MC558 Projeto e Análise de Algoritmos II
Primeiro Semestre de 2021
Teste 3

Instruções:

(i) As respostas devem ser digitadas usando qualquer editor/formatador (sugiro LaTeX, se você souber usar). As submissões devem ser feitas em formato pdf no Google Classroom; você pode anexar figuras, mas gere um único arquivo. Soluções que não respeitem estas condições receberão **nota ZERO**.

(ii) Todas as respostas devem ter justificativas (corretude e/ou complexidade), a menos que a questão diga explicitamente que não são necessárias.

(iii) Você pode usar qualquer resultado ou algoritmo visto em aula. Conforme o caso, enuncie o resultado ou escreva qual é a complexidade do algoritmo, caso seja necessário na análise de complexidade.

(iv) Em qualquer questão que exija um pseudo-código complicado, explique sua ideia antes de escrevê-lo (no máximo uma página, mas isto provavelmente é muito dependendo da questão). Outra forma é você explicar em alto nível o que faz cada trecho de código.

Sua explicação deve ser boa o suficiente para me convencer que o algoritmo funciona (inclua provas de resultados auxiliares, se necessário). Note que explicar bem não é o mesmo que explicar muito. Soluções que tenham pseudo-códigos complicados, mas sem nenhuma tentativa razoável de explicação não serão consideradas.

(v) Os pseudo-códigos devem ter estilo semelhante aos apresentados em aula ou que estão no livro do CLRS. Pseudo-códigos com trechos de linguagem de programação como C ou Python **não** serão aceitos. Você pode implementar um programa para resolver a questão, se quiser, mas **não** aceitarei como resposta um copy-and-paste do código sob nenhuma hipótese.

(vi) Em um pseudo-código você pode devolver diretamente um conjunto (e.g., escreva "devolva Q"). Você pode usar instruções em português também, e.g., "devolva os vértices da árvore T" ou "ordene a sequência X" ou "execute DFS sobre o grafo G'. Há muitas situações em que é razoável usar uma instrução deste tipo. Tenha em mente que uma instrução deste tipo consome uma certa quantidade de tempo que você deve analisar e deve ser razoavelmente óbvio (para mim) que ela pode ser executada no tempo descrito.

(vii) Em qualquer questão que exija uma descrição de um algoritmo em alto nível (sem pseudo-código), descreva-o de maneira clara e precisa em português. A descrição dos seus passos deve ter detalhes suficicientes para eu poder concluir que o algoritmo tem a complexidade exigida.

(viii) Se você usar alguma notação que não está nos slides ou no CLRS, você deve explicar precisamente o que representa. Não tenho como saber toda notação usada em outras fontes. É esperado também que ninguém invente uma notação para algo que já tem uma notação definida e que foi bastante usada nas aulas.

1. (25 pontos) Professor Sabit Udo propôs um novo algoritmo para resolver o Problema da Árvore Geradora Mínima para **grafos completos ponderados**. Assim, o algoritmo supõe que a entrada é um grafo não-orientado **completo** ponderado (G, ω). Uma **partição balanceada** de V é uma partição de V em dois subconjuntos V_1, V_2 tais que $|V_1|$ e $|V_2|$ diferem de no máximo uma unidade (ou seja, se $|V|$ for par, V_1 e V_2 têm o mesmo tamanho, senão um deles tem tamanho $\lfloor |V|/2 \rfloor$ e o outro $\lfloor |V|/2 \rfloor + 1$. Eis o pseudocódigo do algoritmo em alto nível.

```
TALVEZ-AGM(G, ω)  ▷ G é completo
 1  se |V[G]| ≤ 2 então devolva G
 2  seja V1, V2 qualquer partição balanceada de V[G]
 3  seja Gi o subgrafo completo de G contendo apenas os vértices de Vi e todas as arestas
      de G ligando vértices de Vi para i = 1, 2
 4  seja ωi a restrição de ω a Gi para i = 1, 2
 5  T1 ← TALVEZ-AGM(G1, ω1)
 6  T2 ← TALVEZ-AGM(G2, ω2)
 7  seja e uma aresta de peso mínimo no corte (V1, V2) em G
 8  devolva T1 ∪ T2 ∪ {e}
```

Prove que o algoritmo TALVEZ-AGM funciona ou mostre um contra-exemplo. **Justifique.**

2. (75 pontos) A Comissão de Estudo dos Transportes (CET) recebeu a tarefa de analisar o custo de viagem (em termos de pagamento de pedágios) de uma cidade s a uma cidade t. Uma viagem de s a t deve passar por vários trechos com pedágios. Através de um estudo estatístico ela concluiu que um motorista típico que vai de s para t está disposto a pagar no máximo k reais no total. A CET gostaria de saber qual é a rota de s para t na qual o motorista tem que pagar o maior pedágio (em um trecho), com a restrição de que o custo total seja no máximo k.

   Mais precisamente, a CET tem um grafo orientado ponderado (G, ω) que representa a rede rodoviária em que cada vértice representa uma cidade, uma aresta (u, v) representa uma estrada de u para v e o peso $\omega(u, v)$ é o valor do pedágio naquela estrada. O **preço** de um caminho P é peso da aresta mais pesada em P (i.e., o maior preço que deve-se pagar nesta rota). Dado k, a CET quer saber dentre todos os possíveis caminhos de s a t com peso total no máximo k, qual tem o maior preço.

   Exemplo: considere o grafo da Figura 1 e suponha que $k = 22$.

   - O caminho (s, u, v, t) tem peso total $7 + 10 + 6 = 23 > k$ e não deve ser considerado.
   - O caminho (s, x, y, t) tem peso total $6 + 9 + 7 = 22 \leq k$ e o preço é 9.
   - O caminho (s, p, q, t) tem peso total $5 + 8 + 6 = 19 \leq k$ e o preço é 8.

   Neste caso, a resposta é 9.

   Observação: o caminho de maior preço não precisa ser simples.

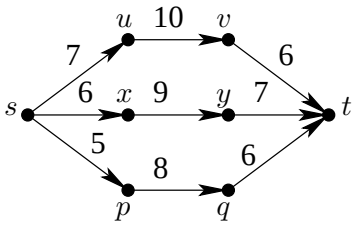

Figura 1: Exemplo com $k = 22$.

Hipóteses:

(a) Nesta questão, todos os grafos são e devem ser representados por listas de adjacências. Além, não têm arestas de peso negativo.

(b) Você tem à disposição um algoritmo chamado Dijkstra que dado um grafo orientado ponderado (G, ω) e um vértice $s \in V[G]$, devolve um vetor $d[\,]$ indexado por $V[G]$ tal que $d[v] = \text{dist}(s, v)$ para todo $v \in V[G]$. Este algoritmo é dado como uma **caixa-preta**, i.e., você não sabe como ele é implementado internamente.

(c) Suponha que Dijkstra tem complexidade de tempo $O(f(V + E))$ (uma função de V e E). A complexidade exata não é importante, mas deve ser respeitada na solução (veja abaixo).

O que a questão pede?

Dada um grafo (G, ω), $s, t \in V[G]$ e um valor $k > 0$, mostre como usar Dijkstra para determinar o maior preço entre todos os caminhos de s a t de peso total no máximo k. Explique sua ideia **sucintamente** e escreva um pseudo-código para seu algoritmo. Você **não** precisa provar que o algoritmo está correto, mas sua explicação deve ser clara o suficiente para eu me convencer disto. A complexidade do seu algoritmo deve ser $O(V + E + f(V + E))$. Note que você não precisa devolver o caminho, apenas o preço dele.

Observações:

- Soluções que "modificam" o Dijkstra ou usam outro algoritmo de caminhos mínimos receberão **nota ZERO**. Você deve usar o algoritmo como especificado.
- Soluções que não atendem à complexidade de tempo exigida serão consideradas erradas (a nota vai depender do que foi feito).
- Você pode usar uma mistura de português e algoritmês para descrever o pseudo-código.
- Você pode modificar o grafo ou construir outros grafos para serem usados como entrada do Dijkstra.
- Você pode usar resultados ou algoritmos vistos em aula ou nas listas de exercícios, excluindo os da primeira observação.
- A nota levará em conta a qualidade de sua explicação (25 pontos).