Prova com consulta. Duração: 2h00m. Exame Época Normal

Nome do estudante: ..... Nº.....

Informação aos estudantes: A consulta permitida inclui slides das aulas teóricas, livros e outros materiais impressos. Não serão permitidas folhas manuscritas avulsas de qualquer tipo ou acesso à Internet (Tablets, portáteis, etc). Telemóveis deverão permanecer **DESLIGADOS** durante a duração do exame. Responder os seguintes grupos de questões em folhas separadas (uma folha para cada grupo): {1, 2}, {3, 4}, {5, 6}.

1. [**4 valores**] Um jardim foi dividido em várias posições. Considerando que é representado em duas dimensões, este tem dimensão **NxM**. Algumas das posições do jardim contêm **K** quantidade de ouro que pode ser obtido se o espaço for escavado. Pretende-se explorar este jardim e tentar extrair o máximo de ouro possível. Porém, há as seguintes limitações:

i. Tem-se que começar pelo canto superior esquerdo do jardim (posição 0x0);

ii. A saída encontra-se no canto inferior direito do jardim;

iii. A cada momento só se pode ir para a direita ou para baixo, nunca podendo voltar para trás;

iv. Todos os locais visitados são escavados e passe-se a possuir o ouro nele existente.

Considere que há disponível uma matriz bidimensional que representa o jardim e que cada posição inclui a quantidade de ouro aí existente. Apresente o código C++ que resolve os seguintes problemas:

a) [2 valores] Usando uma abordagem de retrocesso (*backtracking*), calcule a quantidade máxima de ouro que pode ser extraída do labirinto. Explique a complexidade temporal desta solução.

b) [2 valores] Usando uma abordagem de Programação Dinâmica, calcule qual a quantidade máxima de ouro que pode ser extraída do labirinto. Explique a complexidade temporal desta solução.

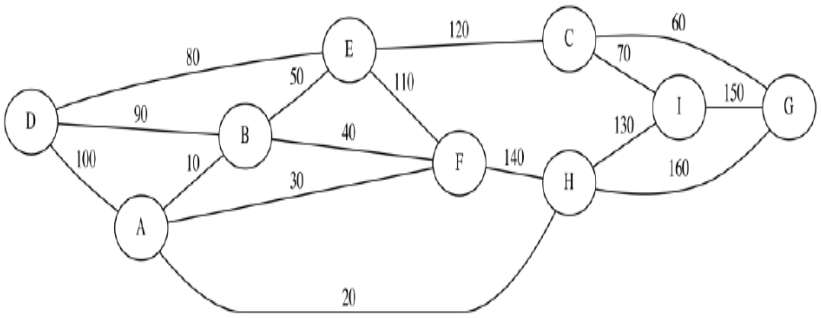
2. [**3 valores**] Um aluno de CAL quer maximizar a nota que vai tirar no exame; porém, sabe que não terá tempo para resolver todos os exercícios. À partida, conhece a lista de problemas no enunciado e o valor de cada problema a resolver. Considere ainda que sabe quanto tempo, em minutos, demoraria a resolver cada problema, e que tem 120 minutos disponíveis para realizar o exame.

a) [1 valor] Elabore o pseudo-código de um algoritmo que, com base na informação disponível, defina qual a ordem pela qual o aluno deve resolver os exercícios, de forma a maximizar a sua nota no tempo disponível.

b) [1 valor] Altere a implementação anterior para mostrar qual a nota esperada, se resolvidos os exercícios pela ordem obtida previamente. Só devem ser contabilizados exercícios totalmente terminados, ignorando-se os exercícios que não têm tempo para ser acabados.

c) [1 valor] Discuta se a solução encontrada é uma solução ótima.

3. [**3 valores**] Considere o grafo **G**, pesado e não dirigido, apresentado na figura, em que os vértices representam locais e o valor de uma aresta representa o custo associado à dificuldade de efetuar a travessia entre vértices adjacentes. O custo (dificuldade) de efetuar uma travessia é um valor inteiro >=1.



a) [0,5 valores] Apresente a ordem em que os vértices são adicionados à árvore de expansão mínima, usando o algoritmo de *Kruskall*. Explique. Apresente a árvore de expansão mínima obtida.

b) [0,5 valores] Apresente a ordem em que os vértices são adicionados à árvore de expansão mínima, usando o algoritmo de *Prim*. Explique. Apresente a árvore de expansão mínima obtida.

c) [1 valor] Uma nova aresta (H, C) é adicionada ao grafo. Qual o intervalo de valores possíveis para o custo desta aresta, de modo a que esta pertença à árvore de expansão mínima? Justifique. Considerando o custo de (H, C) o menor valor do intervalo determinado anteriormente, calcule o valor da nova árvore de expansão mínima.

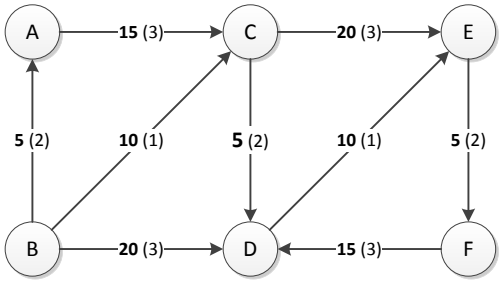
d) [1 valor] Considere que uma aresta de custo≥100 representa uma travessia de dificuldade elevada e uma aresta de custo<100 representa uma travessia de dificuldade mediana. Apresente um algoritmo, em pseudo-código ou C++, que determina a árvore de expansão que minimiza o número de travessias de dificuldade elevada.

4. [**3 valores**] A distância de edição entre duas *strings* **A** e **B** é o menor número de alterações necessárias para transformar **A** em **B**, em que as alterações podem ser substituição, inserção ou eliminação de um carácter.

a) [1,5 valores] Calcule a distância de edição entre as *strings* "**parque**" e "**rega**", apresentando todos os passos/cálculos que efetuar. Identifique as alterações necessárias para transformar a string "**parque**" na string "**rega**".

b) [1,5 valores] Apresente um algoritmo, em pseudo-código ou C++, que determine, não só o valor da distância de edição entres duas *strings* A e B, mas também o conjunto de operações a efetuar. Sugestão: adapte o algoritmo estudado nas aulas.

5. [**3 valores**] Uma empresa de transportes leva continuamente produtos de dois depósitos, situados nos vértices *A* e *B*, ao porto situado no vértice *E*, de onde são exportados para os seus destinos finais. Considere que as arestas do grafo da figura representam estradas, os números em **bold** representam as capacidades de escoamento de produtos pelas respectivas estradas, e os números entre parêntesis o valor a pagar pela unidade transportada. Responda às alíneas seguintes, justificando a sua resposta:



a) [0,5 valores] Qual a capacidade máxima de produtos que a empresa poderá transportar de *A* e *B* até *E*?

b) [1,5 valores] Qual o valor total a pagar pelo volume máximo de produtos transportados pela empresa até *E*?

c) [0,5 valores] Se, adicionalmente ao porto *E*, for construído um novo porto no vértice *F*, o que acontecerá com a capacidade de transporte da empresa?

d) [0,5 valores] Se, ao contrário da alínea anterior, transferirmos o porto do vértice *E* para o vértice *D*, o que acontecerá com a capacidade de transporte da empresa?

6. [**4 valores**] Dados dois grafos $\mathbf{G_1}$ e $\mathbf{G_2}$, o problema de isomorfismo de subgrafos (*Subgraph Isomorphism*, *SI*) procura definir uma função unívoca, *f*, que mapeie vértices de $\mathbf{G_1}$ a vértices de $\mathbf{G_2}$, de tal forma que exista uma aresta (u, v) em $\mathbf{G_1}$ se, e somente se, exista uma aresta $(f(u), f(v))$ pertencente a $\mathbf{G_2}$ – ou seja, procura-se saber se $\mathbf{G_1}$ é um subgrafo de $\mathbf{G_2}$. Este tipo de problema tem aplicações práticas em áreas como, por exemplo, reconhecimento de padrões, visão computacional, ou mesmo química orgânica. Prove que *SI* é NP-completo explicando, passo-a-passo, o procedimento utilizado.

Sugestão: Caso necessário, considere que os problemas seguintes são reconhecidamente NP-completo. Se desejar, poderá também considerar outros problemas da classe NP-completo, para além dos enunciados abaixo:

Cobertura de vértices (*Vertex-Cover Problem*, VCP): Dado um grafo **G**=(***V***, ***E***), encontrar uma cobertura dos vértices de **G** é encontrar um subconjunto $\boldsymbol{W} \subseteq \boldsymbol{V}$ tal que, para toda aresta $(i, j) \in \boldsymbol{E}$, tem-se $i \in \mathbf{W}$ ou $j \in \mathbf{W}$, ou ambos.

Clique: Dado um grafo não dirigido **G**=(***V***, ***E***), um subconjunto $\boldsymbol{W} \subseteq \boldsymbol{V}$ é dito ser um *clique* do grafo **G** se, e se somente, $\forall\ u, v \in \boldsymbol{W}$ então $(u, v) \in \boldsymbol{E}$, ou seja, quaisquer pares de vértices em ***W*** são adjacentes.

Bom Exame!