COS841/MAB704 Complexidade de Algoritmos
Prova P1 – Gabarito

Professores: F. Marquezino, F. Botler, C. Figueiredo
Universidade Federal do Rio de Janeiro

10 de Dezembro de 2021

Critérios de correção da prova P1 realizada de 10 a 17 de Dezembro de 2021.

1. [3 pontos] Diga se cada uma das afirmações abaixo é verdadeira ou falsa. Justifique sua resposta com uma prova ou um contraexemplo. *Cada resposta correta soma* 1 *ponto; cada resposta errada ou com justificativa errada desconta* 1/2 *ponto; itens deixados em branco ou sem justificativa não somam nem descontam pontos.*

   (1.1) **V** Para quaisquer constantes reais a e b, em que $b > 0$, temos $(n + a)^b \in \Theta(n^b)$

   Esboço da solução: Por meio de manipulações algébricas simples pode-se mostrar que existem sempre constantes que satisfazem a definição da notação Θ.

   (1.2) **F** $2^{2n} \in O(2^n)$

   Esboço da solução: Pode-se demonstrar por contradição.

   (1.3) **V/F** $\log(\lceil \log n! \rceil) \notin O(\log n)$

   Esboço da solução:
   Talvez a notação tenha ficado ambígua, e alguns alunos tenham entendido $\log(n!)$ enquanto outros tenham entendido $(\log n)!$. Portanto vou aceitar ambas as soluções, desde que devidamente justificadas.
   $\log(\lceil \log(n!) \rceil) \notin O(\log n)$ é **falso**. Uma sugestão é tentar mostrar que $\log(\lceil \log(n!) \rceil) \leq \log(n) + \log(\log(n))$ e que portanto é $O(\log n)$.
   $\log(\lceil (\log n)! \rceil) \notin O(\log n)$ é **verdadeiro**. Uma sugestão é tentar demonstrar que $\log(\lceil \log n! \rceil)$ é $\Theta(\log n \log \log n)$, ou $\omega(\log n)$. Logo, não pode ser $O(\log n)$.

2. [2 pontos] Para cada situação abaixo, diga se é possível aplicar o teorema mestre para encontrar uma expressão de crescimento assintótico da complexidade. Justifique suas respostas. *Cada resposta correta soma* 1/2 *ponto; cada resposta errada ou com justificativa errada desconta* 1/4 *ponto; itens deixados em branco ou sem justificativa não somam nem descontam pontos.*

   (2.1) **Sim** Um certo algoritmo desenvolvido pelo método dividir e conquistar consiste em dividir o problema original, cuja entrada é de tamanho n, resolvendo recursivamente três subproblemas de mesma natureza, cada um com entrada de tamanho $n/2$. Para combinar as soluções dos subproblemas leva-se tempo $O(n)$.

   Esboço da solução: $3T(n/2) + O(n)$, logo $T(n) = \Theta(n^{\log_2 3})$

   (2.2) **Sim** O algoritmo executa $T(n)$ passos, onde n é o tamanho da entrada, e

   $$T(n) = 2T(\sqrt{n}) + \log n.$$

   Esboço da solução: Faça $m = \log n$. Logo, $T(2^m) = 2T(2^{m/2}) + m$. Faça $S(m) = T(2^m)$. Logo, $S(m) = 2S(m/2) + m$. Resolvendo pelo teorema mestre temos $S(m) = \Theta(m \log m)$. Logo, $T(n) = \Theta(\log n \log \log n)$.

(2.3) **Não** O algoritmo executa $T(n)$ passos, onde n é o tamanho da entrada, e

$$T(n) = T(n/8)\log n + 2^n.$$

> **Esboço da solução:** O termo que multiplica $T(n/8)$ não é constante.

(2.4) **Não** O algoritmo executa $T(n)$ passos, onde n é o tamanho da entrada, e

$$T(n) = 4T(n/2) - n.$$

> **Esboço da solução:** No lado direito da equação a função sendo somada ao termo recursivo é assintoticamente negativa.

3. [3 pontos] Descreva um algoritmo guloso para calcular o custo da árvore geradora mínima de um grafo não-dirigido com custos nas arestas. Aplique o algoritmo ao grafo da Figura 1, demonstrando cada um de seus passos e dizendo a ordem em que as arestas são adicionadas à árvore geradora mínima.

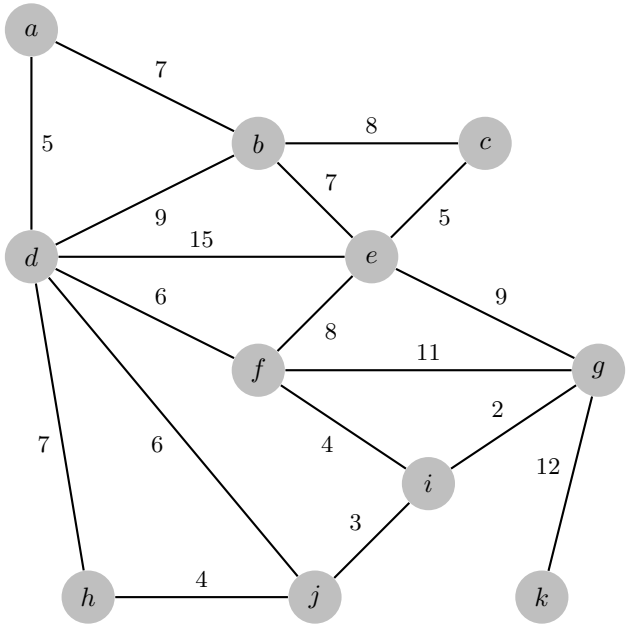

Figura 1: Grafo da Questão 3.

> **Esboço da solução:** As soluções possíveis são diversas, portanto irei analisar caso a caso. Pode-se por exemplo descrever o algoritmo de Prim ou de Kruskal. Para atribuir a nota eu irei levar em consideração também a clareza do texto e posso descontar até 1 ponto de soluções confusas ou com problemas de escrita. A árvore geradora mínima para o grafo da Figura 1 possui custo 55.

4. [2 pontos] Suponha que o algoritmo de Huffman tenha sido usado para obter uma codificação do alfabeto $\{A, T, G, C\}$, cujos caracteres ocorrem com frequências f_A, f_T, f_G e f_C, respectivamente. Para cada um dos códigos abaixo, dê um exemplo de frequências (f_A, f_T, f_G, f_C) que poderiam gerar o código especificado; ou então explique por que o código jamais poderia ser obtido pelo algoritmo de Huffman, independentemente das frequências dos caracteres:

(4.1) Código {0, 10, 110, 111}

> **Esboço da solução:** Diversas soluções possíveis, portanto irei analisar caso a caso. Um exemplo seria: $f_A = 40\%,\ f_T = 30\%,\ f_G = 15\%,\ f_C = 15\%$

(4.2) Código {0, 1, 00, 11}

Esboço da solução: Não pode ser obtido. Não é código de prefixo.