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

Prof. Dr. Ruben Interian

Resumo

Resumo

Revisão do conteúdo

- Diversos problemas 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, ou da sua versão de decisão, é a forma que temos para atestar que o problema é complexo.

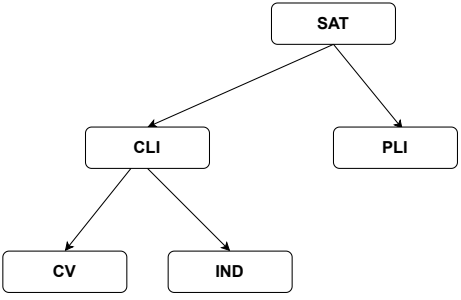
Objetivo

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.

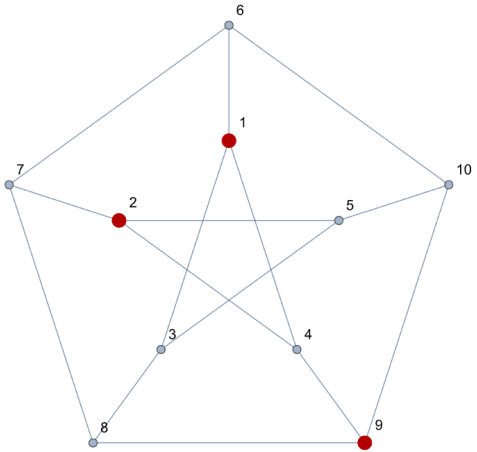
Lembrando: Um conjunto dominante em G é um conjunto de vértices S tal que cada vértice $v \notin S$ é adjacente a pelo menos um vértice em S. Geralmente temos interesse no **menor** conjunto dominante.

Teorema

DS é $\mathcal{NP}$-completo.

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

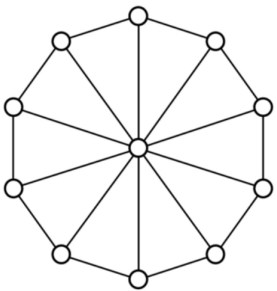
Exemplo de instância do problema do conjunto dominante



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

Fonte de confusão e erros: os problemas **CV** e **DS** são diferentes!

- **CV**: a condição precisa ser cumprida por cada aresta! (Um dos extremos $\in S$.)
- **DS**: a condição precisa ser cumprida por cada vértice! (Um dos adjacentes $\in S$.)



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

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

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

- **DS** $\in \mathcal{NP}$, pois existe um algoritmo determinístico e polinomial que, dado um conjunto de vértices S de um grafo, verifica, em tempo linear, se eles são um conjunto dominante **checando**, para **cada vértice** v, se $v \in S$ ou se entre os adjacentes de v 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 e polinomial que, dado um conjunto de vértices S de um grafo, verifica, em tempo linear, se eles são um conjunto dominante **checando**, para **cada vértice** v, se $v \in S$ ou se entre os adjacentes de v 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.

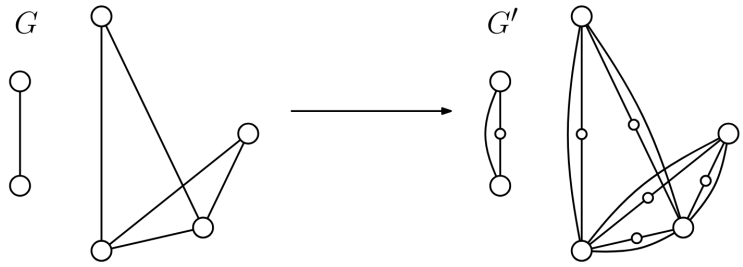
Vértices de G':

- 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':

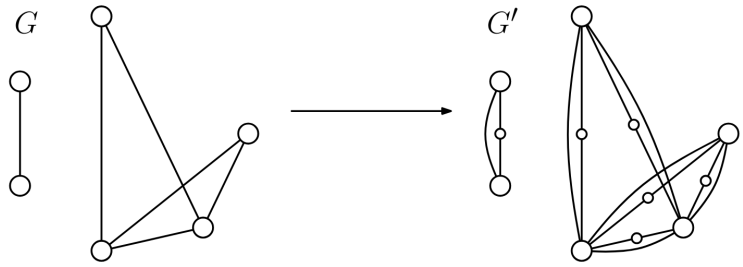
- Toda aresta de G também é aresta de G'.
  Além disso, para cada vértice $v_{ij} \in V_A$ criaremos duas arestas: (v_{ij}, i) e (v_{ij}, j).
- Se E_A é o conjunto das arestas criadas desta forma, $E' = E \cup E_A$.

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**

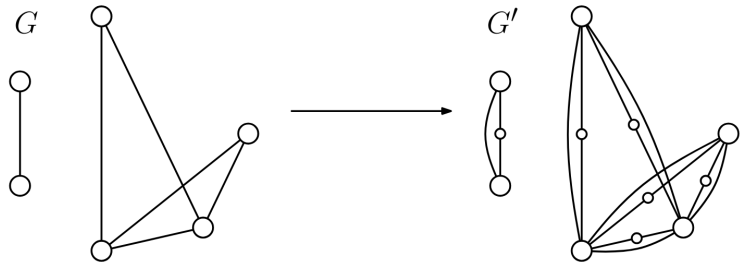


Se $|V| = n$ e $|E| = m$, a instância de entrada de **DS** é obtida em $O(n + m)$.

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'

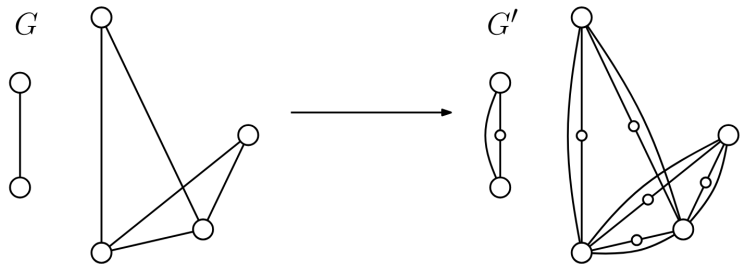
Provas de $\mathcal{NP}$-completude: **DS**



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.

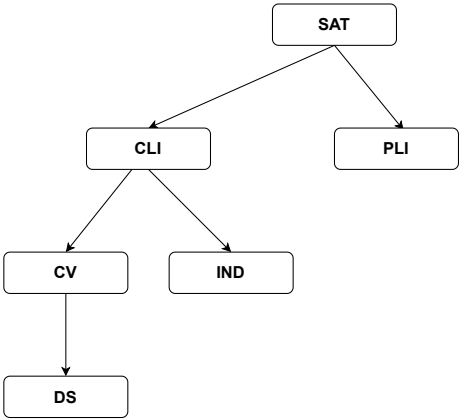
Provas de $\mathcal{NP}$-completude: **DS**



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: **SC**

Problema da Cobertura de Conjuntos (Set Cover, SC)
Versão de decisão

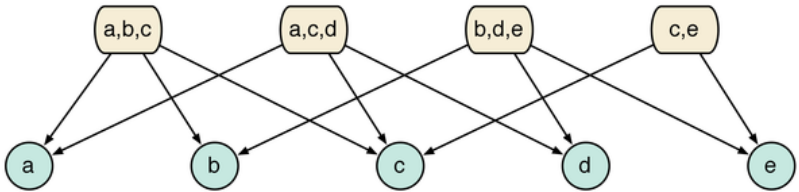
Dado um conjunto U, uma coleção $S_1, \ldots, S_m$ de m subconjuntos de U, e um inteiro k, decidir se existe uma coleção de k subconjuntos cuja união é U.

Teorema

SC é $\mathcal{NP}$-completo.

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

Exemplo de instância do Set Cover:



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

- **SC** $\in \mathcal{NP}$, pois existe um algoritmo determinístico e polinomial que, dada uma coleção de k subconjuntos, verifica se eles cobrem U checando se cada elemento $x \in U$ está em um desses k subconjuntos. ☑

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

- **SC** $\in \mathcal{NP}$, pois existe um algoritmo determinístico e polinomial que, dada uma coleção de k subconjuntos, verifica se eles cobrem U checando se cada elemento $x \in U$ está em um desses k subconjuntos. ☑
- **SC** $\in \mathcal{NP}$-difícil.
  - Precisamos escolher um problema do qual faremos a redução.

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

- **SC** $\in \mathcal{NP}$, pois existe um algoritmo determinístico e polinomial que, dada uma coleção de k subconjuntos, verifica se eles cobrem U checando se cada elemento $x \in U$ está em um desses k subconjuntos. ☑
- **SC** $\in \mathcal{NP}$-difícil.
  - Precisamos escolher um problema do qual faremos a redução.

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

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

Transformação da instância do **CV** em uma instância **SC**.
Seja $G = (V, E)$, k uma instância do **CV**. Construa uma instância do **SC** formada por:

- $U = E$;
- Para cada vértice v_i, um subconjunto S_i de todas as arestas incidentes no v_i;
- O inteiro k.

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

Transformação da instância do **CV** em uma instância **SC**.
Seja $G = (V, E)$, k uma instância do **CV**. Construa uma instância do **SC** formada por:

- $U = E$;
- Para cada vértice v_i, um subconjunto S_i de todas as arestas incidentes no v_i;
- O inteiro k.

A redução é **polinomial**.

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

A redução é **válida**? Ou seja:

$$G,\ k \text{ é } \texttt{SIM} \text{ para } \textbf{CV} \Leftrightarrow U,\ S_1, \ldots, S_m,\ k \text{ é } \texttt{SIM} \text{ para } \textbf{SC}?$$

- G, k é SIM para **CV**
  $\Rightarrow \exists\ C \subseteq V$, $|C| = k$, tal que $\forall e \in E$, e tem ao menos um extremo em C
  $\Rightarrow$ Para toda aresta e, existe um vértice $v_i \in C$, extremo da aresta
  $\Rightarrow$ Para cada $e \in U$, existe um S_i tal que $e \in S_i$
  $\Rightarrow$ Seja C' a coleção dos subconjuntos correspondentes aos vértices em C
  $\Rightarrow$ A coleção C' é um certificado válido, e U, $S_1, \ldots, S_m$, k é SIM para **SC**.

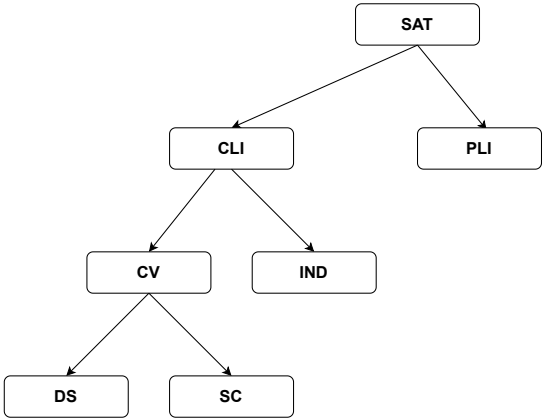
Provas de $\mathcal{NP}$-completude: **SC**

A redução é **válida**? Ou seja:

$$G,\ k \text{ é SIM para } \mathbf{CV} \Leftrightarrow U,\ S_1, \ldots, S_m,\ k \text{ é SIM para } \mathbf{SC}?$$

- $U,\ S_1, \ldots, S_m,\ k$ é SIM para **SC**
  $\Rightarrow$ Existe uma coleção C', $|C'| = k$, que é certificado válido para **SC**
  $\Rightarrow$ Para cada $e \in U$, existe um S_i tal que $e \in S_i$, $S_i \in C'$
  $\Rightarrow$ Para cada $e \in E$, existe $v_i \in V$ extremo de e, seja C o conjunto destes vértices
  $\Rightarrow \exists\ C \subseteq V$, tal que $\forall e \in E$, e tem ao menos um extremo em C
  $\Rightarrow G,\ k$ é SIM para **CV**.

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?

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?

  Sim, pois **SAT** $\propto_{\text{poli}}$ **3SAT**.
  - Esse fato não parece nada obvio...

$\rightarrow$ Estudar. 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 não temos $x_i \cdot \overline{x_i}$:
    $x_i = 1$ para toda x_i não negada, e $x_i = 0$ para toda x_i negada.

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 não temos $x_i \cdot \overline{x_i}$:
    $x_i = 1$ para toda x_i não negada, e $x_i = 0$ para toda x_i negada.
- **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 não temos $x_i \cdot \overline{x_i}$:
    $x_i = 1$ para toda x_i não negada, e $x_i = 0$ para toda x_i negada.
- **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: **3SAT**

Resumo:

- **1SAT** $\in \mathcal{P}$.
- **2SAT** $\in \mathcal{P}$.
- **3SAT** $\in \mathcal{NP}$-completo.

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

Resumo:

- **1SAT** $\in \mathcal{P}$.
- **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**.

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.

Observação: Existe o problema da 2-coloração (**2COL**), 3-coloração (**3COL**), quando fixamos o k em 2 e 3, e o problema geral da k-coloração.

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

Surpreendentemente, temos de novo:

- **1COL** $\in \mathcal{P}$.

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

Surpreendentemente, temos de novo:

- **1COL** $\in \mathcal{P}$.
- **2COL** $\in \mathcal{P}$. (Que problema já visto é esse?)

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

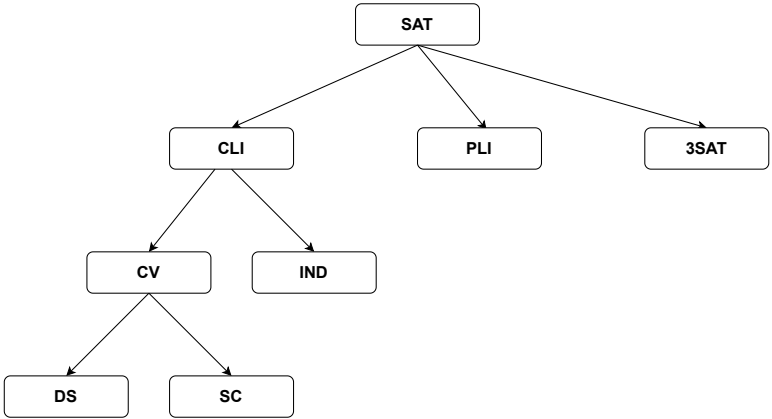
Surpreendentemente, temos de novo:

- **1COL** $\in \mathcal{P}$.
- **2COL** $\in \mathcal{P}$. (Que problema já visto é esse?)
- **3COL** $\in \mathcal{NP}$-completo, pois **3SAT** $\propto_{\text{poli}}$ **3COL**.

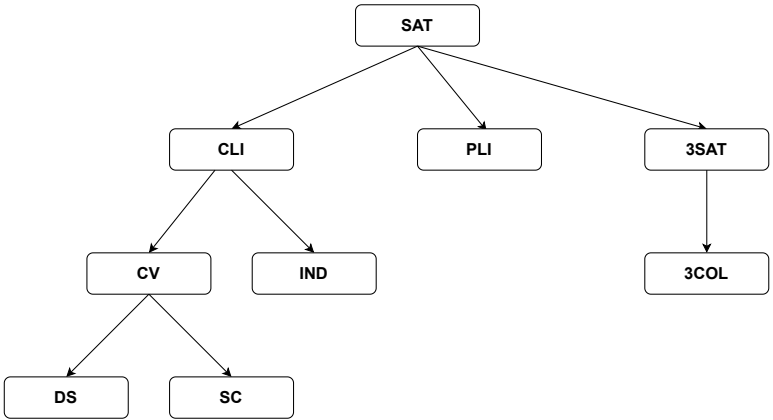
Ver **3SAT** $\propto_{\text{poli}}$ **3COL** no Capítulo 11.4 do Manber.

Portanto, o problema geral da k-coloração **(kCOL)** é $\mathcal{NP}$-completo.

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



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



Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

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

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

- **Ciclo Hamiltoniano** (**HCP**):

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

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

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

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

- **Ciclo Hamiltoniano** (**HCP**):

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

Consideremos a sequência de reduções **3SAT** $\propto_{\text{poli}}$ **HCP** $\propto_{\text{poli}}$ **HPP**.
O que ela mostra?

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

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

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

- **Ciclo Hamiltoniano** (**HCP**):

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

Consideremos a sequência de reduções **3SAT** $\propto_{\text{poli}}$ **HCP** $\propto_{\text{poli}}$ **HPP**.
O que ela mostra? – Que esses dois problemas são $\mathcal{NP}$-completos.

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

Sequência de reduções **3SAT** $\propto_{\text{poli}}$ **HCP** $\propto_{\text{poli}}$ **HPP**.

Omitiremos a redução **3SAT** $\propto_{\text{poli}}$ **HCP**. Vamos provar que **HCP** $\propto_{\text{poli}}$ **HPP**.

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

Sequência de reduções **3SAT** $\propto_{\text{poli}}$ **HCP** $\propto_{\text{poli}}$ **HPP**.

Omitiremos a redução **3SAT** $\propto_{\text{poli}}$ **HCP**. Vamos provar que **HCP** $\propto_{\text{poli}}$ **HPP**.

HPP é $\mathcal{NP}$-completo:

- **HPP** $\in \mathcal{NP}$, pois existe um algoritmo determinístico e polinomial que, dada uma sequência de vértices P (um caminho Hamiltoniano), verifica se P é uma solução checando se todos os vértices estão em P, e se as arestas do caminho existem. ☑

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

Sequência de reduções **3SAT** $\propto_{\text{poli}}$ **HCP** $\propto_{\text{poli}}$ **HPP**.

Omitiremos a redução **3SAT** $\propto_{\text{poli}}$ **HCP**. Vamos provar que **HCP** $\propto_{\text{poli}}$ **HPP**.

HPP é $\mathcal{NP}$-completo:

- **HPP** $\in \mathcal{NP}$, pois existe um algoritmo determinístico e polinomial que, dada uma sequência de vértices P (um caminho Hamiltoniano), verifica se P é uma solução checando se todos os vértices estão em P, e se as arestas do caminho existem. ☑
- **HPP** $\in \mathcal{NP}$-difícil: **HCP** $\propto_{\text{poli}}$ **HPP**.

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

Transformação da instância do **HCP** em uma instância **HPP**.
Seja $G = (V, E)$ uma instância do **HCP**. Construa uma instância do **HPP**, um grafo direcionado $G' = (V', E')$, da seguinte forma:

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

Transformação da instância do **HCP** em uma instância **HPP**.
Seja $G = (V, E)$ uma instância do **HCP**. Construa uma instância do **HPP**, um grafo direcionado $G' = (V', E')$, da seguinte forma:

- Seja $v \in V$ um vértice arbitrário de G. Em V' teremos todos os vértices de V, mas em vez do vértice v, teremos dois vértices, v' e v''.

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

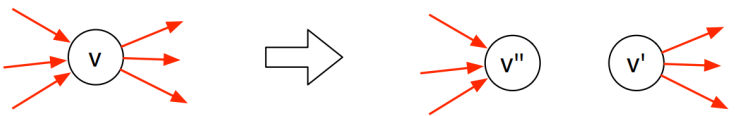
Transformação da instância do **HCP** em uma instância **HPP**.
Seja $G = (V, E)$ uma instância do **HCP**. Construa uma instância do **HPP**, um grafo direcionado $G' = (V', E')$, da seguinte forma:

- Seja $v \in V$ um vértice arbitrário de G. Em V' teremos todos os vértices de V, mas em vez do vértice v, teremos dois vértices, v' e v''.
- Em E' teremos todos os arcos de E, mas aqueles arcos que saem de v sairão de v', e aqueles que entram em v entrarão em v''.

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

Transformação da instância do **HCP** em uma instância **HPP**.
Seja $G = (V, E)$ uma instância do **HCP**. Construa uma instância do **HPP**, um grafo direcionado $G' = (V', E')$, da seguinte forma:

- Seja $v \in V$ um vértice arbitrário de G. Em V' teremos todos os vértices de V, mas em vez do vértice v, teremos dois vértices, v' e v''.
- Em E' teremos todos os arcos de E, mas aqueles arcos que saem de v sairão de v', e aqueles que entram em v entrarão em v''.



A redução é **polinomial**.

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

A redução é **válida**? Ou seja:

G é SIM para **HCP** $\Leftrightarrow$ G' é SIM para **HPP**?

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

A redução é **válida**? Ou seja:

$$G \text{ é } \texttt{SIM} \text{ para } \textbf{HCP} \Leftrightarrow G' \text{ é } \texttt{SIM} \text{ para } \textbf{HPP}?$$

- G é `SIM` para **HCP** $\Rightarrow$ Temos um ciclo Hamiltoniano em G, portanto podemos usar os mesmos arcos do ciclo para encontrar um caminho Hamiltoniano no grafo G', iniciando o caminho em v' e terminando em v''.

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

A redução é **válida**? Ou seja:

$$G \text{ é } \texttt{SIM} \text{ para } \textbf{HCP} \Leftrightarrow G' \text{ é } \texttt{SIM} \text{ para } \textbf{HPP}?$$

- G é `SIM` para **HCP** $\Rightarrow$ Temos um ciclo Hamiltoniano em G, portanto podemos usar os mesmos arcos do ciclo para encontrar um caminho Hamiltoniano no grafo G', iniciando o caminho em v' e terminando em v''.
- G' é `SIM` para **HPP** $\Rightarrow$ Temos um caminho Hamiltoniano P em G', e este caminho obrigatoriamente começa em v' e termina em v''. (Por quê?) Para encontrar um ciclo Hamiltoniano em G, podemos usar os mesmos arcos em P, substituindo v' e v'' por v.

Provas de $\mathcal{NP}$-completude: **HPP** e **HCP**

Esclarecendo:

Os problemas **HPP** e **HCP** são $\mathcal{NP}$-completos em grafos direcionados e não direcionados.

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

- Problema do **Caixeiro-Viajante** (**TSP**) – Dezenas de livros, artigos, filmes ...

  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 $n!$.)

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

- Problema do **Caixeiro-Viajante** (**TSP**) – Dezenas de livros, artigos, filmes ...

  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.*

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

- Problema do **Caixeiro-Viajante** (**TSP**) – Dezenas de livros, artigos, filmes ...

  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**:

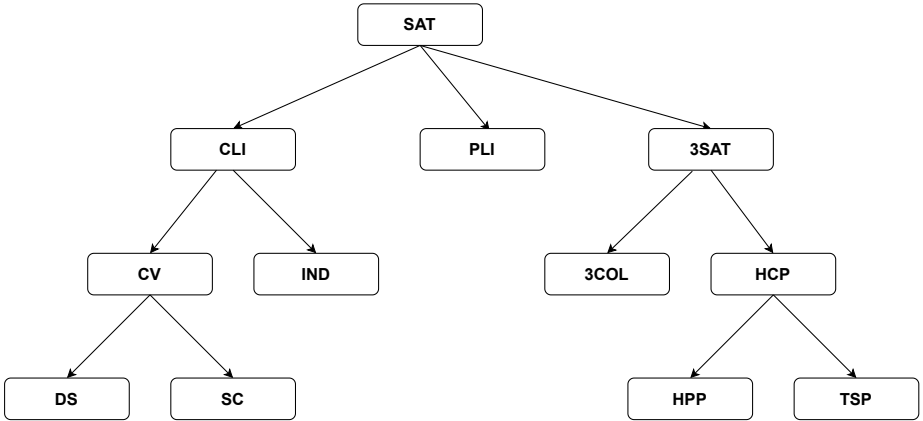
  > *Dado um grafo não-direcionado e ponderado $G = (V, E)$, com custos d_e nas arestas, e um valor D, decidir se G tem um ciclo hamiltoniano de custo $\leq D$.*

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

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

Consegue imaginar essa redução?

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



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 **Escolha**s 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

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?