CCM-001 – Análise de Algoritmos e Estruturas de Dados
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC
Profs. Carla N. Lintzmayer e Guilherme O. Mota

Lista 3

- Seja o mais **formal** possível em todas as respostas.
- A lista é uma forma de treino para a prova, que não terá consulta. Evite plágio!
- Fique à vontade para procurar exercícios nos livros recomendados na bibliografia.

1. Dado um grafo $G = (V, E)$, diga como funciona a representação de G em termos de uma matriz de adjacências e de uma lista de adjacências. Explique quais as vantagens e desvantagens de cada uma dessas formas de representação de grafos.

2. O problema da Mochila Inteira é semelhante ao da mochila fracionária, mas não é permitido adicionar à mochila somente frações dos itens. A estratégia gulosa de escolher sempre o item com maior *valor/peso* encontra uma solução ótima para o problema da Mochila Inteira? Prove ou mostre um contra-exemplo.

3. Defina árvore geradora mínima de um grafo e caminho mínimo de um vértice s a um vértice v em um grafo. Seja $x = 20$ e considere o grafo $G = (V, E)$ com conjunto de vértices $V(G) = \{v_1, v_2, \ldots, v_{\lfloor x/2 \rfloor}\}$, arestas $v_i v_{i+1}$ para $1 \leq i < \lfloor x/2 \rfloor$, e $v_i v_{i+2}$ para i ímpar com $1 \leq i \leq \lfloor x/2 \rfloor - 2$. Considere a função w de pesos em $E(G)$ tal que $w(v_i v_{i+1}) = 1$ para $1 \leq i < \lfloor x/2 \rfloor$, e $w(v_i v_{i+2}) = \left\lfloor \frac{i-1}{2} \right\rfloor$ para i ímpar com $1 \leq i \leq \lfloor x/2 \rfloor - 2$.

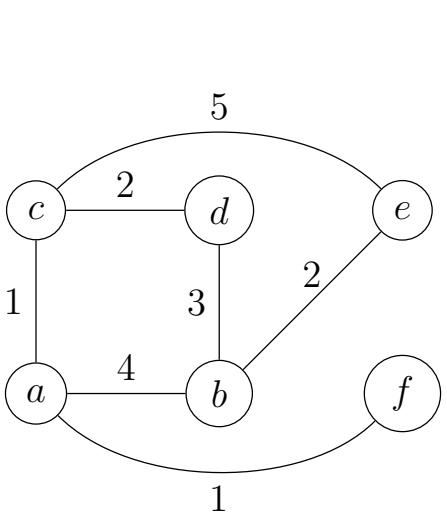
   - Mostre uma árvore geradora mínima de G.
   - Mostre uma árvore geradora de G com a seguinte propriedade: para todo vértice v, o caminho de v_1 a v na árvore é um caminho mínimo de v_1 a v.

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.

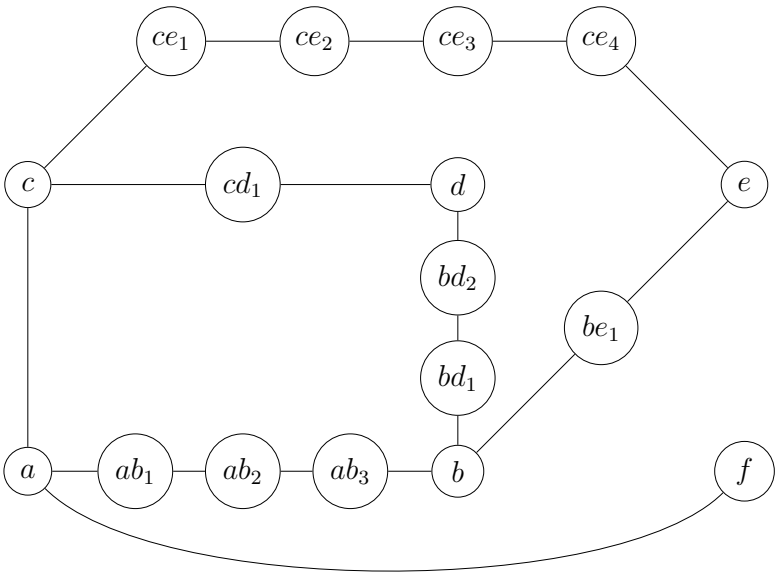
5. Quantas árvores geradoras mínimas possui o grafo da Figura 2a? Execute passo a passo o algoritmo de Kruskal sobre ele para gerar uma delas.

6. 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^σ é a soma dos comprimentos das tarefas feitas até a tarefa j em σ (incluindo o próprio comprimento de j), então $custo(\sigma) = \sum_{j=1}^{n} w_j c_j^\sigma$ é mínimo. Descreva um algoritmo guloso que resolve esse problema otimamente. Prove que o seu algoritmo é de fato ótimo usando um argumento de troca.

7. Seja $G = (V, E)$ um grafo e $c\colon E(G) \to \mathbb{Z}_+$ uma função de custo nas arestas tal que $c(e) \geq 1$. Construa o grafo H tal que cada aresta $e \in E(G)$ é substituída por um caminho com $c(e)$ arestas sem custo em H (ou, similarmente, de custo 1). Assim, H possui os mesmos vértices de G e alguns vértices extras. Veja Figuras 1a e 1b.



(a) Grafo G.



(b) Grafo H.

Figura 1: Exemplos para o exercício 7.

Dado um vértice $s \in V(G)$, qual é o tempo de execução da busca em largura sobre H a partir de s? É possível escrever esse tempo em função de $|V(G)|$ e $|E(G)|$? Para todo $v \in V(G)$, o custo do sv-caminho mínimo em H encontrado pela BFS é o mesmo do sv-caminho mínimo em G?

8. Execute o algoritmo de Dijkstra no grafo da Figura 2b a partir do vértice c.

9. Escreva (com as suas palavras) a prova de que o algoritmo de Dijkstra funciona corretamente. Todos os argumentos essenciais devem estar muito claros em sua prova.

10. (Desafio) Seja $G = (V, E)$ um grafo e $w\colon E(G) \to \{1, \ldots, k\}$ uma função de pesos nas arestas que associa cada aresta a um peso que é um inteiro entre 1 e k. Modifique o algoritmo de Dijkstra para que seu tempo de execução seja $O(kn + m)$.

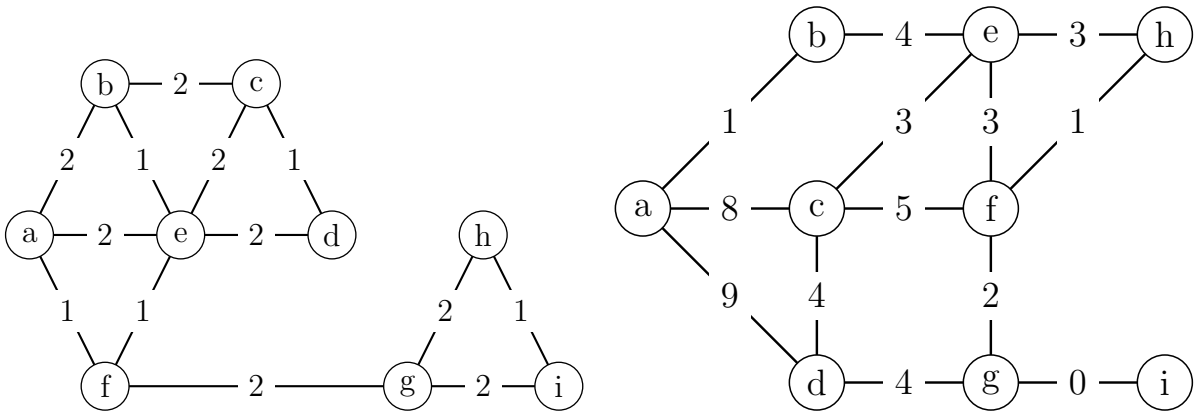
11. Considere um inteiro positivo n e m conjuntos $C_1, \ldots, C_m \subseteq \{1, \ldots, n\}$. Uma *cobertura* é um conjunto de índices $I \subseteq \{1, \ldots, m\}$ tal que

$$\bigcup_{i \in I} C_i = \{1, \ldots, n\} \ .$$

Uma *cobertura mínima* é uma cobertura de menor tamanho possível. O problema da *Cobertura de Conjuntos* consiste em, dados n e $C_1, \ldots, C_m \subseteq \{1, \ldots, n\}$, encontrar a cardinalidade de uma cobertura mínima.

Dado um $I \subseteq \{1, \ldots, m\}$, dizemos que $\{1, \ldots, n\} \setminus \bigcup_{i \in I} C_i$ é o conjunto dos elementos *não cobertos por* I. Considere o seguinte algoritmo guloso para o problema.

```
1: function CoberturaConjuntos(n, C_1, ..., C_m ⊆ {1, ..., n})
2:     I ← ∅
3:     while I não é cobertura do
4:         seja i tal que C_i contém a maior quantidade de elementos não cobertos por I
5:         I ← I ∪ {i}
6:     return I
```



(a) Grafo A. (b) Grafo B.

Figura 2: Grafos para os exercícios da lista.

Esse algoritmo nem sempre encontra uma cobertura mínima para o problema. Mostre uma instância, i.e., descreva conjuntos $C_1, \ldots, C_m \subseteq \{1, \ldots, n\}$ específicos, tal que a cobertura mínima tem somente 2 conjuntos, mas o algoritmo CoberturaConjuntos retorna uma cobertura com $\Omega(\log n)$ conjuntos.