Projeto e Análise de Algoritmos II (MC558)
Classes de Problemas

Prof. Dr. Ruben Interian

Resumo

Resumo

1. Revisão do conteúdo e objetivo
2. Provas de $\mathcal{NP}$-completude
3. Complexidade de espaço
4. Quiz
5. Síntese

Revisão do conteúdo

- Diversos problemas que são $\mathcal{NP}$-completos.
- A descoberta de um algoritmo eficiente para um deles implica automaticamente na existência de algoritmo eficiente para todos eles.
- Perceber ou mostrar a $\mathcal{NP}$-completude de um problema é a forma que temos para atestar que o problema é complexo.

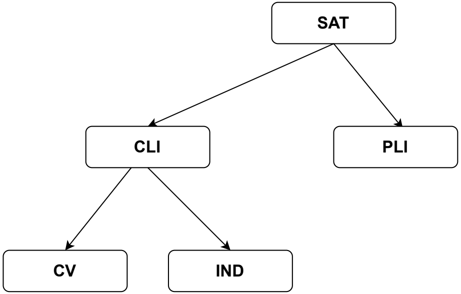
Objetivo

Responder à pergunta:

- Como identificar um problema complexo? Como mostrar que ele é complexo?
  - "Eu não consigo resolver" não é suficiente!

Resumo

Provas de $\mathcal{NP}$-completude: nosso estado da arte



Provas de $\mathcal{NP}$-completude: **DS**

Problema do conjunto dominante (DS)
Versão de decisão

Dado um grafo não-direcionado $G = (V, E)$, e um valor inteiro k, decidir se G contém um conjunto dominante com k vértices.

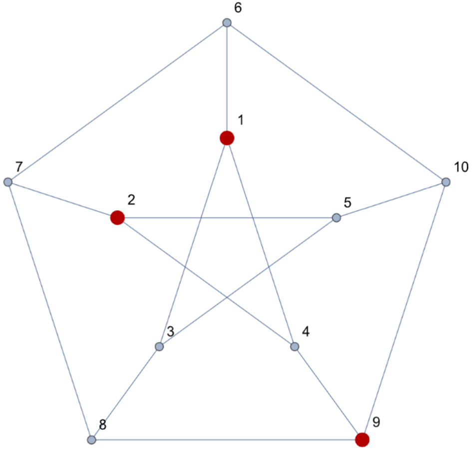
Lembrando: Um conjunto dominante em G é um conjunto de vértices S tal que cada vértice $v \notin S$ é adjacente a pelo menos um vértice em S. Geralmente temos interesse no **menor** conjunto dominante.

Teorema

DS é $\mathcal{NP}$-completo.

Provas de $\mathcal{NP}$-completude: **DS**

Exemplo de instância do problema do conjunto dominante



Provas de $\mathcal{NP}$-completude: **DS**

- **DS** $\in \mathcal{NP}$, pois existe um algoritmo determinístico que, dado uma conjunto de vértices S de um grafo, verifica, em tempo polinomial, se eles são um conjunto dominante **checando se** entre os adjacentes de **cada vértice** tem pelo menos um vértice que está em S. ☑

Provas de $\mathcal{NP}$-completude: **DS**

- **DS** $\in \mathcal{NP}$, pois existe um algoritmo determinístico que, dado uma conjunto de vértices S de um grafo, verifica, em tempo polinomial, se eles são um conjunto dominante **checando se** entre os adjacentes de **cada vértice** tem pelo menos um vértice que está em S. ☑
- **DS** $\in \mathcal{NP}$-difícil.
  - Precisamos escolher um problema do qual faremos a redução.

Provas de $\mathcal{NP}$-completude: **DS**

- **DS** $\in \mathcal{NP}$, pois existe um algoritmo determinístico que, dado uma conjunto de vértices S de um grafo, verifica, em tempo polinomial, se eles são um conjunto dominante **checando se** entre os adjacentes de **cada vértice** tem pelo menos um vértice que está em S. ☑
- **DS** $\in \mathcal{NP}$-difícil.
  - Precisamos escolher um problema do qual faremos a redução.

Vamos provar que **DS** $\in \mathcal{NP}$-difícil mostrando que **CV** $\propto_{\text{poli}}$ **DS**.

Provas de $\mathcal{NP}$-completude: **DS**

Transformação da instância do **CV** em uma instância **DS**.
Seja $G = (V, E)$, k uma instância do **CV**. Construa uma instância do **DS** formada por um grafo $G' = (V', E')$, e pelo mesmo inteiro k.

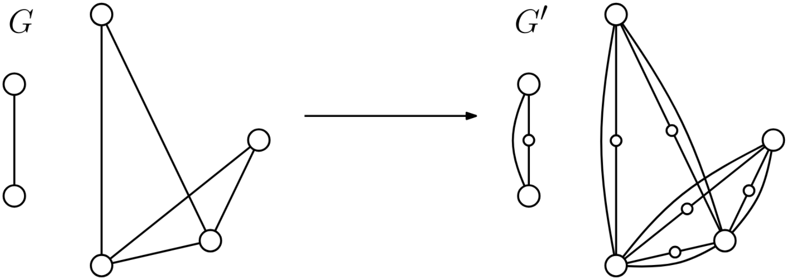
Vértices de G':

- Todo vértice de G também é vértice de G'.
  Além disso, criaremos um vértice v_{ij} para cada aresta (i, j) em E.
- Se V_A é o conjunto de vértices criados dessa forma, então $V' = V \cup V_A$.

Arestas de G':

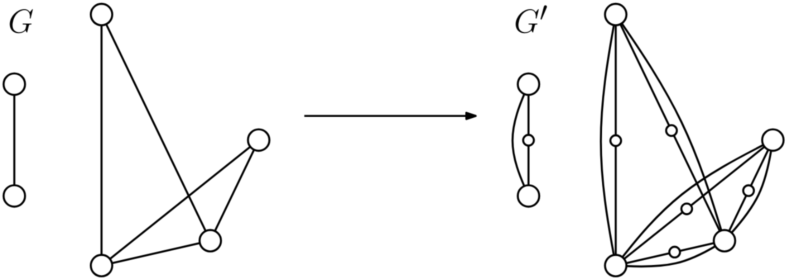
- Toda aresta de G também é aresta de G'.
  Além disso, para cada vértice $v_{ij} \in V_A$ criaremos duas arestas: (v_{ij}, i) e (v_{ij}, j).
- Se E_A é o conjunto das arestas criadas desta forma, $E' = E \cup E_A$.

Provas de $\mathcal{NP}$-completude: **DS**



Se $|V| = n$ e $|E| = m$, a instância de entrada de **DS** é obtida em $O(n+m)$.

Provas de $\mathcal{NP}$-completude: **DS**

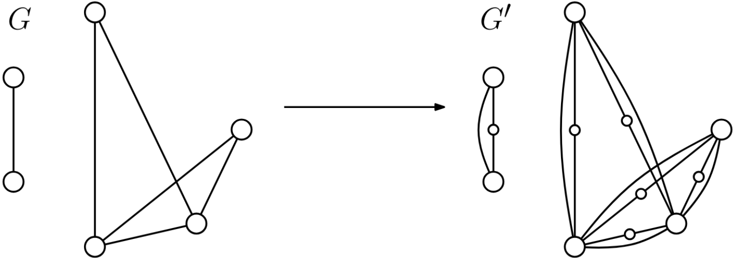


Se $|V| = n$ e $|E| = m$, a instância de entrada de **DS** é obtida em $O(n+m)$.

G, k é `SIM` para **CV** $\Leftrightarrow$ G', k é `SIM` para **DS**

Existe uma CV de tamanho k em G $\Leftrightarrow$ Existe um DS de tamanho k em G'

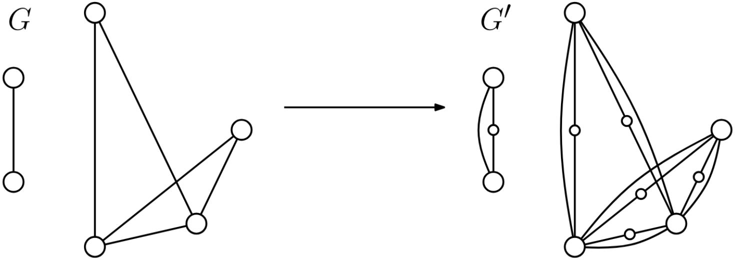
Provas de $\mathcal{NP}$-completude: **DS**



Existe uma CV de tamanho k em G $\Leftrightarrow$ Existe um DS de tamanho k em G'

$\Leftarrow$ Seja D um conjunto dominante com k vértices em G'. Se D possui algum vértice criado v_{ij}, podemos substituir v_{ij} por i (ou por j). Podemos supor então que D apenas possui vértices de V. Como D domina a todos os vértices novos, D possui ao menos um vértice de cada aresta em E, portanto D é uma cobertura de vértices.

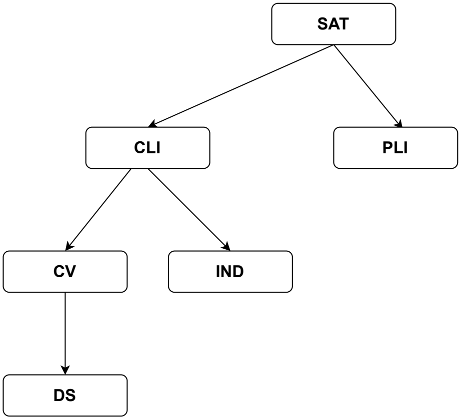
Provas de $\mathcal{NP}$-completude: **DS**



Existe uma CV de tamanho k em G $\Leftrightarrow$ Existe um DS de tamanho k em G'

$\Rightarrow$ Seja C uma cobertura de vértices com k vértices em G. Então, cada aresta está coberta por C (possui um extremo em C), e todos os vértices em G' estão dominados, incluindo os vértices em V e os vértices criados em V_A.

Provas de $\mathcal{NP}$-completude: nosso estado da arte



Provas de $\mathcal{NP}$-completude: **3SAT**

Problema da 3-satisfatibilidade (3SAT)

Dada uma fórmula lógica $\mathcal{F}$ na forma normal conjuntiva, cada cláusula contendo exatamente 3 literais, decidir se existe uma atribuição de valores às variáveis de modo que $\mathcal{F} = 1$.

Provas de $\mathcal{NP}$-completude: **3SAT**

Teorema

3SAT é $\mathcal{NP}$-completo.

Provas de $\mathcal{NP}$-completude: **3SAT**

Teorema

3SAT é $\mathcal{NP}$-completo.

- É evidente que **3SAT** $\in \mathcal{NP}$, pois **SAT** $\in \mathcal{NP}$ e **3SAT** é caso particular de **SAT**. Podemos usar o mesmo algoritmo verificador de **SAT** para **3SAT**.

Provas de $\mathcal{NP}$-completude: **3SAT**

Teorema

3SAT é $\mathcal{NP}$-completo.

- É evidente que **3SAT** $\in \mathcal{NP}$, pois **SAT** $\in \mathcal{NP}$ e **3SAT** é caso particular de **SAT**. Podemos usar o mesmo algoritmo verificador de **SAT** para **3SAT**.
- **3SAT** $\in \mathcal{NP}$-difícil, pois **SAT** $\propto_{\text{poli}}$ **3SAT**.
  - Esse fato não parece nada obvio...

Omitiremos essa redução aqui. Ver no Capítulo 11.4 do Manber.

Provas de $\mathcal{NP}$-completude: **3SAT**

OK, sabemos que **3SAT** é $\mathcal{NP}$-completo.

Mas **1SAT**, **2SAT**?...

Provas de $\mathcal{NP}$-completude: **3SAT**

OK, sabemos que **3SAT** é $\mathcal{NP}$-completo.

Mas **1SAT**, **2SAT**?...

- **1SAT**: $\mathcal{F} = x_1 \cdot x_2 \cdot \ldots \cdot x_n$. Quando $\mathcal{F} = 1$?
  - $\mathcal{F} = 1$ se cada $x_i = 1$ para todo $i = 1, \ldots, n$. Bem fácil de resolver.

Provas de $\mathcal{NP}$-completude: **3SAT**

OK, sabemos que **3SAT** é $\mathcal{NP}$-completo.

Mas **1SAT**, **2SAT**?...

- **1SAT**: $\mathcal{F} = x_1 \cdot x_2 \cdot \ldots \cdot x_n$. Quando $\mathcal{F} = 1$?
  - $\mathcal{F} = 1$ se cada $x_i = 1$ para todo $i = 1, \ldots, n$. Bem fácil de resolver.
- **2SAT**: $\mathcal{F} = (x_i + x_j) \cdot \ldots \cdot (x_k + x_l)$.
  - Será que **2SAT** é $\mathcal{NP}$-completo?

Provas de $\mathcal{NP}$-completude: **3SAT**

OK, sabemos que **3SAT** é $\mathcal{NP}$-completo.

Mas **1SAT**, **2SAT**?...

- **1SAT**: $\mathcal{F} = x_1 \cdot x_2 \cdot \ldots \cdot x_n$. Quando $\mathcal{F} = 1$?
  - $\mathcal{F} = 1$ se cada $x_i = 1$ para todo $i = 1, \ldots, n$. Bem fácil de resolver.
- **2SAT**: $\mathcal{F} = (x_i + x_j) \cdot \ldots \cdot (x_k + x_l)$.
  - Será que **2SAT** é $\mathcal{NP}$-completo?

    Não. Existe um algoritmo determinístico polinomial para resolver **2SAT**!

Provas de ~~$\mathcal{NP}$-completude~~: **2SAT**

2SAT $\in \mathcal{P}$.

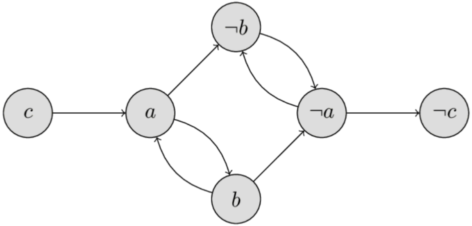
$\mathcal{F} = (a + \overline{b}) \cdot (\overline{a} + b) \cdot (\overline{a} + \overline{b}) \cdot (a + \overline{c})$.

- Veja que $x + y$ é equivalente a $\overline{x} \Rightarrow y$, e também equivalente a $\overline{y} \Rightarrow x$.
- Cada cláusula vira duas implicações. No exemplo, são oito implicações ao todo.

Provas de ~~$\mathcal{NP}$-completude~~: 2SAT

2SAT $\in \mathcal{P}$. $\mathcal{F} = (a + \overline{b}) \cdot (\overline{a} + b) \cdot (\overline{a} + \overline{b}) \cdot (a + \overline{c})$.

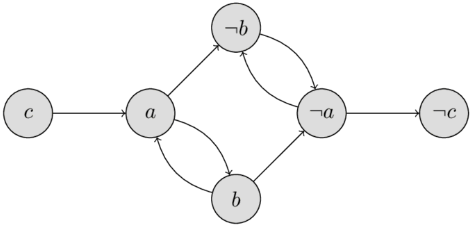
- Veja que $x + y$ é equivalente a $\overline{x} \Rightarrow y$, e também equivalente a $\overline{y} \Rightarrow x$.
- Cada cláusula vira duas implicações. No exemplo, são oito implicações ao todo.
- Vamos criar um grafo direcionado $G = (V, E)$. Para cada variável teremos dois vértices, por exemplo, a é um vértice e $\overline{a}$ é outro vértice. Teremos seis vértices.
- As implicações são os arcos. O número de arcos é o dobro do número de cláusulas.



Provas de ~~$\mathcal{NP}$-completude~~: 2SAT

2SAT $\in \mathcal{P}$. $\mathcal{F} = (a + \overline{b}) \cdot (\overline{a} + b) \cdot (\overline{a} + \overline{b}) \cdot (a + \overline{c})$.

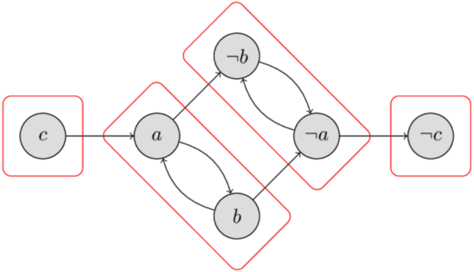
- Se neste grafo, temos uma variável a alcançável a partir de $\overline{a}$, e $\overline{a}$ alcançável a partir de a, não há solução do **2SAT**. Isso significaria $a \Leftrightarrow \overline{a}$!
- O problema fica reduzido a encontrar vértices mutuamente alcançáveis em G.



Provas de ~~$\mathcal{NP}$-completude~~: 2SAT

2SAT $\in \mathcal{P}$. $\mathcal{F} = (a + \overline{b}) \cdot (\overline{a} + b) \cdot (\overline{a} + \overline{b}) \cdot (a + \overline{c})$.

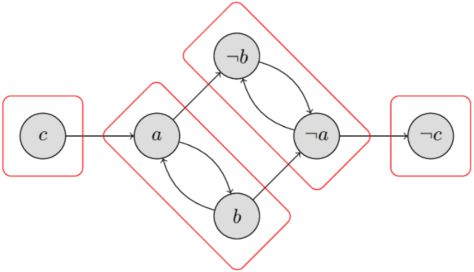
- Se neste grafo, temos uma variável a alcançável a partir de $\overline{a}$, e $\overline{a}$ alcançável a partir de a, não há solução do **2SAT**. Isso significaria $a \Leftrightarrow \overline{a}$!
- O problema fica reduzido a encontrar vértices mutuamente alcançáveis em G.
- Este é o problema de encontrar **componentes fortemente conexas** em G.
- É possível resolver **CFC** em tempo linear (lembre o algoritmo visto em aula)!



Provas de ~~$\mathcal{NP}$-completude~~: 2SAT

2SAT $\in \mathcal{P}$. $\mathcal{F} = (a + \overline{b}) \cdot (\overline{a} + b) \cdot (\overline{a} + \overline{b}) \cdot (a + \overline{c})$.

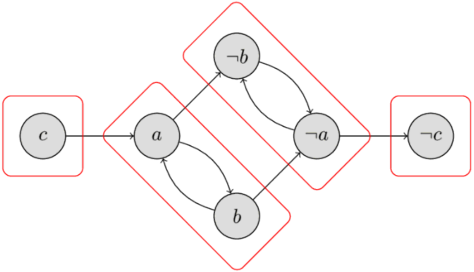
- Neste exemplo em particular, todos os pares das variáveis e as suas negações estão em componentes diferentes.
- Portanto, $\mathcal{F}$ é satisfatível. Mas como encontrar a solução?



Provas de ~~$\mathcal{NP}$-completude~~: 2SAT

2SAT $\in \mathcal{P}$. $\mathcal{F} = (a + \overline{b}) \cdot (\overline{a} + b) \cdot (\overline{a} + \overline{b}) \cdot (a + \overline{c})$.

- Neste exemplo em particular, todos os pares das variáveis e as suas negações estão em componentes diferentes.
- Portanto, $\mathcal{F}$ é satisfatível. Mas como encontrar a solução?
- Simples: faça uma ordenação topológica.
  $\text{comp}[x] < \text{comp}[\overline{x}]$ (x aparece antes de $\overline{x}$) $\Rightarrow$ $x = \texttt{False}$. Senão, $x = \texttt{True}$.



Provas de $\mathcal{NP}$-completude: **3SAT**

Resumo:

- **2SAT** $\in \mathcal{P}$.
- **3SAT** $\in \mathcal{NP}$-completo.

Provas de $\mathcal{NP}$-completude: **3SAT**

Resumo:

- **2SAT** $\in \mathcal{P}$.
- **3SAT** $\in \mathcal{NP}$-completo.

Em muitos livros e estudos, ao invés de usar reduções a partir de **SAT**, são usadas reduções a partir de **3SAT**. Algumas até ficam mais fáceis!

Provas de $\mathcal{NP}$-completude: **3SAT** e problemas relacionados

Problema da k-coloração (kCOL)

Dado um grafo $G = (V, E)$, decidir se G possui uma coloração válida com k cores.

Lembrando: Uma **coloração válida** de um grafo é uma atribuição de cores aos seus vértices tal que dois vértices adjacentes tenham cores distintas. Existe o problema da 2-coloração (**2COL**), 3-coloração (**3COL**)...

Provas de $\mathcal{NP}$-completude: **3SAT** e problemas relacionados

Problema da k-coloração (kCOL)

Dado um grafo $G = (V, E)$, decidir se G possui uma coloração válida com k cores.

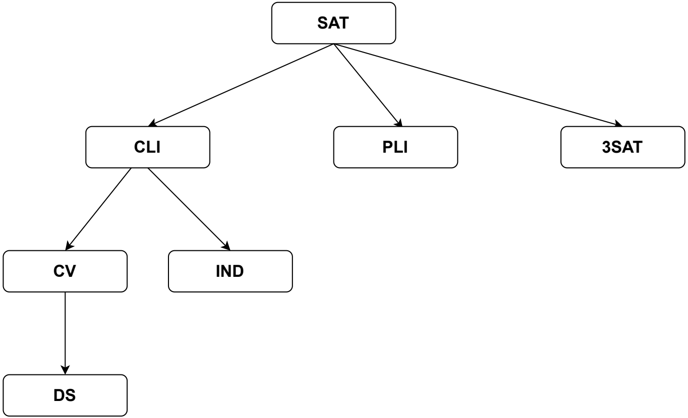
Lembrando: Uma **coloração válida** de um grafo é uma atribuição de cores aos seus vértices tal que dois vértices adjacentes tenham cores distintas. Existe o problema da 2-coloração (**2COL**), 3-coloração (**3COL**)...

Surpreendentemente, temos de novo:

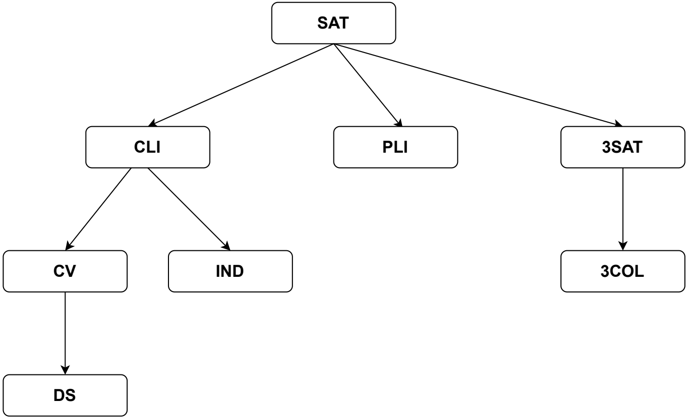
- **2COL** $\in \mathcal{P}$ (Que problema já visto é esse?).
- **3COL** $\in \mathcal{NP}$-completo, pois **3SAT** $\propto_{\text{poli}}$ **3COL**.

Ver **3SAT** $\propto_{\text{poli}}$ **3COL** no Capítulo 11.4 do Manber.

Provas de $\mathcal{NP}$-completude: nosso estado da arte



Provas de $\mathcal{NP}$-completude: nosso estado da arte



Outros problemas $\mathcal{NP}$-completos

- **Caminho Hamiltoniano**: (não confundir com Caminho Euleriano, que $\in \mathcal{P}$)

  Dado um grafo não-direcionado $G = (V, E)$, decidir se G contém um caminho que passa uma única vez por todos os vértices.

Outros problemas $\mathcal{NP}$-completos

- **Caminho Hamiltoniano**: (não confundir com Caminho Euleriano, que $\in \mathcal{P}$)
  Dado um grafo não-direcionado $G = (V, E)$, decidir se G contém um caminho que passa uma única vez por todos os vértices.

- **Ciclo Hamiltoniano**:
  Dado um grafo não-direcionado $G = (V, E)$, decidir se G contém um ciclo que passa uma única vez por todos os vértices.

Outros problemas $\mathcal{NP}$-completos

- **Caminho Hamiltoniano**: (não confundir com Caminho Euleriano, que $\in \mathcal{P}$)
  Dado um grafo não-direcionado $G = (V, E)$, decidir se G contém um caminho que passa uma única vez por todos os vértices.

- **Ciclo Hamiltoniano**:
  Dado um grafo não-direcionado $G = (V, E)$, decidir se G contém um ciclo que passa uma única vez por todos os vértices.

- Podemos mostrar que esses dois problemas são $\mathcal{NP}$-completos usando uma redução polinomial do problema **CV**. Há outra redução do problema **3SAT**.

Outros problemas $\mathcal{NP}$-completos

- Problema do **Caixeiro-Viajante** (TSP)

  Relato original: Um caixeiro precisa viajar para n cidades e retornar ao ponto de partida. Qual rota ele deve seguir para minimizar a distância percorrida?

  (O número de rotas possíveis pode ser até $n!$.)

Outros problemas $\mathcal{NP}$-completos

- Problema do **Caixeiro-Viajante** (TSP)

  Relato original: Um caixeiro precisa viajar para n cidades e retornar ao ponto de partida. Qual rota ele deve seguir para minimizar a distância percorrida?

  (O número de rotas possíveis pode ser até $n!$.)

  Versão de **otimização**:

  Dado um grafo não-direcionado e ponderado $G = (V, E)$, com custos d_e nas arestas, encontrar um ciclo hamiltoniano de custo mínimo.

Outros problemas $\mathcal{NP}$-completos

- Problema do **Caixeiro-Viajante** (TSP)

  Relato original: Um caixeiro precisa viajar para n cidades e retornar ao ponto de partida. Qual rota ele deve seguir para minimizar a distância percorrida?

  (O número de rotas possíveis pode ser até $n!$.)

  Versão de **otimização**:

  Dado um grafo não-direcionado e ponderado $G = (V, E)$, com custos d_e nas arestas, encontrar um ciclo hamiltoniano de custo mínimo.

  Versão de **decisão**:

  Dado um grafo não-direcionado e ponderado $G = (V, E)$, com custos d_e nas arestas, e um valor D, decidir se G tem um ciclo hamiltoniano de custo $\leq D$.

Outros problemas $\mathcal{NP}$-completos

- Problema do **Caixeiro-Viajante** (TSP)

  Relato original: Um caixeiro precisa viajar para n cidades e retornar ao ponto de partida. Qual rota ele deve seguir para minimizar a distância percorrida?

  (O número de rotas possíveis pode ser até $n!$.)

  Versão de **otimização**:

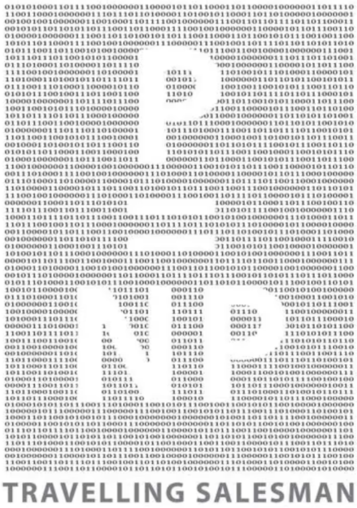
  Dado um grafo não-direcionado e ponderado $G = (V, E)$, com custos d_e nas arestas, encontrar um ciclo hamiltoniano de custo mínimo.

  Versão de **decisão**:

  Dado um grafo não-direcionado e ponderado $G = (V, E)$, com custos d_e nas arestas, e um valor D, decidir se G tem um ciclo hamiltoniano de custo $\leq D$.

- Há uma redução simples do problema Ciclo Hamiltoniano para o problema do Caixeiro-Viajante.

Outros problemas $\mathcal{NP}$-completos



Resumo

Complexidade de espaço

- **Definição**: um problema A possui **complexidade de espaço** $f(n)$ se existe um algoritmo que, para toda instância de tamanho n, usa $f(n)$ espaço (**memória**) no pior caso para resolvê-lo, e não há algoritmo com uso de espaço menor.
  - Definição de $O(f(n))$ usual.

Complexidade de espaço

- **Definição**: um problema A possui **complexidade de espaço** $f(n)$ se existe um algoritmo que, para toda instância de tamanho n, usa $f(n)$ espaço (**memória**) no pior caso para resolvê-lo, e não há algoritmo com uso de espaço menor.
  - Definição de $O(f(n))$ usual.
- **Definição**: $\mathcal{PSPACE}$ é a classe dos problemas que admitem algoritmos **determinísticos** que usam **espaço polinomial** no tamanho da entrada.

Complexidade de espaço

- **Definição**: um problema A possui **complexidade de espaço** $f(n)$ se existe um algoritmo que, para toda instância de tamanho n, usa $f(n)$ espaço (**memória**) no pior caso para resolvê-lo, e não há algoritmo com uso de espaço menor.
    - Definição de $O(f(n))$ usual.
- **Definição**: $\mathcal{PSPACE}$ é a classe dos problemas que admitem algoritmos **determinísticos** que usam **espaço polinomial** no tamanho da entrada.
- Alguns fatos:
    - $\mathcal{P} \in \mathcal{PSPACE}$
    - $\mathcal{NP} \in \mathcal{PSPACE}$

Complexidade de espaço

- **Definição**: um problema A possui **complexidade de espaço** $f(n)$ se existe um algoritmo que, para toda instância de tamanho n, usa $f(n)$ espaço (**memória**) no pior caso para resolvê-lo, e não há algoritmo com uso de espaço menor.
  - Definição de $O(f(n))$ usual.
- **Definição**: $\mathcal{PSPACE}$ é a classe dos problemas que admitem algoritmos **determinísticos** que usam **espaço polinomial** no tamanho da entrada.
- Alguns fatos:
  - $\mathcal{P} \in \mathcal{PSPACE}$
  - $\mathcal{NP} \in \mathcal{PSPACE}$
- **Definição**: $\mathcal{NPSPACE}$ é a classe dos problemas que admitem algoritmos **não determinísticos** que usam **espaço polinomial** no tamanho da entrada.

Complexidade de espaço

$$\mathcal{PSPACE} \stackrel{?}{=} \mathcal{NPSPACE}$$

Complexidade de espaço

$$\mathcal{PSPACE} \stackrel{?}{=} \mathcal{NPSPACE}$$

- Sabemos que $\mathcal{PSPACE} \subseteq \mathcal{NPSPACE}$.

Complexidade de espaço

$$\mathcal{PSPACE} \stackrel{?}{=} \mathcal{NPSPACE}$$

- Sabemos que $\mathcal{PSPACE} \subseteq \mathcal{NPSPACE}$.
- Complexidade de tempo e de espaço de algoritmos não-determinísticos:
  - Complexidade de tempo: algoritmos não-determinísticos parecem ter alguma vantagem. O tempo perdido com **Escolhas** erradas não pode ser recuperado!

Complexidade de espaço

$$\mathcal{PSPACE} \stackrel{?}{=} \mathcal{NPSPACE}$$

- Sabemos que $\mathcal{PSPACE} \subseteq \mathcal{NPSPACE}$.
- Complexidade de tempo e de espaço de algoritmos não-determinísticos:
  - Complexidade de tempo: algoritmos não-determinísticos parecem ter alguma vantagem. O tempo perdido com **Escolha**s erradas não pode ser recuperado!
  - Complexidade de espaço: a memória pode ser reaproveitada! Todas as sequências de **Escolha**s podem ser simuladas usando "quase" o mesmo espaço.

Complexidade de espaço

$$\mathcal{PSPACE} \stackrel{?}{=} \mathcal{NPSPACE}$$

- Sabemos que $\mathcal{PSPACE} \subseteq \mathcal{NPSPACE}$.
- Complexidade de tempo e de espaço de algoritmos não-determinísticos:
  - Complexidade de tempo: algoritmos não-determinísticos parecem ter alguma vantagem. O tempo perdido com **Escolha**s erradas não pode ser recuperado!
  - Complexidade de espaço: a memória pode ser reaproveitada! Todas as sequências de **Escolha**s podem ser simuladas usando "quase" o mesmo espaço.
  - Consequência: uma Máquina de Turing não determinística pode ser "simulada" por uma Máquina de Turing determinística sem usar muito mais espaço.

Complexidade de espaço

$$\mathcal{PSPACE} \stackrel{?}{=} \mathcal{NPSPACE}$$

- Sabemos que $\mathcal{PSPACE} \subseteq \mathcal{NPSPACE}$.
- Complexidade de tempo e de espaço de algoritmos não-determinísticos:
  - Complexidade de tempo: algoritmos não-determinísticos parecem ter alguma vantagem. O tempo perdido com **Escolha**s erradas não pode ser recuperado!
  - Complexidade de espaço: a memória pode ser reaproveitada! Todas as sequências de **Escolha**s podem ser simuladas usando "quase" o mesmo espaço.
  - Consequência: uma Máquina de Turing não determinística pode ser "simulada" por uma Máquina de Turing determinística sem usar muito mais espaço.

$$\mathcal{PSPACE} = \mathcal{NPSPACE}.$$

Resumo

Quiz

Seja X um problema que pertence à classe $\mathcal{NP}$.

Quais das seguintes afirmações são verdadeiras?

- Não existe algoritmo polinomial para X.
- Se existe um algoritmo determinístico polinomial que resolve X, então $\mathcal{P} = \mathcal{NP}$.
- Se X é $\mathcal{NP}$-difícil, então X é $\mathcal{NP}$-completo.

Quiz

Seja X um problema que pertence à classe $\mathcal{NP}$.

Quais das seguintes afirmações são verdadeiras?

- Não existe algoritmo polinomial para X.
- Se existe um algoritmo determinístico polinomial que resolve X, então $\mathcal{P} = \mathcal{NP}$.
- Se X é $\mathcal{NP}$-difícil, então X é $\mathcal{NP}$-completo. ☑

Quiz

Seja A o problema de decisão de determinar se existe um ciclo em um grafo não direcionado. Quais das seguintes afirmações são verdadeiras?

(1) A está em $\mathcal{P}$. (2) A está em $\mathcal{NP}$.

- Nenhuma
- Apenas 1
- Apenas 2
- Ambas, 1 e 2

Quiz

Seja A o problema de decisão de determinar se existe um ciclo em um grafo não direcionado. Quais das seguintes afirmações são verdadeiras?

(1) A está em $\mathcal{P}$. (2) A está em $\mathcal{NP}$.

- Nenhuma
- Apenas 1
- Apenas 2
- Ambas, 1 e 2 ☑

Quiz

Seja X um problema $\mathcal{NP}$-completo e Q e R dois problemas de decisão. Q é polinomialmente redutível a X, e X é polinomialmente redutível a R.

Quais das seguintes afirmações são verdadeiras?

- O problema Q é $\mathcal{NP}$-difícil.
- O problema R é $\mathcal{NP}$-difícil.
- O problema Q é $\mathcal{NP}$-completo.
- O problema R é $\mathcal{NP}$-completo.

Quiz

Seja X um problema $\mathcal{NP}$-completo e Q e R dois problemas de decisão. Q é polinomialmente redutível a X, e X é polinomialmente redutível a R.

Quais das seguintes afirmações são verdadeiras?

- O problema Q é $\mathcal{NP}$-difícil.
- O problema R é $\mathcal{NP}$-difícil. ☑
- O problema Q é $\mathcal{NP}$-completo.
- O problema R é $\mathcal{NP}$-completo.

Quiz

Um problema A é polinomialmente redutível ao problema **CLI**, e **CLI** é polinomialmente redutível a A.

Quais das seguintes afirmações podem ser inferidas dessas reduções?

- A não é nem $\mathcal{NP}$, nem $\mathcal{NP}$-difícil.
- A é $\mathcal{NP}$, mas não é $\mathcal{NP}$-completo.
- A é $\mathcal{NP}$-difícil, mas não é $\mathcal{NP}$-completo.
- A é $\mathcal{NP}$-completo.

Quiz

Um problema A é polinomialmente redutível ao problema **CLI**, e **CLI** é polinomialmente redutível a A.

Quais das seguintes afirmações podem ser inferidas dessas reduções?

- A não é nem $\mathcal{NP}$, nem $\mathcal{NP}$-difícil.
- A é $\mathcal{NP}$, mas não é $\mathcal{NP}$-completo.
- A é $\mathcal{NP}$-difícil, mas não é $\mathcal{NP}$-completo.
- A é $\mathcal{NP}$-completo. ☑

Resumo

Síntese

- Temos um repertório com diversos problemas $\mathcal{NP}$-completos, os quais podem ser usados para mostrar a complexidade de problemas reais e para provar $\mathcal{NP}$-completude de outros problemas.

Material bibliográfico e exercícios

U. Manber. Introduction to Algorithms. – **Cap. 11**.

Exercícios: ver exercícios no final do Capítulo 11.

Bibliografia complementar:

Combinatorial Optimization: Algorithms and Complexity
C.H. Papadimitriou e K. Steiglitz, Dover, 1982.

The Design and Analysis of Computer Algorithms
A.V. Aho, J.E. Hopcroft e J.D. Ullman, Addison-Wesley, 1974.

Dúvidas

Dúvidas?