MCTA003-17 – Análise de Algoritmos
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC
Profa. Carla Negri Lintzmayer

Lista 3

Entrega: até 23h55 do dia 30/07/2018

- Submeta ao tidia um único arquivo PDF com as suas soluções escaneadas.
- Seja o mais **formal** possível em todas as respostas.
- Identifique devidamente cada exercício.
- Identifique devidamente os autores da lista (caso você tenha feito em dupla).
- A lista é uma forma de treino para a prova, que não terá consulta. Evite plágio!

1. Vimos em sala de aula o problema da Mochila Fracionária, onde temos como entrada um conjunto de n itens $\{1, 2, \ldots, n\}$, cada item i com um peso $w_i \in \mathbb{N}$ e um valor $v_i \in \mathbb{N}$ e uma mochila com capacidade de peso W. A saída desse problema é uma função f sobre os itens que indica a fração escolhida do mesmo ($0 \leq f_i \leq 1$) tal que $\sum_{i=1}^{n} f_i w_i \leq W$ e $\sum_{i=1}^{n} f_i v_i$ é máximo. O problema da Mochila tem a mesma entrada, mas exige que os valores de f_i sejam inteiros, especificamente 0 ou 1. O algoritmo guloso visto em aula para a Mochila Fracionária é ótimo para o problema da Mochila? Justifique.

2. Execute o algoritmo de Huffman passo a passo na entrada $A = \{a, b, c, d, e, g\}$ onde $f_a = 3$, $f_b = 2$, $f_c = 6$, $f_d = 8$, $f_e = 2$ e $f_g = 6$.

3. Reescreva o algoritmo de Kruskal visto em sala de aula considerando explicitamente a estrutura Union-Find. Escreva as funções $\textsc{Find}(v)$ e $\textsc{Union}(g_1,g_2)$ considerando os vetores `lider` e `tam` mencionados em sala.

4. Mostre que se todos os pesos das arestas são distintos, então o grafo só tem uma árvore geradora mínima. *Dica:* por contradição, suponha que o grafo tem duas árvores geradoras mínimas diferentes e utilize um argumento de troca.

5. Escreva um algoritmo que recebe uma árvore construída pelo algoritmo de Huffman, um alfabeto, uma sequência de bits e retorna a string decodificada.

6. Quantas árvores geradoras mínimas possui o grafo da Figura 1? Execute passo a passo o algoritmo de Kruskal sobre ele para gerar uma delas.

7. Para o problema da árvore geradora mínima, vimos em sala o algoritmo de Kruskal e existe um documento na página da disciplina no meu site sobre o algoritmo de Prim. Existe ainda um terceiro algoritmo clássico para esse problema, que é inclusive mais antigo que os dois mencionados: o algoritmo de Boruvka. Esse também é um algoritmo guloso e ele funciona da seguinte forma: cada vértice começa como uma componente conexa (similar ao Kruskal) e, a cada iteração, de todas as arestas possíveis que conectam dois componentes (existentes naquela iteração), adicione todas as que tenham menor custo. Mostre a execução desse algoritmo sobre o grafo da Figura 2 (certifique-se de que você entendeu realmente sua ideia) e escreva seu pseudocódigo. Esse algoritmo pode ser

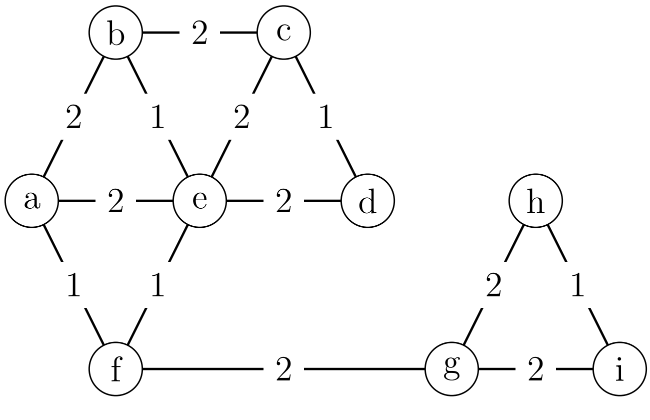

Figura 1: Grafo A.

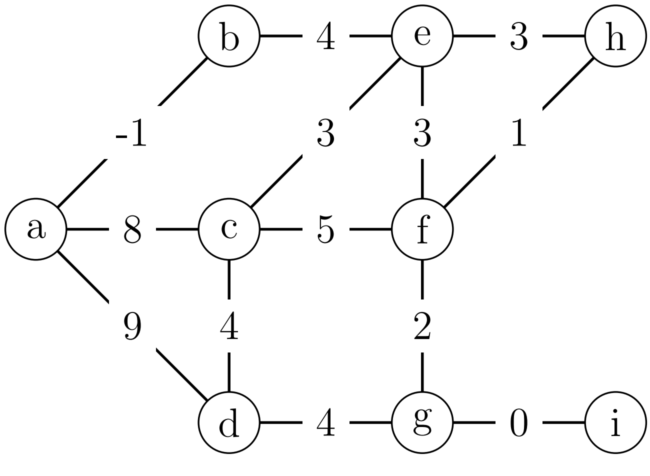

Figura 2: Grafo B.

implementado em tempo $O(|E| \log |V|)$ (o mesmo tempo do Kruskal e do Prim, inclusive). Explique sucintamente como é possível chegar a esse tempo de execução.

8. Considere a seguinte variação de um problema de escalonamento. A entrada é um conjunto de n tarefas $\{1, 2, \ldots, n\}$ onde cada tarefa j tem um peso w_j e um comprimento ℓ_j. Dado um único processador, a saída é uma reordenação das n tarefas $\sigma = (x_1, x_2, \ldots, x_n)$ (isto é, não necessariamente $x_i = i$) onde, se c_j é a soma dos comprimentos das tarefas feitas até a tarefa j na sequência (incluindo o próprio comprimento de j), então $custo(\sigma) = \sum_{j=1}^{n} w_j c_j$ é mínimo.

   Por exemplo, suponha que temos 3 tarefas onde $w_1 = 3$, $w_2 = 2$, $w_3 = 1$, $\ell_1 = 1$, $\ell_2 = 2$ e $\ell_3 = 3$. Um possível escalonamento delas é $\sigma = (2, 1, 3)$ (isto é, a tarefa 2 é executada primeiro, em seguida a tarefa 1 é executada e por último a tarefa 3 é executada). Nesse escalonamento, temos que $c_1 = \ell_2 + \ell_1 = 3$, $c_2 = \ell_2 = 2$ e $c_3 = \ell_2 + \ell_1 + \ell_3 = 6$ e, com isso, $custo(\sigma) = c_1 w_1 + c_2 w_2 + c_3 w_3 = 19$.

   Note que se todas as tarefas tivessem o mesmo comprimento, então a solução ótima seria escolher tarefas de maior peso primeiro. Por outro lado, se as tarefas tivessem o mesmo peso, então a solução ótima seria escolher as tarefas de menor comprimento primeiro. A intuição, portanto, é escolher tarefas de maior peso que ao mesmo tempo tenham comprimento pequeno.

   Descreva um algoritmo guloso que resolve esse problema otimamente. Prove que o seu algoritmo é de fato ótimo usando as ideias de prova por troca vistas em sala de aula.