EXERCÍCIOS

Obs.: Para os exercícios em que se pede para provar que um certo problema Π é NP-completo, você deve fazer esta demonstração fazendo cada um dos pontos abaixo:

A. Prove que Π pertence a NP.

B. Prove que Π é NP-difícil:

B.1. Escolha um problema L que já provado ser NP-Completo. Descreva o problema L e justifique que ele já foi provado ser NP-completo (e.g., provamos a demonstração que é NP-completo em aula).

B.2. Faça uma redução de tempo polinomial de $L \leq_P \Pi$:

B.2.i. Existe algoritmo T que transforma cada entrada $I_L \in \mathcal{I}(L)$ em entrada uma entrada $I_\Pi \in \mathcal{I}(\Pi)$. Apresente/descreva este algoritmo.

B.2.ii. Mostre que o algoritmo T executa em tempo polinomial. Prove/justifique isso.

B.2.iii. Prove que I_L tem resposta sim se e someente I_Π tem resposta sim.

Isto é, sua demonstração deve necessariamente conter descrição e/ou a demonstração quando for o caso, para os itens **A.**, **B.1.**, **B.2.i.**, **B.2.ii.**, **B.2.iii.**. Nos casos em que a demonstração é muito simples e direta, pode só explicar/demonstrar de maneira simplificada também. Mas neste caso, estamos assumindo que você poderá fazer a demonstração formal com detalhes, se necessário.

Exercícios de NP-completude, NP-difícil,...

1. Suponha que L é uma linguagem em NP tal que para todo $x \in L$, existe um certificado y_x de x que usa $\lceil \log(|x|) \rceil$ bits para representá-lo. Mostre que esta linguagem pertence a classe P.

2. Dê uma codificação formal de um grafo dirigido (as arestas tem orientação) como strings binárias usando uma representação de matriz de adjacência. Faça o mesmo usando uma representação em lista de adjacência. Justifique que as duas representações estão polinomialmente relacionadas.

3. Suponha que uma linguagem L possa aceitar qualquer string $x \in L$ em tempo polinomial, mas que o algoritmo que decide isso executa em tempo superpolinomial se $x \notin L$. Justifique que L pode ser decidido em tempo polinomial.

4. Mostre que um algoritmo que faz um número constante de chamadas a subrotinas de tempo polinomial tem também tempo polinomial. Mostre também que se um algoritmo faz um número polinomial de chamadas a subrotinas polinomiais, podemos obter um algoritmo de tempo exponencial (Sug.: considere um algoritmo que faz n chamadas de uma rotina, cada chamada altera uma certa estrutura).

5. Uma linguagem L é dita ser *completa* para uma classe de linguagens C, com respeito a reduções polinomiais se $L \in C$ e $L' \leq_P L$ para todo $L' \in C$. Mostre que $\emptyset$ e $\{0,1\}^*$ são as duas únicas linguagens em P que não são completas para P, com respeito a reduções de tempo polinomial.

6. Mostre que qualquer linguagem em NP pode ser decidida por um algoritmo que executa em tempo $2^{O(n^k)}$ para algum k.

7. Para cada uma variante ou caso especial dos problemas relativos ao ciclo ou caminho hamiltoniano, mais abaixo, responda se cada problema pertence à classe P ou à classe NP-Completo. Caso esteja em P, você deve apresentar o algoritmo que decide o problema em tempo polinomial e provar sua complexidade de tempo. Caso seja NP-completo, você deve provar esta afirmação.

- Dado grafo simples, bipartido, não-dirigido $G = (X, Y, A)$, onde o conjunto de vértices é a partiçao X, Y, decidir se G possui um ciclo hamiltoniano.
- Dado grafo simples e dirigido $G = (N, A)$, decidir se G possui um ciclo hamiltoniano.
- Dado grafo simples e dirigido $G = (N, A)$, pesos inteiros nos arcos, w_a, para todo $a \in A$, e valor inteiro K, decidir se G possui um ciclo dirigido com peso total de suas arestas no máximo K. Neste caso, o ciclo não necessariamente é hamiltoniano.
- Dado grafo simples e não-dirigido $G = (V, E)$, decidir se G tem um caminho hamiltoniano.
- Dado grafo simples e não-dirigido $G = (V, E)$ e dois vértices distintos $s, t \in V$, decidir se G tem um caminho hamiltoniano onde as extremidades são s e t.
- Dado grafo simples e dirigido $G = (N, A)$, decidir se G tem um caminho hamiltoniano dirigido.
- Dado grafo simples e dirigido $G = (N, A)$ e dois vértices distintos $s, t \in V$, decidir se G tem um caminho hamiltoniano (dirigido) começando em s e terminando em t.
- Dado grafo simples, acíclico e dirigido $G = (N, A)$, decidir se G tem um caminho hamiltoniano dirigido.
- Dado grafo simples, acíclico e dirigido $G = (N, A)$ e dois vértices distintos $s, t \in V$, decidir se G tem um caminho hamiltoniano (dirigido) começando em s e terminando em t.
- Dado grafo simples e dirigido $G = (N, A)$, tal que para todo par de vértices distintos u, v, há exatamente um arco, ou o arco (u, v) ou o arco (v, u). Decidir se G tem um caminho hamiltoniano dirigido.
- Dado grafo simples e dirigido $G = (N, A)$, pesos nos arcos w_a para todo $a \in A$, e valor inteiro K, decidir se G possui um caminho halmiltoniano dirigido com peso total de suas arestas no máximo K.

8. O problema de decidir se um dado grafo $G = (V, E)$ tem uma árvore geradora é um problema em P. Mas e o problema de decidir se um grafo tem uma árvore geradora onde todos os vértices da árvore tem grau máximo 3. Este problema está em P ou está em NP-completo ? Prove sua afirmação. Obs.: E se no lugar de 3, colocarmos um inteiro constante K?

9. Um grafo não-dirigido $G = (V, E)$ é dito ser *3-Ciclo-Particionável* se existirem três circuitos (que não repetem vértices), digamos C_1, C_2 e C_3, sem vértices em comum e todo vértice de G está em exatamente um destes circuitos. Mostre que o problema de decidir se um grafo é 3-Ciclo-Particionável é um problema NP-completo.

10. Você acha que o seguinte problema é polinomial ou NP-completo ? Justifique.
Instância: grafo simples não-dirigido G.
Pergunta: existe um circuito simples (sem repetir vértices) em G que passa por pelo menos metade dos vértices de G ? E se no lugar de 'pelo menos metade' colocássemos 'pelo menos um terço'?

11. Dois grafos $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$ são isomorfos se existe uma bijeção $f : V_1 \to V_2$ tal que $(u, v) \in E_1$ se e somente se a aresta $(f(u), f(v)) \in E_2$.
Considere uma linguagem SUBISO $= \{\langle G_1, G_2 \rangle : G_1$ é isomorfo a um subgrafo de $G_2\}$. Prove que SUBISO $\in$NP-completo.

12. Considere o *Problema da Partição:* Dado um conjunto de inteiros $S = \{s_1, s_2, \ldots, s_n\}$, verificar se existe um subconjunto $S' \subset S$ tal que S' tem peso total igual a metade do peso total de S. Mostre que o problema da partição é NP-completo. Você pode usar supor resultados vistos em sala de aula.

13. O problema do *bin-packing* consiste no seguinte: São dados recipientes de capacidades C e n objetos. Cada objeto i tem peso p_i. O problema consiste em empacotar os objetos nos recipientes tal que a soma dos pesos dos objetos empacotados em um recipiente não ultrapasse a sua capacidade C. O objetivo é minimizar o número de recipientes usados para empacotar os n objetos. Escreva a versão de decisão deste problema e mostre que a versão de decisão é NP-completa.

14. O problema *Escalonamento de Tarefas em Máquinas (Homogêneas)* consiste em: Dados uma lista de n tarefas $T = \{1, \ldots, n\}$, cada tarefa $i \in T$ com um valor t_i (referente a tempo) e m máquinas $M = \{1, \ldots, m\}$. Cada tarefa deve ser alocada (escalonada) em exatamente uma das máquinas. Caso a tarefa $i \in T$ seja alocada na máquina $j \in M$, seu tempo de execução será de t_i segundos. Após a alocação das tarefas, o tempo total de execução de uma máquina é a soma dos tempos de execução das tarefas alocadas a ela. O *makespan* de um escalonamento é definido como o maior tempo de execução de uma máquina, considerando todas as máquinas disponíveis. Neste problema, buscamos por um escalonamento das tarefas com o menor makespan possível. Descreva a versão correspondente de decisão deste problema (da maneira natural, como fizemos em aula para transformar um problema de otimização em decisão) e mostre que o correspondente problema está em NP-completo.

15. João é dono de uma empresa que possui 5 funcionários. A empresa tem um conjunto de atividades A, cada uma com um tempo t_i de dedicação. Cada atividade deve ser feita por apenas um funcionário. João precisa atribuir cada atividade de A para um dos funcionários. Dada uma atribuição T, de atividades para os funcionários, denote por T_j o tempo de dedicação total do funcionário j de acordo com a atribuição T. Pensando em balancear a carga de trabalho entre seus funcionários, João quer encontrar uma atribuição que minimize $(\max_j T_j - \min_j T_j)$. O problema de atribuição do João é NP-difícil ou é um problema que pode ser resolvido em tempo polinomial? Em ambos os casos, prove sua resposta.

16. Considere o *Problema do Caminho Mínimo Par:* Dado um grafo $G = (V, A)$, com pesos w_a em cada aresta $a \in A$ (o peso de cada aresta pode ser positivo ou negativo) e dois vértices, s e t, o problema do *caminho mínimo par* consiste em encontrar um camimho de s a t, com número par de arestas e com peso total mínimo. Para este problema e possível variante/caso especial, mais abaixo, responda se o problema pertence à classe P ou à classe NP-Completo. Caso esteja em P, você deve apresentar um algoritmo que decide o problema em tempo polinomial (tente na melhor complexidade que puder) e provar sua complexidade de tempo. Caso seja NP-completo, você deve provar esta afirmação.

    - Problema do Caminho Mínimo Par (é o problema descrito no enunciado).
    - Problema do Caminho Mínimo Par em Grafos Não-Direcionados Acíclicos.
    - Problema do Caminho Mínimo Par em Grafos Não-Direcionados Bipartidos.
    - Problema do Caminho Mínimo Par em Grafos Direcionados Acíclicos.
    - Problema do Caminho Mínimo Par em Grafos Direcionados Bipartidos.

17. O problema da cobertura por conjuntos é o seguinte:
    Set-Cover: Dado família de conjuntos $\mathcal{S} = \{S_1, \ldots, S_m\}$ de um conjunto $E = \{e_1, \ldots, e_n\}$, tal que $E = \cup_{i=1}^{m} S_i$, e dado um valor inteiro k, decidir se existe um subconjunto $\mathcal{S}' \subseteq \mathcal{S}$ que cobre E e $|\mathcal{S}'| = k$. Prove que este problema é NP-completo.

18. Suponha que alguém apresentou um algoritmo de tempo polinomial A para decidir o problema SAT. Com isso, dado uma fórmula ϕ em FNC, apresente um algoritmo B para encontrar uma atribuição das variáveis de ϕ que a satisfaz.

19. Considere o problema de se achar uma atribuição lógica para uma fórmula booleana em forma normal disjuntiva (SAT-D). Uma fórmula em forma normal disjuntiva é uma disjunção de cláusulas $E_1, E_2, \ldots, E_k$ onde cada cláusula é uma conjunção de literais. Exemplo:

    $$(x_1 \cdot x_2) + (\overline{x_2} \cdot x_3) + (\overline{x_3})$$

    O problema consiste em se determinar valores booleanos para as váriáveis de modo que a fórmula se torne verdadeira. Mostre que o SAT-D pode ser resolvido em tempo polinomial.

20. No problema 3SAT, usa-se a forma normal conjuntiva e em cada cláusula há exatamente 3 literais. Sabe-se que esse problema é NP-completo. Por outro lado, poderíamos transformar uma instância de 3SAT numa instância do problema SAT-D usando a propriedade distribuitiva, como no exemplo abaixo:

$$(x_1 + x_2 + \overline{x_3}) \cdot (\overline{x_1} + \overline{x_2}) = (x_1 \cdot \overline{x_2}) + (x_2 \cdot \overline{x_1}) + (\overline{x_3} \cdot \overline{x_1}) + (\overline{x_3} \cdot \overline{x_2}).$$

Conforme o exercício anterior, SAT-D pode ser resolvido em tempo polinomial, e portanto a instância equivalente de 3SAT também pode ser resolvida em tempo polinomial. Isto mostra que 3SAT$\in$P e portanto $P = NP$. Verdadeiro ou falso ? Justifique provando sua resposta.

21. O problema 4-SAT é o mesmo que o problema 3-SAT, com a diferença que cada cláusula tem exatamente 4 literais. O 4-SAT é NP-completo ou não ? Justifique provando sua resposta.

22. O problema do 2-SAT é análogo ao problema do 3-SAT, com a diferença que cada cláusula contém exatamente duas literais. Observe que $x_i + x_j = V$ (satisfeita) é equivalente a $\overline{x_i} \rightarrow x_j$ e $\overline{x_j} \rightarrow x_i$. Isto é, se x_i é falso então x_j é verdadeiro e vice-versa. Observe agora que estas implicações definem um grafo dirigido. Com isso em mente, faça um algoritmo de tempo polinomial para decidir o 2-SAT.
Obs.: Apesar do 2-SAT poder ser resolvido em tempo polinomial, a versão de maximização, onde queremos uma atribuição que maximiza o número de cláusulas satisfeitas não está provada ser polinomial.

23. Dado uma matriz A em $\mathbb{Z}^{m \times n}$, um vetor inteiro b no $\mathbb{Z}^m$, e um vetor inteiro c no $\mathbb{Z}^n$, o problema de programação linear $0/1$ consiste em encontrar um vetor x em $\{0,1\}^n$ tal que $Ax \leq b$ e o valor cx é mínimo. Prove que a versão de decisão do problema de programação linear $0/1$ é NP-completo.
Obs.: O problema de programação linear inteiro (variáveis inteiras não necessariamente 0/1 é também NP-completo. Mas a prova é difícil. No Cormem et al. tem isso como exercício, mas o difícil é mostrar que o problema está em NP. Para isso tem que mostrar que os números do certificado tem tamanho polinomial. O Papadimitriou mostrou num artigo esta complexidade. No livro de Combinatorial Optimization dele com o Steiglitz tem a solução, mas é extensa.

24. Resolva o problema de *Graph-Coloring*, no fim do capítulo de *NP-completeness* do livro do Comen, Leiserson, Rivest, Stein.

25. Decidir se um grafo tem um conjunto independente de tamanho K, onde K faz parte da entrada é um problema NP-completo. Mas e se K fosse constante ? (por exemplo, se $K = 10$). Este problema continua em NP-completo ou está em P ?