Universidade de São Paulo
Instituto de Matemática e Estatística
Aluna: Nara Avila Moraes
Nro. USP: 5716734

2o. semestre, 2025
Exercício 02 - 31/08

MAC5722/IME: Complexidade Computacional.
Prof.: Benjamin Merlin Bumpus

Arora, Barak - Capítulo 02:

(2.10) Suppose $L_1, L_2 \in \mathbf{NP}$. Then is $L_1 \cup L_2$ in **NP**? What about $L_1 \cap L_2$?

(2.18) Prove that the language HAMPATH of *undirected* graphs with Hamiltonian paths is **NP**-complete. Prove that the language TSP described in Example 2.3 is **NP**-complete. Prove that the language HAMCYCLE of *undirected* graphs that contain Hamiltonian cycle (a simple cycle involving all the vertices) is **NP**-complete.

(2.21) Prove that 3COL (see Exercise 2.2) is **NP**-complete.

Resposta para o Ítem 2.10

Este exercício trabalha as propriedades de fechamento da classe **NP**. Após demonstrar os ítens pedidos, podemos concluir que a classe **NP** é fechada sob as operações de união e interseção.

Vamos desenvolver a prova para cada um dos casos utilizando a definição formal de **NP**: uma linguagem L está em **NP** se existe um *verificador* de tempo polinomial M que, para uma entrada $x \in L$, pode verificar uma prova (ou certificado) u **de tamanho polinomial** em tempo polinomial.

União ($L_1 \cup L_2$):

Para demonstrar que se $L_1 \in \mathbf{NP}$ e $L_2 \in \mathbf{NP}$, então $L_1 \cup L_2 \in \mathbf{NP}$, precisamos construir um verificador de tempo polinomial, que chamaremos de M_{uniao}, para a linguagem $L_{\text{uniao}} = L_1 \cup L_2$.

Hipóteses

Como L_1 e L_2 estão em **NP**, sabemos que:

1. Existe um verificador M_1 e um polinômio p_1 tais que para qualquer entrada x:

$$x \in L_1 \iff \exists u_1 \text{ com } |u_1| \leq p_1(|x|) \text{ tal que } M_1(x, u_1) \text{ aceita.}$$

2. Existe um verificador M_2 e um polinômio p_2 tais que para qualquer entrada x:

$$x \in L_2 \iff \exists u_2 \text{ com } |u_2| \leq p_2(|x|) \text{ tal que } M_2(x, u_2) \text{ aceita.}$$

Construção do Novo Verificador (M_{uniao})

Para uma entrada x estar na união, ela precisa estar em L_1 **OU** em L_2. O novo certificado, u_{uniao}, precisa indicar a qual linguagem x pertence e fornecer a prova correspondente.

Podemos definir o novo certificado como um par $u_{\text{uniao}} = \langle b, u \rangle$, onde:

- b é um bit que indica a linguagem (e.g., $b = 1$ para L_1, $b = 2$ para L_2).
- u é o certificado original para a respectiva linguagem (u_1 ou u_2).

O algoritmo para $M_{\text{uniao}}(x, \langle b, u \rangle)$ é:

1. Se $b = 1$, execute $M_1(x, u)$. Se M_1 aceitar, aceite. Caso contrário, rejeite.
2. Se $b = 2$, execute $M_2(x, u)$. Se M_2 aceitar, aceite. Caso contrário, rejeite.

Verificação e Análise

- **Existência:** Se $x \in L_1 \cup L_2$, então x está em L_1 ou L_2. Se $x \in L_1$, o certificado $\langle 1, u_1 \rangle$ existe e será aceito. Se $x \in L_2$, o certificado $\langle 2, u_2 \rangle$ existe e será aceito. Reciprocamente, se M_{uniao} aceita um certificado $\langle b, u \rangle$, então $x \in L_b$, e portanto $x \in L_1 \cup L_2$.
- **Eficiência:** O tamanho de u_{uniao} é polinomial, pois o tamanho de u é polinomial. O tempo de execução de M_{uniao} é o tempo de execução de M_1 ou M_2, que são ambos polinomiais.

Portanto demonstramos que $L_1 \cup L_2 \in \mathbf{NP}$.

Interseção ($L_1 \cap L_2$):

Para demonstrar que se $L_1 \in \mathbf{NP}$ e $L_2 \in \mathbf{NP}$, então $L_1 \cap L_2 \in \mathbf{NP}$ vamos aplicar uma construção análoga. Precisamos construir um verificador M_{inter} para a linguagem $L_{\text{inter}} = L_1 \cap L_2$.

Construção do Novo Verificador (M_{inter})

Para uma entrada x estar na interseção, ela precisa estar em L_1 **E** em L_2. Portanto, o certificado precisa conter a prova para ambas as linguagens.

Podemos definimos o novo certificado u_{inter} como um par dos dois certificados originais: $u_{\text{inter}} = \langle u_1, u_2 \rangle$.

O algoritmo para $M_{\text{inter}}(x, \langle u_1, u_2 \rangle)$ é:

1. Execute $M_1(x, u_1)$.
2. Execute $M_2(x, u_2)$.
3. Se **ambos** aceitarem, aceite. Caso contrário, rejeite.

Verificação e Análise

- **Existência:** Se $x \in L_1 \cap L_2$, então $x \in L_1$ e $x \in L_2$. Logo, ambos os certificados u_1 e u_2 existem. O certificado composto $\langle u_1, u_2 \rangle$ fará com que M_{inter} aceite. Reciprocamente, se M_{inter} aceita $\langle u_1, u_2 \rangle$, significa que ambos M_1 e M_2 aceitaram, implicando que $x \in L_1$ e $x \in L_2$.
- **Eficiência:** O tamanho de u_{inter} é a soma dos tamanhos de u_1 e u_2, que continua sendo polinomial. O tempo de execução de M_{inter} é a soma dos tempos de M_1 e M_2, que também é polinomial.

Portanto também está demonstrado que $L_1 \cap L_2 \in \mathbf{NP}$.

Resposta para o Ítem 2.18

A prova de que um problema é **NP-completo** exige a demonstração de duas propriedades fundamentais. Para cada problema, precisamos provar que:

1. **O problema está em NP:** É suficiente mostrar que, dado um certificado, é possível verificar se ele é válido, em tempo polinomial.
2. **O problema é NP-difícil:** É necessário mostrar que o problema é pelo menos tão difícil quanto algum problema NP-completo conhecido. Para isso escolhemos um problema **NP-completo** e o transformamos através de uma **redução de tempo polinomial** no problema em questão.

1 HAMCYCLE (Ciclo Hamiltoniano)

Problema: Dado um grafo G, existe um ciclo simples que visita **todos** os vértices de G exatamente uma vez?

1.1 Passo 1: hamcycle está em NP

- **Certificado:** Uma sequência de n vértices, $(v_1, v_2, \ldots, v_n)$.
- **Verificador (em tempo polinomial):**
  1. Verifique se a sequência contém todos os n vértices do grafo, sem repetições. (Pode ser feito em tempo $O(n \log n)$ com ordenação ou $O(n)$ com uma tabela hash).
  2. Verifique se existe uma aresta entre v_i e v_{i+1} para todo i de 1 a $n-1$. (Leva $O(n)$ checagens na matriz ou lista de adjacência).
  3. Verifique se existe uma aresta entre o último vértice, v_n, e o primeiro, v_1, para fechar o ciclo. (Uma checagem).

Como a verificação é eficiente, concluímos que **hamcycle** $\in$ **NP**.

1.2 Passo 2: hamcycle é NP-difícil (Redução de 3-sat)

Esta é uma das reduções mais famosas e engenhosas. Mostramos que qualquer instância da fórmula 3-SAT (que sabemos ser NP-completa) pode ser convertida em um grafo G tal que a fórmula é satisfatível **se, e somente se**, G tiver um ciclo Hamiltoniano. A construção dos "gadgets" funciona da seguinte forma:

- **Gadgets de Variável:** Para cada variável x_i, criamos uma estrutura com dois caminhos paralelos. Um ciclo Hamiltoniano que passar por este gadget é forçado a escolher um dos caminhos, o que corresponde a atribuir `TRUE` ou `FALSE` para x_i.
- **Gadgets de Cláusula:** Para cada cláusula C_j, criamos um vértice.
- **Conexões:** Conectamos os gadgets de variável aos de cláusula de uma forma inteligente. Se a variável x_i aparece na cláusula C_j, conectamos o caminho "TRUE" de x_i ao vértice da cláusula C_j. Se $\neg x_i$ aparece, conectamos o caminho "FALSE". As conexões permitem que o ciclo faça um "desvio" para visitar o vértice da cláusula somente se a atribuição de verdade escolhida para as variáveis satisfizer aquela cláusula.

A lógica é que um ciclo Hamiltoniano no grafo construído corresponde a uma atribuição de verdade que satisfaz todas as cláusulas. Se não houver uma atribuição satisfatível, é impossível visitar todos os vértices em um único ciclo. Como essa transformação pode ser feita em tempo polinomial, **hamcycle é NP-difícil**.

Conclusão: Como está em **NP** e é **NP**-difícil, **hamcycle é NP-completo**.

2 hampath (Caminho Hamiltoniano)

Problema: Dado um grafo G, existe um caminho simples que visita **todos** os vértices de G exatamente uma vez?

2.1 Passo 1: hampath está em NP

A prova é quase idêntica à de HAMCYCLE. O certificado é uma sequência de vértices, e o verificador checa se todos os vértices estão presentes e se as conexões existem. A única diferença é que ele não precisa checar a aresta de volta do final para o começo. Claramente polinomial. **hampath** $\in$ **NP**.

2.2 Passo 2: hampath é NP-difícil (Redução de hamcycle)

Mostramos que se pudéssemos resolver HAMPATH rapidamente, poderíamos resolver HAMCYCLE rapidamente.

- **Entrada:** Um grafo G e a pergunta "G tem um ciclo Hamiltoniano?".
- **Construção:** Escolha um vértice qualquer v em G. "Quebre" v em dois novos vértices, v_{in} e v_{out}, em um novo grafo G'. Para cada aresta (u, v) que chegava em v, crie uma aresta (u, v_{in}) em G'. Para cada aresta (v, w) que saía de v, crie uma aresta (v_{out}, w) em G'.
- **A Lógica:** Um ciclo Hamiltoniano em G que passava por v (ex: $\cdots \to u \to v \to w \to \ldots$) corresponde diretamente a um caminho Hamiltoniano em G' que vai de v_{out} até v_{in} (ex: $v_{out} \to w \to \cdots \to u \to v_{in}$).

Portanto, G tem um ciclo Hamiltoniano se, e somente se, G' tem um caminho Hamiltoniano. A redução é polinomial. **hampath é NP-difícil**.

Conclusão: hampath é NP-completo.

3 tsp (Problema do Caixeiro Viajante – Versão de Decisão)

Problema: Dadas n cidades, as distâncias entre elas, e um orçamento k, existe um tour (um ciclo) que visita todas as cidades com uma distância total $\leq k$?

3.1 Passo 1: tsp está em NP

- **Certificado:** Uma permutação das n cidades, representando a ordem do tour.
- **Verificador (em tempo polinomial):**
  1. Verifique se a permutação é válida (contém todas as cidades, uma vez cada).
  2. Some as distâncias entre as cidades consecutivas na permutação (incluindo a volta da última para a primeira).
  3. Verifique se a soma total é menor ou igual ao orçamento k.

Como a verificação é eficiente, **tsp** $\in$ **NP**.

3.2 Passo 2: tsp é NP-difícil (Redução de hamcycle)

Mostramos que se pudéssemos resolver TSP rapidamente, poderíamos resolver HAMCYCLE rapidamente.

- **Entrada:** Um grafo $G = (V, E)$ e a pergunta "G tem um ciclo Hamiltoniano?".
- **Construção (transformando G em uma instância de tsp):**
  1. **Cidades:** Crie uma cidade para cada vértice v de V.

2. **Distâncias:** Para cada par de cidades u e v:
   - Se a aresta $\{u, v\}$ **existe** em E, a distância é **1**.
   - Se a aresta $\{u, v\}$ **não existe** em E, a distância é **2**.
3. **Orçamento k:** Defina $k = n$ (onde $n = |V|$).

- **A Lógica:**
  - Se G tem um ciclo Hamiltoniano, então existe um tour de n cidades usando apenas arestas que estavam em G. Cada passo desse tour terá distância 1. O custo total será $n \times 1 = n$. Este tour satisfaz o orçamento $k = n$.
  - Se existe um tour de TSP com custo total $\leq n$, então esse tour (que tem n passos) deve usar apenas arestas de custo 1, pois se usasse uma única aresta de custo 2, o custo total seria $> n$. Um tour que usa apenas arestas de custo 1 corresponde diretamente a um ciclo Hamiltoniano no grafo original G.

A redução é polinomial. **tsp é NP-difícil**.

Conclusão: tsp é NP-completo.

Resposta para o Ítem 2.21
