Departamento de Informática, PUC-Rio
1 de julho de 2020

ANÁLISE DE ALGORITMOS (INF 1721)

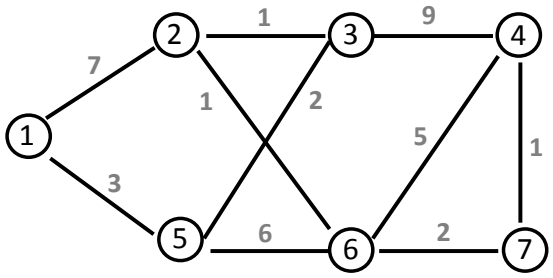
2ª Prova

- Você deve justificar todas as suas respostas;
- **As suas respostas são estritamente pessoais. Qualquer plágio ou tentativa de plágio implicará na nota zero**.
- Escolha **quatro** exercícios para resolver e os anote no início do seu caderno de respostas. Se houver **cinco** exercícios resolvidos, somente os quatro primeiros serão considerados.

---

1. (2.5 pts)

   (a) (1.0 pt) Escreva um pseudocódigo do algoritmo de Djikstra que encontra o caminho mais curto entre dois vértices i e j em um grafo conexo G (com pesos positivos nas arestas). Em seguida, aplique esse algoritmo para encontrar o caminho mais curto entre os vértices 1 e 4 no seguinte grafo.



   (b) (1.5 pts) Em um grafo bipartido completo, todos os vértices de uma partição são ligados a todos os vértices da outra partição. Qual é a altura da árvore gerada por uma BFS em um grafo bipartido completo? Qual é a altura da árvore gerada por uma DFS em um grafo bipartido completo?

---

2. (2.5 pts) Responda se as seguintes afirmações são **verdadeiras (V)** ou **falsas (F)**. Cada resposta correta vale 0.25 pontos. Para cada resposta **F** escreva uma justificativa, correção ou contraexemplo de até duas linhas.

   Exemplo: *Merge sort* tem complexidade linear.
   Resposta: F: Falso. A complexidade de pior caso do *Merge sort* é $\Theta(n \log n)$.

   a) É conhecido que o problema da mochila integral admite um algoritmo de programação dinâmica. Logo, esse problema está na classe P;
   b) Se um problema X se reduz ao problema Z em tempo polinomial e $Z \in$ NP-completo, então $X \in$ NP-completo;
   c) Encontrar uma árvore geradora mínima e testar se um número é primo são problemas que estão em P mas não em NP;
   d) O algoritmo de caminho mais curto de Bellman-Ford só funciona com pesos positivos;
   e) O problema da mochila fracionada pode ser resolvido por um algoritmo guloso;
   f) Algoritmos gulosos são conhecidos por acharem a solução ótima tomando decisões locais que não são ótimas;
   g) Se P=NP, então todos os problemas NP-completos podem ser resolvidos em tempo polinomial;
   h) Busca binária, *merge sort*, e *quick sort* podem ser categorizados como algoritmos de divisão e conquista;
   i) Algoritmos *branch-and-bound* podem resolver problemas NP-completos;
   j) O algoritmo de Prim usa iterativamente a propriedade de corte para achar uma árvore geradora de peso mínimo.

---

3. (2.5 pts) Seja $G = (V, E)$ um grafo completo não direcionado que possui exatamente k arestas de peso 0 e o restante possui peso 1. Assuma que também temos disponível para cada vértice $v \in V$ uma lista das arestas adjacentes com peso 0. Proponha um algoritmo eficiente que encontra uma árvore geradora de custo mínimo em G.

4. (2.5 pts) Considere o seguinte problema de otimização:

   SCHEDULING PROBLEM: Considere uma lista de n tarefas, cada uma com uma data de começo e_i, uma data de término l_i, e um peso positivo inteiro q_i. Assuma que essas tarefas são listadas em ordem crescente de data de término. O objetivo consiste em achar um subconjunto de tarefas realizadas tal que:

   - a qualquer momento no tempo, existe no máximo uma tarefa sendo realizada;
   - a soma dos pesos das tarefas selecionadas não ultrapassa uma capacidade Q;
   - o número total de tarefas realizadas é maximizado.

   (a) (0.8 pts) Mostre que INTEGER KNAPSACK PROBLEM $\leq_p$ SCHEDULING PROBLEM;

   (b) (1 pt) Seja $OPT[k, q]$ o maior número de tarefas que podem ser realizadas usando unicamente as tarefas do intervalo {1,. . . ,k} com uma capacidade total menor que q. Escreva o caso base e a relação de recorrência que permitem o cálculo de $OPT[k, q]$ para cada $k \in \{1, \ldots, n\}$ e cada $q \in \{0, \ldots, Q\}$;

   (c) (0.7 pts) Baseado em (b), descreva um pseudocódigo para o SCHEDULING PROBLEM ;

   (d) (BÔNUS: 0.5 pts) Faça a análise da complexidade assintótica desse algoritmo.

---

5. (2.5 pts) Considere os seguintes algoritmos:

   - O algoritmo 1 resolve o problema de tamanho n dividindo-o em duas recursões de tamanho $n - 1$ seguido por uma etapa que combina os resultados em tempo constante.
   - O algoritmo 2 resolve o problema de tamanho n dividindo-o em 4 subproblemas de tamanho $n/2$ em cada recursão. Todos os subproblemas são combinados em tempo constante.
   - O algoritmo 3 resolve o problema de tamanho n abrindo uma recursão somente na primeira metade seguido por uma etapa polinomial em $O(n^2)$.

   a) (0.6 para cada algoritmo) Apresente o $T(n)$ e a complexidade assintótica em termos de $O(\cdot)$ de cada algoritmo;

   b) (0.7 pts) Qual algoritmo possui a pior complexidade?