MAUÁ	PREENCHA OS CAMPOS EM AMARELO	PROVA SUB2
Disc.: **ECM306 – TÓPICOS AVANÇADOS EM ESTRUTURA DE DADOS**		
Curso: **ENGENHARIA DA COMPUTAÇÃO**		

Aluno: **Pedro Wilian Palumbo Bevilacqua**		*Test Code*: **ITTNB**
Curso: **ECM** Série **3ª** Período: **Matutino**		
São Caetano do Sul, 4 de dezembro de 2025.	RA: **23.01307-9**	
Assinatura: **Pedro Wilian Palumbo Bevilacqua**	Nota:	

Instruções da Prova

1. Esta prova é individual e prática, devendo ser realizada nos computadores do IMT, sendo permitido ao aluno, se assim desejar, utilizar seu próprio computador, sob sua inteira responsabilidade;
2. Não poderá haver acesso à Internet, sob nenhuma circunstância, exceto ao CanvasLMS do próprio aluno e, mesmo assim, somente em duas etapas: para receber ("baixar") as questões de sua prova na máquina em uso; e para entregar ("subir") as resoluções das questões de sua prova no devido local de entrega, na mesma plataforma.

   IMPORTANTE: O aluno deverá informar ao professor quando fará os dois acessos permitidos a ele ao CanvasLMS, pela Internet;
3. Poderá haver consulta a qualquer material do próprio aluno, seja ele físico (livros, artigos, material de aula etc.) ou virtual (livros, artigos, material de aula, exercícios, resoluções etc.), desde que esse material esteja previamente armazenado em seu computador ou em qualquer dispositivo de armazenamento externo (*pendrive*, *hd* externo etc.).

   IMPORTANTE: Não será permitido o acesso pela Internet a pastas compartilhadas (*Google Drive*, *OneDrive* etc.), nem a repositórios virtuais (*GitHub*, *GitLab*, *BitBucket* etc.), mesmo sendo de posse e administração do próprio aluno;
4. O aluno deverá responder às questões da prova no próprio arquivo da prova (*.docx*) devidamente identificado (RA e Nome completo do Aluno) e quando for o caso, gerando os códigos necessários, somente na linguagem de programação JAVA e utilizando sempre e somente o paradigma da Programação Orientada à Objetos visto nas aulas;
5. Para realizar a entrega da prova na plataforma CanvasLMS, o aluno deverá gerar e entregar um único arquivo compactado (*.rar* ou *.zip*), tendo seu RA e seu NOME completo como nome desse arquivo. Nesse arquivo compactado o aluno deverá fornecer, obrigatoriamente, os seus dados acadêmicos preenchidos na prova, bem como a resolução das questões nela solicitadas (*.docx*), além dos arquivos e códigos gerados em suas resoluções, uma pasta para cada questão, contendo as suas classes e demais arquivos que possibilitem sua posterior execução pelo professor, durante a resolução.

   IMPORTANTE: Na correção, para executar o código gerado pelo aluno em sua prova, o professor seguirá exatamente as instruções fornecidas pelos alunos e contidas nas resoluções das provas! Caso não obtenha sucesso, a questão será considerada errada.
6. Não poderá haver troca de informações, nem de materiais, sejam físicos ou virtuais, entre os alunos durante a prova;
7. Não é permitido ao estudante se ausentar da sala antes da entrega da prova;
8. Celulares e outros equipamentos eletrônicos (exceto o notebook do aluno, se assim optar) devem permanecer desligados enquanto o estudante estiver na sala;
9. O tempo limite para realização da prova é de **90** minutos;
10. Mantenha sobre a carteira apenas um documento com foto, caneta, lápis e borracha;
11. O entendimento das questões faz parte da avaliação;
12. O tempo mínimo de permanência na sala é de **30** minutos;
13. O estudante que chegar atrasado em até **30** minutos do início da prova poderá fazê-la.

<u>Questão 1</u>: (*1 ponto*)

Árvores Binárias de Pesquisa (*ABP*) são estruturas que mantêm a ordem de seus elementos (chaves menores à esquerda, maiores à direita) para otimizar operações de busca, inserção e remoção. No entanto, o algoritmo de remoção é, tipicamente, o mais complexo, especialmente quando o nó a ser excluído possui dois filhos. O método escolhido para substituição é crucial para manter a propriedade fundamental da ABP e, consequentemente, a corretude de buscas futuras. A eficiência de todas as operações depende diretamente da altura da árvore (*h*).

Com base na bibliografia e no material estudado em aula, avalie a Asserção (A) e a Razão (R), bem como suas relações, dadas a seguir:

(A) A remoção de um nó *n* com dois filhos em uma *Árvore Binária de Pesquisa* (*ABP*) garante que a estrutura da árvore permaneça uma *ABP* válida, e a complexidade de tempo da operação, em média, permanece *O(log N)*, onde *N* é o número de nós.

(R) A estratégia de substituição para a remoção de um nó com dois filhos envolve a troca de *n* pelo seu sucessor em ordem (*in-order successor*), que é o nó com a menor chave na sub-árvore direita. Este sucessor é, por definição, um nó que possui no máximo um filho (o filho à direita), simplificando a remoção subsequente e custando *O(h)* para ser encontrado e removido.

A respeito da asserção e razão dadas, **assinale** a única opção **correta**:

a) Ambas, (A) e (R) são proposições verdadeiras, e (R) é uma justificativa correta de (A).

b) Ambas, (A) e (R) são proposições verdadeiras, mas (R) não é uma justificativa correta de (A).

c) A asserção (A) é uma proposição verdadeira, e a razão (R) é uma proposição falsa.

d) A asserção (A) é uma proposição falsa, e a razão (R) é uma proposição verdadeira.

e) Ambas, (A) e (R) são proposições falsas.

Questão 2: (*1 ponto*)

As *Tabelas Hash* (Tabelas de Dispersão) são a principal escolha para implementação de dicionários (mapeamentos) quando a velocidade é a prioridade, com o custo de uma maior exigência de memória e um pior caso de complexidade potencialmente problemático. A eficiência média é alcançada desde que a função *hash* distribua uniformemente as chaves e o fator de carga ($\alpha = N/M$) seja mantido baixo. A forma como as colisões são tratadas (Encadeamento Separado vs. Endereçamento Aberto) tem impacto direto no desempenho no cenário médio e, crucialmente, no pior caso.

Com base na bibliografia e no material estudado em aula, avalie a Asserção (A) e a Razão (R), bem como suas relações, dadas a seguir:

(A) O principal benefício de uma *Tabela Hash*, quando implementada com resolução de colisões por encadeamento separado (*separate chaining*) e um fator de carga (α) pequeno e constante, é atingir uma complexidade de tempo esperada de *O(1)* para as operações de busca, inserção e remoção.

(R) No cenário de pior caso (*worst-case*) para uma *Tabela Hash* implementada com encadeamento, onde todas as *N* chaves são mapeadas para o mesmo *bucket*, a complexidade de tempo para uma operação de busca degenera para *O(N)* , o que ainda é uma complexidade superior à complexidade média de *O(log N)* de uma *Árvore Binária de Pesquisa Balanceada* (como *AVL* ou *Red-Black*) no mesmo cenário.

A respeito da asserção e razão dadas, **assinale** a única opção **correta**:

a) Ambas, (A) e (R) são proposições verdadeiras, e (R) é uma justificativa correta de (A).

b) Ambas, (A) e (R) são proposições verdadeiras, mas (R) não é uma justificativa correta de (A).

c) A asserção (A) é uma proposição verdadeira, e a razão (R) é uma proposição falsa.

d) A asserção (A) é uma proposição falsa, e a razão (R) é uma proposição verdadeira.

e) Ambas, (A) e (R) são proposições falsas.

Questão 3: (*1 ponto*)

Em um grafo bipartido $G = (X \cup Y, E)$, onde X e Y são as partições de vértices, a questão de interesse prático é se é possível formar um *emparelhamento* que cubra todos os vértices do conjunto menor (assumindo $|X| \leq |Y|$), resultando em um *Emparelhamento Completo* (ou *Emparelhamento de X em Y*). O problema da atribuição (alocar tarefas a pessoas, por exemplo) depende fundamentalmente da verificação de uma condição teórica que garanta a existência da solução, antes de se recorrer a algoritmos de busca (como o *M-Aumentador*).

Considere um grafo bipartido $G = (X \cup Y, E)$, onde $|X| \leq |Y|$. Analise as afirmações a seguir, relativas à existência de um *Emparelhamento Completo de X em Y*, com base na bibliografia e no material estudado em aula:

I. A condição de *Hall* ($|N(S)| \geq |S|$ *para* todo $S \subseteq X$) é uma condição suficiente, mas não necessária, para garantir a existência de um emparelhamento completo de X em Y.

II. Um emparelhamento em G que cobre todos os vértices de X deve, obrigatoriamente, ser um *Emparelhamento Máximo* (aquele com o maior número possível de arestas).

III. Se a condição de *Hall* for violada para algum subconjunto $S \subseteq X$, ou seja, se $|N(S)| < |S|$, então não existe nenhum emparelhamento que cubra todos os vértices do conjunto X.

IV. O algoritmo que utiliza *Caminhos Aumentadores* (*Augmenting Paths*), como o algoritmo de *Hopcroft-Karp* ou o baseado em *Fluxo Máximo* (*Ford-Fulkerson*), é a principal ferramenta algorítmica para encontrar o emparelhamento, caso o *Teorema de Hall* garanta sua existência.

Assinale a alternativa que apresenta somente as **afirmações verdadeiras**:

a) I e IV, apenas.

b) II e III, apenas.

c) III e IV, apenas.

d) II, III e IV, apenas.

e) I, II, III e IV.

Questão 4: (*1 ponto*)

A *Teoria dos Grafos* fornece teoremas fundamentais para determinar a possibilidade de percorrer um *grafo* visitando cada *aresta* exatamente uma vez (*percursos Eulerianos*) ou visitando cada *vértice* exatamente uma vez (*percursos Hamiltonianos*). O *grau* dos *vértices* desempenha um papel central nos *percursos Eulerianos*. O *Teorema da Mão de Shake* garante que o número de *vértices* de *grau ímpar* em qualquer *grafo* deve ser sempre *par*.

Analise as afirmações a seguir, relativas aos percursos *Eulerianos* em um *grafo não-direcionado G = (V, E) conexo*, com base na bibliografia e no material estudado em aula:

I. Um *circuito Euleriano* é um ciclo que atravessa todas as arestas de *G* exatamente uma vez e, por definição, inicia e termina no mesmo vértice.

II. Se um *grafo conexo G* possui exatamente dois *vértices* de grau *ímpar*, qualquer *trilha Euleriana* deve, obrigatoriamente, iniciar em um desses *vértices* de grau *ímpar* e terminar no outro.

III. A existência de um *circuito Euleriano* em *G* é uma condição suficiente, mas não necessária, para a existência de uma *trilha Euleriana*.

IV. Pelo *Teorema de Euler*, se um *grafo conexo G* tem zero *vértices* de grau *ímpar*, ele deve ser percorrível por uma *trilha Euleriana*, mas não necessariamente por um *circuito Euleriano*.

Assinale a alternativa que apresenta somente as **afirmações verdadeiras**:

a) I e IV, apenas.

b) I, II e III, apenas.

c) I e II, apenas.

d) II, III e IV, apenas.

e) I e III, apenas.

Questão 5: (*1 ponto*)

Segundo o Teorema da Curva de *Jordan*, apresentado originalmente por *Camille Jordan* (1893), toda curva simples e fechada no plano — isto é, uma curva contínua que não se auto intersecta — divide o plano em exatamente duas regiões disjuntas: uma região interior, limitada pela curva, e uma região exterior, ilimitada. Esse resultado é fundamental em diversas áreas da computação, como computação gráfica, geometria computacional e análise de colisões, pois garante a separação do plano em duas partes bem definidas, mesmo quando a curva tem formato irregular.

Considere agora uma curva simples e fechada C definida no plano cartesiano, de forma que sua projeção delimita uma região interna composta por todos os pontos (x, y) que satisfazem uma relação contínua de contorno, sem auto intersecções, e uma região externa composta por todos os demais pontos.

Com base no Teorema da Curva de *Jordan* e nos conceitos apresentados e discutidos em aula, assinale a alternativa que melhor descreve a consequência lógica da aplicação desse teorema à curva C:

a) A curva C necessariamente separa o plano em uma região interior e uma exterior, ambas conexas e disjuntas, independentemente do formato da curva, desde que ela seja simples e fechada.

b) A curva C pode delimitar múltiplas regiões internas desde que o contorno seja contínuo e fechado.

c) A curva C só divide o plano em duas regiões se for uma curva convexa.

d) A curva C pode dividir o plano em mais de duas regiões desde que possua trechos diferenciáveis.

e) A curva C só garante duas regiões distintas se for definida por funções polinomiais contínuas.

Questão 6: (5 *pontos*)

Gerado por qualquer *IDE* fornecer o código fonte, em *Java*, das classes necessárias (incluindo a de execução), implementadas sob o paradigma da *Programação Orientada a Objetos*, além do programa ***.jar*** que as executa diretamente, objetivando resolver o problema descrito a seguir.

ATENÇÃO:

i. Se o programa *.jar* fornecido não executar automaticamente a aplicação desenvolvida, a resolução da questão será invalidada (0 ponto);

ii. As classes soluções desta questão (arquivos *.java*) deverão ser compactadas em um único arquivo (*.zip* ou *.rar*), em conjunto com o respectivo arquivo *.jar* funcional, além deste arquivo *.docx* da prova, contendo todas as respostas para as outras questões e a devida identificação do aluno e entregue em resposta à tarefa do *CanvasLMS* da prova.

Problema:

Utilizando apenas os códigos vistos e praticados em aula e os códigos desenvolvidos pelo próprio aluno para esta disciplina, através de uma aplicação implementada durante a prova, e utilizando os conceitos apresentados pela Teoria dos Grafos, deseja-se resolver o seguinte problema:

"Em uma clínica, através de um sistema computacional *online*, pacientes precisarão agendar consultas com os médicos que lá trabalham. Num determinado dia e horário (por exemplo, na próxima segunda-feira, às 9h), 5 (cinco) pacientes precisam escolher 1 (um) entre 6 (seis) médicos disponíveis de sua preferência. Cada paciente poderá escolher mais de um médico como opção alternativa".

Elabore um programa que receba, via digitação, cada um dos pacientes (*P1*, *P2*, *P3*, *P4* ou *P5*) a serem consultados e com qual(is) médico(s) (*M1*, *M2*, *M3*, *M4*, *M5* e *M6*) cada um desses pacientes deseja se consultar.

O sistema deverá acomodar a(s) preferência(s) de cada um dos 5 (cinco) pacientes para esse determinado dia e horário em uma **Estrutura de Dados GRAFO baseada em nós (alocação dinâmica)**.

A seguir, partindo do GRAFO gerado e preenchido, o sistema deverá apresentar em tela a situação geral dos agendamentos, **RESOLVENDO O PROBLEMA DE EMPARELHAMENTO**.

Como praticante da técnica de *TDD* já estudada anteriormente, a seguir estão definidos os casos de testes que deverão guiar o programador na elaboração do programa solicitado:

Casos de Teste:

CASO DE TESTE 1:

Entradas	Digitação
P1:	*M1, M2*
P2:	*M1, M3*
P3:	*M3*
P4:	*M4*
P5:	*M6*

Saída

Clínica Tabajara
Agenda de Alocação de Consultas
Dia: *Próxima 2a feira, 9h00min*

Médicos:	M1	M2	M3	M4	M5	M6
Pacientes:	*P2*	*P1*	*P3*	*P4*	-	*P5*

CASO DE TESTE 2:

Entradas	Digitação
P1:	*M2*
P2:	*M1, M3*
P3:	*M3*
P4:	*M3, M5*
P5:	*M5*

Saída *NÃO HÁ EMPARELHAMENTO POSSÍVEL!*

Clínica Tabajara
Agenda de Alocação de Consultas
Dia: *Próxima 2a feira, 9h00*

Médicos:	M1	M2	M3	M4	M5	M6
Pacientes:	*P2*	*P1*	*P2, P3, P4*	-	*P4, P5*	