:

$$\min \sum_{j=1}^{n} x_j$$

$$\text{sujeito a} \sum_{i \in Adj[j] \cup \{j\}} x_i \geq 1 \text{ para } j = 1, \ldots, n$$

$$x_j \in \{0, 1\} \text{ para } j = 1, \ldots, n$$

Reduções entre problemas: Exemplo MDS $\propto$ ILP

- Formulação completa do problema **ILP**:

$$\min \sum_{j=1}^{n} x_j$$

$$\text{sujeito a} \sum_{i \in Adj[j] \cup \{j\}} x_i \geq 1 \text{ para } j = 1, \ldots, n$$

$$x_j \in \{0, 1\} \text{ para } j = 1, \ldots, n$$

- τ_S devolve o conjunto de vértices S correspondente às variáveis com valor 1 (variáveis "escolhidas") na saída do Alg_{ILP}.

Reduções entre problemas: Exemplo MDS $\propto$ ILP

- Formulação completa do problema **ILP**:

$$\min \sum_{j=1}^{n} x_j$$

$$\text{sujeito a} \sum_{i \in Adj[j] \cup \{j\}} x_i \geq 1 \text{ para } j = 1, \ldots, n$$

$$x_j \in \{0, 1\} \text{ para } j = 1, \ldots, n$$

- τ_S devolve o conjunto de vértices S correspondente às variáveis com valor 1 (variáveis "escolhidas") na saída do Alg_{ILP}.
  - **Observe**: Alg_{ILP} apenas devolve o valor de cada variável!

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Observações importantes:

- A redução apresentada "transforma" um problema em grafos em um problema matemático, com um conjunto de variáveis, uma função objetivo linear, e um conjunto de desigualdades lineares. Muitos problemas em grafos podem ser transformados nesse tipo de problema, chamado de **Programação Linear**.

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Observações importantes:

- A redução apresentada "transforma" um problema em grafos em um problema matemático, com um conjunto de variáveis, uma função objetivo linear, e um conjunto de desigualdades lineares. Muitos problemas em grafos podem ser transformados nesse tipo de problema, chamado de **Programação Linear**.
- Se as variáveis são reais, o problema é chamado simplesmente Problema de Programação Linear, **PL**.

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Observações importantes:

- A redução apresentada "transforma" um problema em grafos em um problema matemático, com um conjunto de variáveis, uma função objetivo linear, e um conjunto de desigualdades lineares. Muitos problemas em grafos podem ser transformados nesse tipo de problema, chamado de **Programação Linear**.
- Se as variáveis são reais, o problema é chamado simplesmente Problema de Programação Linear, **PL**.
- Se as variáveis são inteiras ou binárias, o problema é chamado Problema de Programação Linear Inteira, **PLI**.

Reduções entre problemas: Exemplo MDS $\propto$ ILP

Observações importantes:

- A redução apresentada "transforma" um problema em grafos em um problema matemático, com um conjunto de variáveis, uma função objetivo linear, e um conjunto de desigualdades lineares. Muitos problemas em grafos podem ser transformados nesse tipo de problema, chamado de **Programação Linear**.
- Se as variáveis são reais, o problema é chamado simplesmente Problema de Programação Linear, **PL**.
- Se as variáveis são inteiras ou binárias, o problema é chamado Problema de Programação Linear Inteira, **PLI**.
- A complexidade computacional de um Problema de Programação Linear e de um Problema de Programação Linear Inteira é bem diferente: é mais fácil resolver **LP**. Pode imaginar o porquê?

Reduções entre problemas

- Durante a 2da Guerra Mundial, devido a escassez de recursos em diversos setores, os primeiros a financiar a Programação Linear foram os militares (em particular, a força aérea norte-americana), que apoiou projetos para resolver modelos de PL.
  Após o fim da 2da Guerra Mundial, a indústria começou a aplicar técnicas de PL, sendo hoje indispensável para diversas empresas (e.g., **Amazon**, **Mercado Livre**).

Reduções entre problemas

- Durante a 2da Guerra Mundial, devido a escassez de recursos em diversos setores, os primeiros a financiar a Programação Linear foram os militares (em particular, a força aérea norte-americana), que apoiou projetos para resolver modelos de PL.
  Após o fim da 2da Guerra Mundial, a indústria começou a aplicar técnicas de PL, sendo hoje indispensável para diversas empresas (e.g., **Amazon**, **Mercado Livre**).
  PL é aplicada em diversas áreas:
  - **Produção**: otimizar a alocação de recursos, minimizar custos, maximizar o retorno;

Reduções entre problemas

- Durante a 2da Guerra Mundial, devido a escassez de recursos em diversos setores, os primeiros a financiar a Programação Linear foram os militares (em particular, a força aérea norte-americana), que apoiou projetos para resolver modelos de PL.
  Após o fim da 2da Guerra Mundial, a indústria começou a aplicar técnicas de PL, sendo hoje indispensável para diversas empresas (e.g., **Amazon**, **Mercado Livre**).
  PL é aplicada em diversas áreas:
  - **Produção**: otimizar a alocação de recursos, minimizar custos, maximizar o retorno;
  - **Logística**: otimizar sistemas de transporte, distribuição de mercadorias;

Reduções entre problemas

- Durante a 2da Guerra Mundial, devido a escassez de recursos em diversos setores, os primeiros a financiar a Programação Linear foram os militares (em particular, a força aérea norte-americana), que apoiou projetos para resolver modelos de PL.
  Após o fim da 2da Guerra Mundial, a indústria começou a aplicar técnicas de PL, sendo hoje indispensável para diversas empresas (e.g., **Amazon**, **Mercado Livre**).
  PL é aplicada em diversas áreas:
  - **Produção**: otimizar a alocação de recursos, minimizar custos, maximizar o retorno;
  - **Logística**: otimizar sistemas de transporte, distribuição de mercadorias;
  - **Finanças**: alocação de investimentos em carteiras de ações, minimizar os riscos.

Reduções entre problemas

- Durante a 2da Guerra Mundial, devido a escassez de recursos em diversos setores, os primeiros a financiar a Programação Linear foram os militares (em particular, a força aérea norte-americana), que apoiou projetos para resolver modelos de PL. Após o fim da 2da Guerra Mundial, a indústria começou a aplicar técnicas de PL, sendo hoje indispensável para diversas empresas (e.g., **Amazon**, **Mercado Livre**). PL é aplicada em diversas áreas:
  - **Produção**: otimizar a alocação de recursos, minimizar custos, maximizar o retorno;
  - **Logística**: otimizar sistemas de transporte, distribuição de mercadorias;
  - **Finanças**: alocação de investimentos em carteiras de ações, minimizar os riscos.
- O termo "programação" dentro de "Programação Linear", inicialmente, não teve nada a ver com a computação. Foi empregado porque os militares se referem ao planejamento de tarefas usando o termo "programar atividades" (*program*, ou *schedule*).

Resumo

Síntese

- Vimos mais exemplos de reduções, em particular algumas reduções envolvendo grafos, e vimos um exemplo de um novo tipo de redução de problemas: formulação do problema como PLI.

Material bibliográfico e exercícios

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

Exercícios: ver exercícios no final do Capítulo 10.

Dúvidas

Dúvidas?