Nome: _____

RA: _____

Assinatura: _____

Observações:

- Leia todas as perguntas antes de começar. Você pode solicitar esclarecimentos durante a prova.
- Explique e justifique todas as respostas.
- Você também pode obter pontos por respostas parciais.
- É proibido qualquer tipo de consulta bibliográfica.
- A prova é individual, qualquer detecção de fraude implicará em zerar a nota.

Questão 1 (2.5 pt) *Responda verdadeiro ou falso. Justifique cada caso mediante prova ou contraexemplo:*

(a) ***(0.5 pt)*** *Em grafos não direcionados, o DFS não pode ter arestas de cruzamento.*

Resposta. *Verdadeiro. Suponha que (u, v) seja uma aresta de cruzamento em um DFS e, sem perda de generalidade, que foi detectada durante o processamento do vértice u. Observe que, no momento da descoberta da aresta, v não pode ser branco, senão seria aresta da árvore; v também não pode ser cinza, senão seria ancestral de u e a aresta seria de retorno. Isso significa que o vértice v é preto e seu processamento já foi finalizado. Mas como o grafo é não direcionado, a aresta teria sido detectada durante o processamento de v, contradizendo que sua descoberta fosse no processamento de u.*

(b) ***(0.5 pt)*** *Qualquer aresta de retorno em um DFS pertence a um ciclo.*

Resposta. *Verdadeiro. Seja (u, v) uma aresta de retorno em um DFS e, sem perda de generalidade, que foi detectada no processamento de u. Isso significa que v é ancestral de u na árvore do DFS. Usando as arestas da árvore do DFS, temos então um caminho P de v até u e tal caminho não contém a aresta (u, v), pois ela é de retorno e não da árvore. Portanto, ao adicionarmos a aresta (u, v) a P temos um ciclo, que contém dita aresta.*

(c) ***(0.5 pt)*** *Todo grafo de n vértices e $n - 1$ arestas é uma árvore.*

Resposta. *Falso. Considere um grafo formado por um triângulo (um ciclo com três vértices) e um vértice isolado. Tal grafo tem $n = 4$ vértices e $n-1 = 3$ arestas. Contudo, ele não é uma árvore, pois possui um ciclo e não é conexo.*

(d) ***(0.5 pt)*** *Se em um grafo de n vértices o grau de cada vértice for pelo menos k, então o grafo tem pelo menos $\frac{k \cdot n}{2}$ arestas.*

Resposta. *Verdadeiro. O Teorema de Aperto de Mãos nos diz que a soma dos graus dos vértices é igual a duas vezes o número de arestas, ou seja:*

$$2|E| = \sum_{v \in V} d(v) \geq \sum_{v \in V} k = k \cdot |V|$$

Daí temos que, $|E| \geq \frac{k \cdot |V|}{2}$.

(e) ***(0.5 pt)*** *Todo grafo bipartido e conexo tem pelo menos um ciclo de comprimento par.*

Resposta. *Falso. Em aula foi visto que toda árvore com pelo menos dois vértices é um grafo bipartido e árvores são conexas e acíclicas (não tem ciclos de qualquer comprimento). Assim, qualquer árvore com pelo menos dois vértices é um contraexemplo válido.*

Questão 2 (2.5 pt) *Uma certa cidade precisa dar manutenção à rede de metrô. A manutenção será realizada em alguns dos trechos que conectam diretamente duas estações, enquanto os trechos que não receberem manutenção serão desativados. Como o prefeito deseja investir também em saúde e educação, seu objetivo é minimizar o número de trechos que receberão manutenção. Contudo, a prefeitura se preocupa com a qualidade do serviço para os cidadãos e, por isso, deseja garantir que:*

- *Deve existir pelo menos um caminho (formado pelos trechos que vão receber manutenção) conectando quaisquer duas estações do metrô.*
- *Os trechos mantidos devem maximizar a média de uso diário. Para isso, a prefeitura conta com informações sobre o número médio de cidadãos que utilizam cada trecho por dia.*

Proponha um algoritmo eficiente para decidir quais trechos receberão manutenção. Argumente sobre a correção e a complexidade da sua solução.

Resposta. *Podemos modelar o problema com um ponderado grafo, onde cada estação de metrô é um vértice e cada trecho é uma aresta cujo peso é o número médio de cidadãos que a utilizam.*

Uma restrição do problema é que na solução, deve haver um caminho entre quaisquer duas estações, ou seja uma solução válida deve ser um grafo gerador conexo. Mas desejamos desativar tantos trechos como seja possível, isto é, procuramos por um subgrafo gerador conexo com número de arestas mínimo. Note que, tal grafo deve ser acíclico, pois se tiver um ciclo, podemos eliminar mais uma aresta mantendo a conexidade. Desta forma, procuramos por uma árvore (subgrafo acíclico e conexo) geradora com número mínimo de arestas. Mas, qualquer árvore com $|V|$ vértices, tem $|V|-1$ arestas e como desejamos maximizar a média de uso diário nos trechos que fiquem, estamos procurando por uma árvore geradora de peso máximo.

Se T^* é uma árvore geradora de peso máximo, isso significa que a soma dos pesos das suas arestas é maior ou igual que a de qualquer outra árvore geradora T:

$$\sum_{e \in T^*} \omega(e) \geq \sum_{e \in T} \omega(e) \qquad \equiv \qquad \sum_{e \in T^*} (-\omega(e)) \leq \sum_{e \in T} (-\omega(e)).$$

Ou seja, encontrar uma árvore geradora de peso máximo, equivale a encontrar uma árvore geradora de peso mínimo, quando no mesmo grafo multiplicamos os pesos das arestas por -1.
Desta forma, basta usar um dos algoritmos para árvore geradora mínima visto em aula: o de Prim com complexidade $O(E \log V)$ ou o de Kruskal cm complexidade $O(E \log E)$ (neste caso consideramos que o grafo é conexo e portanto $|E| \geq |V| - 1$).

Questão 3 (2.5 pt) *Uma certa universidade definiu uma rota e pontos de ônibus para um fretado que buscará seus alunos. Sabendo que cada aluno irá ao ponto de ônibus mais próximo de sua moradia, a administração gostaria de saber o número de alunos que irão a cada ponto. Para isso, a administração digitalizou um mapa da cidade. Esse mapa possui informações sobre o comprimento de cada rua, assim como dos cruzamentos, os locais dos pontos de ônibus fretado e as moradias dos alunos.*
Se o mapa tem n_c cruzamentos de ruas, a rota possui n_p pontos de ônibus do fretado, na universidade há n_a alunos e na cidade existem m ruas. Projete um algoritmo de complexidade $O(m \cdot \log(n_c + n_p + n_a))$ que encontre o número de alunos que irá a cada ponto de ônibus. Pode considerar que consultas sobre a distância entre quaisquer dois locais na mesma rua têm complexidade $O(1)$. Também tem complexidade $O(1)$ obter a lista de locais de interesse (cruzamentos, moradias ou pontos de ônibus) adjacentes a um local qualquer. São adjacentes dois locais se estão na mesma rua e não há um outro local de interesse entre eles nessa rua.
Argumente a correção e complexidade da sua solução.
Resposta. *Podemos modelar o problema como um grafo ponderado, onde cada local de interesse (como pontos de ônibus, moradias ou cruzamentos) é representado por um vértice, e as arestas conectam os locais de interesse adjacentes, sendo os pesos das arestas as distâncias entre dois locais adjacentes.*
Observe que, com exceção dos locais que estão em cruzamentos, os outros locais estão no interior de alguma rua e, portanto, têm apenas dois adjacentes: os extremos da rua (os cruzamentos) ou outros locais de interesse no interior da mesma rua. Logo, o número total de adjacências é igual ao número de ruas (m), mais o número de adjacentes internos nas ruas, o que resulta em $O(n_p + n_a)$, onde n_p é o número de pontos de ônibus e n_a é o número de moradias. Como, em uma cidade, o número de pontos de ônibus e moradias no mesmo quarteirão é limitado por uma constante, temos que $n_p + n_a = O(m)$. Portanto, o número de arestas do grafo é $O(m)$.
A construção do grafo usando uma lista de adjacências a partir do mapa digitalizado pode ser feita em $O(n_c + n_p + n_a + m)$, da seguinte forma:

1. *Defina um vértice para local de interesse.*

2. *Por cada local de interesse x, percorra os adjacentes e, para cada adjacente y, insira a resta (x, y) na lista ligada de x com peso igual à distância entre x e y.*

A primeira linha pode ser executada com complexidade $O(n_c + n_p + n_a)$, enquanto a segunda linha tem complexidade $O(n_c + n_p + n_a + m)$. No caso da segunda, percorrer os locais de interesse e seus adjacentes requer $O(n_c + n_p + n_a + m)$ operações: obter a lista dos adjacentes de um ponto é $O(1)$, assim como calcular a distância entre um local e seu adjacente.
O problema se reduz a encontrar, no grafo construído, o ponto de ônibus mais próximo de cada vértice que corresponde a um estudante. Para isso, adicionamos ao grafo um vértice extra s, com uma aresta de peso 0 conectando s a cada vértice correspondente a um ponto de ônibus. Note que isso mantém o número de arestas no grafo em $O(m)$ e o número de vértices em $O(n_c + n_p + n_a)$.
Agora, provaremos que, ao calcular os caminhos mínimos a partir de s para o restante do grafo utilizando o algoritmo de Dijkstra, que tem complexidade $O(m \cdot \log(n_c + n_p + n_a))$, obtemos uma árvore de caminhos mínimos em que o ponto de ônibus que serve como ancestral de um vértice é o ponto de ônibus mais próximo desse vértice.
Observe que qualquer caminho de s a outro vértice do grafo deve passar por pelo menos um vértice correspondente a um ponto de ônibus, pois, por construção, estes são os únicos adjacentes de s.
Considere um vértice qualquer $x \neq s$ do grafo. Denote por v o vértice de ponto de ônibus adjacente a s no caminho entre s e x na árvore encontrada por Dijkstra. Suponha que existe um vértice de ponto de ônibus w que esteja mais perto de x do que v. Isso implicaria que o caminho de x até w, somado à aresta (s, w) (de peso 0), seria um caminho de menor custo entre s e x do que o caminho de x até v, somado à aresta (s, v) (de peso 0), que foi retornado pelo algoritmo de Dijkstra — o que seria uma contradição. Portanto, concluímos a prova da propriedade.
Com isso, basta realizar uma busca em profundidade (DFS) na árvore retornada por Dijkstra, partindo de cada vértice correspondente a um ponto de ônibus, para contar os vértices do tipo moradia de estudantes que são seus descendentes. Como a estrutura é uma árvore, o número de arestas é igual ao número de vértices menos um, e, portanto, a complexidade da DFS depende apenas do número de vértices, resultando em uma complexidade de $O(n_c + n_p + n_a)$.
Desta forma, a complexidade total da solução é dominada pela de Dijkstra e resulta $O(m \cdot \log(n_c + n_p + n_a))$.

Questão 4 (2.5 pt) *Sejam k um inteiro positivo e uma rede (G, c, s, t) onde toda capacidade é múltiplo de k. Responda as seguintes questões:*

(a) ***(1.5 pt)*** *Existe um fluxo máximo em que, por cada aresta, passa um fluxo que é múltiplo de k? Argumente a sua resposta através de uma prova ou um contraexemplo.*

Resposta. *Sim. Para isso, provaremos que, no início de cada iteração do algoritmo de Ford-Fulkerson, o fluxo que está sendo construído satisfaz a condição de que o valor que passa por cada aresta é múltiplo de k:*

- *Caso base. Na inicialização, o fluxo em cada aresta é zero, e zero é múltiplo de qualquer inteiro $k > 0$.*
- *Passo. Considere o início de uma iteração e, por hipótese de indução, suponha que o fluxo que passa por cada aresta é múltiplo de k.*

  A capacidade residual $c'(e)$ de uma aresta e da rede residual é calculada da seguinte forma:

  - *$c'(e) = c(e) - f(e)$, se a aresta e está em G (o grafo original).*
  - *$c'(e) = f(e)$, se a aresta e não está em G (a direção reversa na rede residual).*

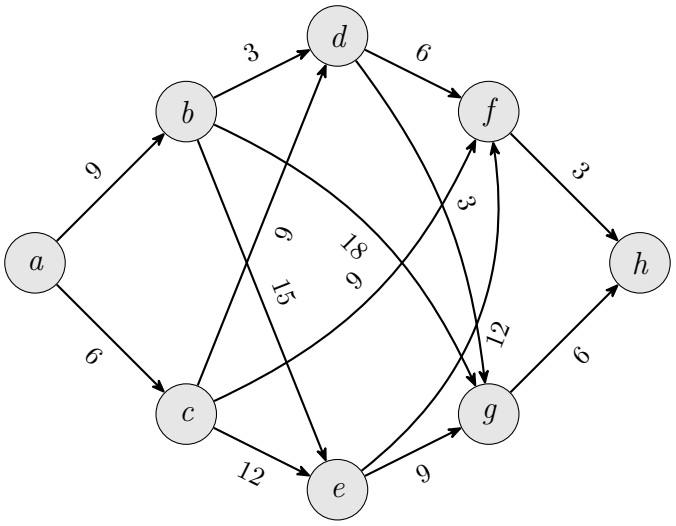
  Como tanto a capacidade $c(e)$ quanto o fluxo $f(e)$ são múltiplos de k por hipótese de indução, a capacidade residual $c'(e)$ também será múltiplo de k. Assim, todas as capacidades da rede residual são múltiplos de k.

  Se houver um caminho aumentador, a capacidade residual desse caminho será um múltiplo de k, e o fluxo em cada aresta será modificado somando ou subtraindo um múltiplo de k. Portanto, o fluxo em cada aresta, ao final da iteração, continuará sendo um múltiplo de k.

  Se não existir um caminho aumentador, o algoritmo para, mantendo o mesmo fluxo, que é múltiplo de k em cada aresta.

Como o algoritmo encontra um fluxo máximo e, nessa solução, o fluxo que passa por cada aresta é múltiplo de k, concluímos que existe um fluxo máximo em que o fluxo em cada aresta é múltiplo de k.

(b) ***(0.5 pt)*** *Considere a seguinte rede com a fonte sendo a e o terminal h:*



O fluxo seguinte fluxo é máximo? Argumente.

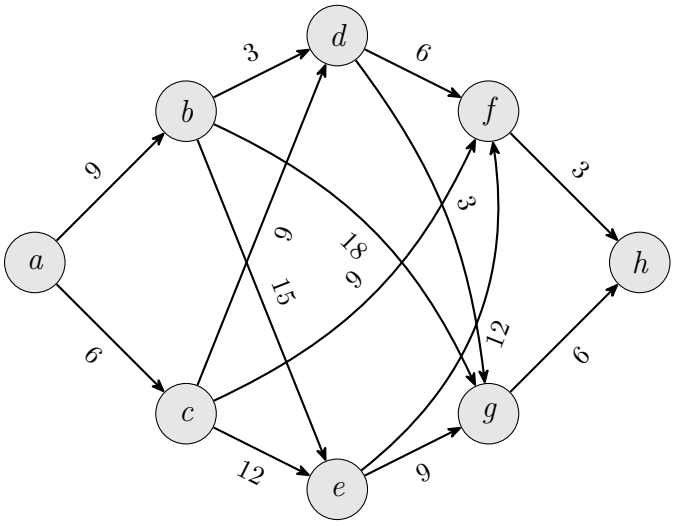
$$f(e) = \begin{cases} 9, \text{ se } e = (a,b) \\ 3, \text{ se } e \in \{(b,d),(d,f),(f,h)\} \\ 6, \text{ se } e \in \{(b,e),(e,g),(g,h)\} \\ 0, \text{ em outro caso.} \end{cases}$$

Resposta. *Sim. O valor desse fluxo é* 9 *(a soma do fluxo que sai de a menos o que entra). Note que o corte* (a,b,c,d,e,f,g,h) *tem capacidade* 9*. Pelo teorema do Fluxo Máximo e Corte Mínimo, sabemos que, se o valor de um fluxo coincide com a capacidade de um corte, então esse fluxo é máximo.*

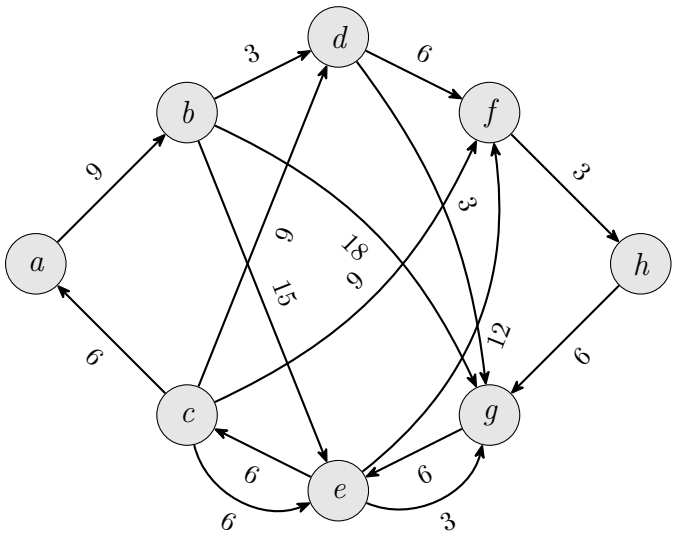
(c) ***(0.5 pt)*** *Usando a rede do item anterior (**4.(b)**), simule as duas primeiras iterações de alguma implementação de Ford-Fulkerson. Em cada iteração, deve ilustrar os principais componentes do algoritmo: fluxo no início e fim da iteração, caminho aumentador selecionado, rede residual, etc.*

Resposta. *Na inicialização o fluxo que passa por cada aresta é* 0*. Seguem as duas primeiras iterações de alguma implementação de Ford-Fulkerson:*

- *Iteração 1:*
  - *Fluxo inicial:* $f(e) = 0$ *para toda aresta* $e \in E$*.*
  - *Rede residual (mesmo grafo de entrada):*



  - *Caminho aumentador:* (a,c,e,g,h) *com capacidade residual* 6*.*
  - *Fluxo final:* $f(e) = \begin{cases} 6, \text{ se } e \in \{(a,c),(c,e),(e,g),(g,h)\} \\ 0, \text{ em outro caso.} \end{cases}$
- *Iteração 2:*
  - *Fluxo inicial:* $f(e) = \begin{cases} 6, \text{ se } e \in \{(a,c),(c,e),(e,g),(g,h)\} \\ 0, \text{ em outro caso.} \end{cases}$
  - *Rede residual (só mudam as arestas do caminho aumentador na iteração anterior):*



– *Caminho aumentador:* (a, b, d, f, h) *com capacidade residual* 3.

– *Fluxo final:* $f(e) = \begin{cases} 6, \text{ se } e \in \{(a,c),(c,e),(e,g),(g,h)\} \\ 3, \text{ se } e \in \{(a,b),(b,d),(d,f),(f,h)\} \\ 0, \text{ em outro caso.} \end{cases}$