

Projeto e Análise de Algoritmos II (MC558)

Reduções entre problemas

Prof. Dr. Ruben Interian

Resumo

- 1 Revisão do conteúdo e objetivo
- 2 Reduções entre problemas
- 3 Síntese

Objetivo

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

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 $ES \propto PCM$

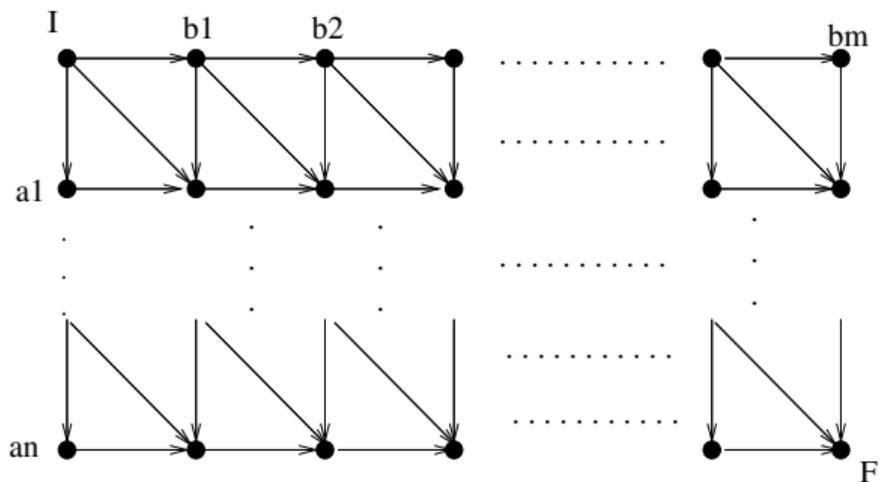
Redução: $ES \propto PCM$

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

Reduções entre problemas: Exemplo $ES \propto PCM$

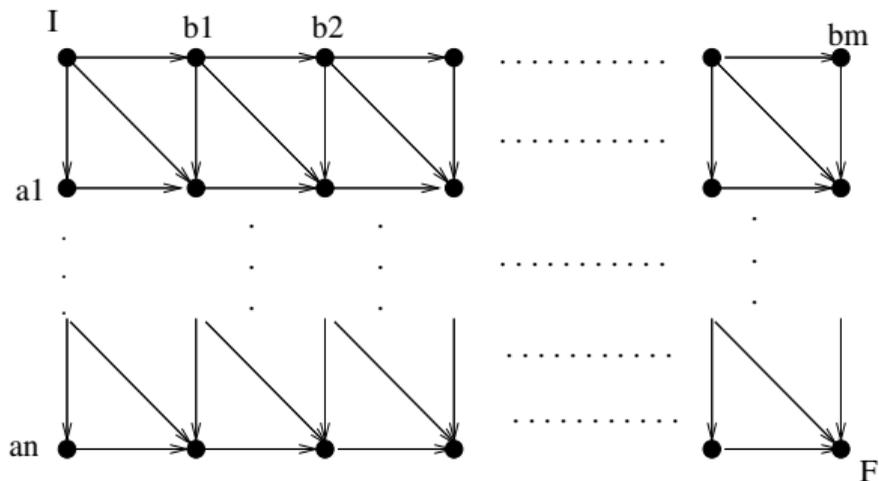
Redução: $ES \propto PCM$

- Instância de **ES**: duas strings $A = a_1a_2 \dots a_n$ e $B = b_1b_2 \dots b_m$.
- τ_I constrói a instância de **PCM**, um grafo direcionado G com $|V| = (n+1)(m+1)$:



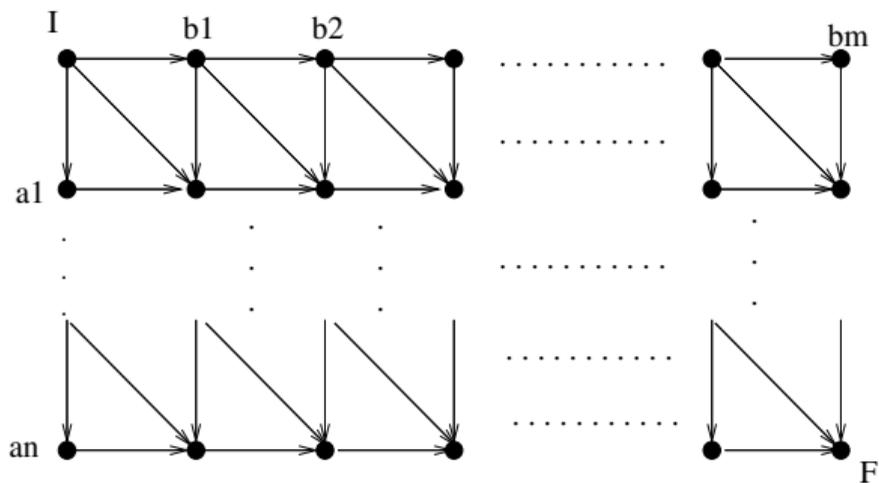
Reduções entre problemas: Exemplo $ES \propto PCM$

- 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 $CLI \propto IND$

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:

- Em uma rede social, uma 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:

- Em uma rede social, uma clique é um grupo de pessoas em que todas se conhecem.
- Em redes funcionais de genes, uma 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 $CLI \propto IND$

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 .

Observações:

- Em uma rede social, um conjunto independente é um grupo de pessoas entre as quais não há duas que se conheçam.

Reduções entre problemas: Exemplo $CLI \propto IND$

Redução: $CLI \propto IND$

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

Reduções entre problemas: Exemplo $\text{CLI} \propto \text{IND}$

Redução: $\text{CLI} \propto \text{IND}$

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

Reduções entre problemas: Exemplo $CLI \propto IND$

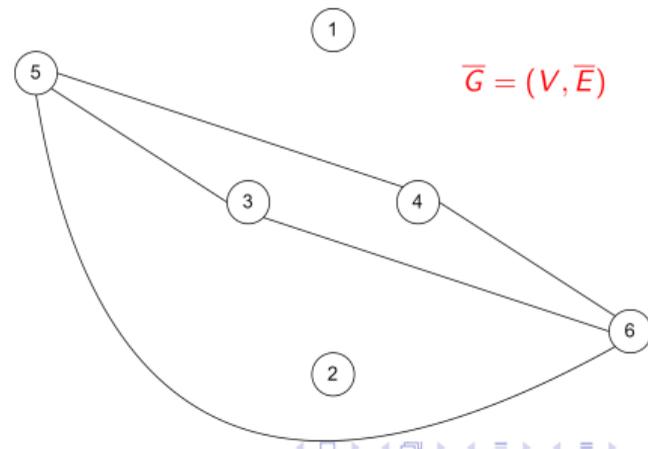
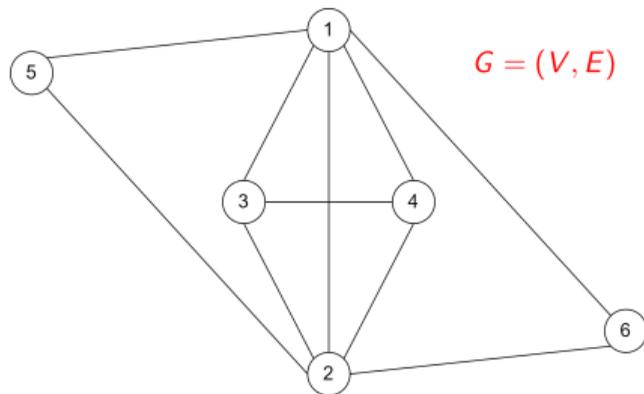
Redução: $CLI \propto IND$

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

Reduções entre problemas: Exemplo $CLI \propto IND$

Redução: $CLI \propto IND$

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



Reduções entre problemas: Exemplo $CLI \propto IND$

Observações importantes:

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

Reduções entre problemas: Exemplo $\text{CLI} \propto \text{IND}$

Observações importantes:

- Veja que não apenas $\text{CLI} \propto \text{IND}$, mas também $\text{IND} \propto \text{CLI}$.
- 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 $\text{CLI} \propto \text{IND}$

Observações importantes:

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

Reduções entre problemas: Exemplo $\text{CLI} \propto \text{IND}$

Observações importantes:

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

Observações importantes:

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

Observações importantes:

- Veja que não apenas $\text{CLI} \propto \text{IND}$, mas também $\text{IND} \propto \text{CLI}$.
- A redução **não é linear**. (Por quê?)
- Intuitivamente, quando faria sentido usar essa redução para resolver **CLI**?
- O problema do conjunto independente (**IND**) 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 $\bar{G} = (V, \bar{E})$:

Algorithm $\tau_I(G)$

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

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 $\tau_S(S)$

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 $\tau_S(S)$

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} .

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 $G = (V, E)$.

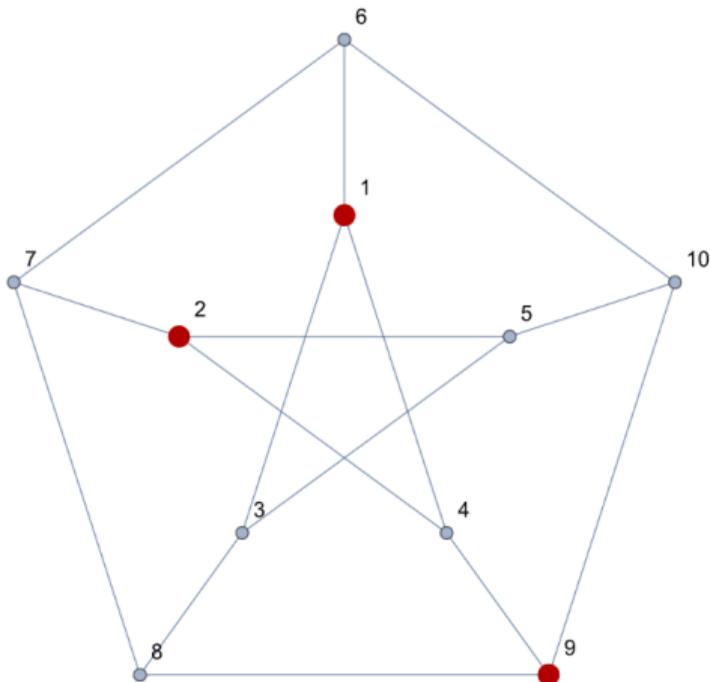
Objetivo: encontrar o menor conjunto dominante em G , ou seja, o menor conjunto de vértices S tal que cada vértice $v \notin S$ é adjacente a pelo menos um vértice em S .

Observações:

- Em uma rede social, um conjunto dominante é um grupo de propagadores capazes de atingir a (se comunicar com) todos os indivíduos da rede.
- Este problema e suas variantes são usadas 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 $x_1, x_2, \dots, x_n = x$ 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 $x_1, x_2, \dots, x_n = x$ que minimizem a soma $\sum_{j=1}^n c_j x_j$, sujeito às restrições $Ax \leq b$.

Observação: matematicamente, o problema geralmente é formulado assim:

$$\min \sum c^T x$$

sujeito a $Ax \leq b, x \geq 0$

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 $x_1, x_2, \dots, x_n = x$ que minimizem a soma $\sum_{j=1}^n c_j x_j$, sujeito às restrições $Ax \leq b$.

Observação: matematicamente, o problema geralmente é formulado assim:

$$\min \sum c^T x$$

sujeito a $Ax \leq b, x \geq 0$

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

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

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

Reduções entre problemas: Exemplo $MDS \propto ILP$

Redução: $MDS \propto ILP$

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

Reduções entre problemas: Exemplo $\text{MDS} \propto \text{ILP}$

Redução: $\text{MDS} \propto \text{ILP}$

- 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 $j \in \text{MDS}$;

Reduções entre problemas: Exemplo $\text{MDS} \propto \text{ILP}$

Redução: $\text{MDS} \propto \text{ILP}$

- 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 $j \in \text{MDS}$;
 - Para cada vértice $j \in V$ cria uma restrição $\sum_{i \in \text{Adj}[j] \cup \{j\}} x_i \geq 1$.
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**:

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

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

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

Reduções entre problemas: Exemplo $MDS \propto ILP$

- Formulação completa do problema **ILP**:

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

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

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

- 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**:

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

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

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

- 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.

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.
- Diversos problemas em grafos podem ser transformados nesse tipo de problema, chamado de **problema 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.
- Diversos problemas em grafos podem ser transformados nesse tipo de problema, chamado de **problema de programação linear**.
- Neste caso, usamos variáveis inteiras (em particular, binárias, **0** ou **1**), por isso chamamos essa variante de **Problema de Programação Linear Inteira – ILP**.

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.
- Diversos problemas em grafos podem ser transformados nesse tipo de problema, chamado de **problema de programação linear**.
- Neste caso, usamos variáveis inteiras (em particular, binárias, **0** ou **1**), por isso chamamos essa variante de **Problema de Programação Linear Inteira – ILP**.
- Se as variáveis são reais, o problema é chamado **Problema de Programação Linear – LP** (sem o sobrenome “inteira”).

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.

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 indispensável hoje para a sobrevivência e o crescimento de companhias de grande e médio porte (e.g., **Amazon**).

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 indispensável hoje para a sobrevivência e o crescimento de companhias de grande e médio porte (e.g., **Amazon**). 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 indispensável hoje para a sobrevivência e o crescimento de companhias de grande e médio porte (e.g., **Amazon**). 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 uso de recursos e tarefas usando o termo “programar atividades” (*program*, ou *schedule*).

Material bibliográfico e exercícios

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

Exercícios: ver exercícios no final do Capítulo 10.

Dúvidas

Dúvidas?