Prova I - Teoria

Entrega 18 abr em 10:30 **Pontos** 10 **Perguntas** 14
Disponível 18 abr em 8:50 - 18 abr em 10:30 1 hora e 40 minutos
Limite de tempo 100 Minutos

Instruções

Nossa primeira prova de Algoritmos e Estruturas de Dados II tem duas partes: teórica e prática. Cada uma vale 10 pontos. A prova teórica será realizada no Canvas e a prática foi no Verde.

A prova teórica tem 14 questões. A primeira questão é uma autoavaliação da cola autorizada para esta prova e vale 0,5 ponto. Em seguida, temos 11 questões fechadas e 2 abertas. Cada fechada vale 0.5 pontos e cada aberta, 2 pontos. No caso das questões abertas, o aluno poderá enviar um arquivo contendo sua resposta ou entregá-la em na folha em branco disponibilizada pelo professor.

Abaixo, seguem as regras para a prova.
1) O código da prova será fornecido pelo professor no início da prova.
2) Após o envio de uma questão **não** é permitido que o aluno volte na mesma.
3) A prova é individual e é permitida a consulta à cola que contém o nome do aluno.
4) A interpretação faz parte da prova.
5) Se existir algum erro, após a divulgação do gabarito, peça a anulação da questão.
6) Os alunos da turma 1/manhã farão a prova nos labs 1 e 2.
7) Os alunos da turma 2/manhã farão a prova nos labs 8 e 9.
8) Os alunos da tarde farão a prova no lab 11.

Desejo uma excelente prova para todos.

Este teste não está mais disponível, pois o curso foi concluído.

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	Tentativa 1	97 minutos	4,5 de 10

ⓘ As respostas corretas não estão mais disponíveis.

Pontuação deste teste: **4,5** de 10
Enviado 18 abr em 10:30
Esta tentativa levou 97 minutos.

Parcial

Pergunta 1

0,5 / 0,5 pts

Autoavaliação sobre sua preparação para esta prova (mínimo 0 e máximo 0,5).

- [ ] 0,3
- [ ] 0,4
- [ ] 0,2
- [x] 0,5
- [ ] 0
- [ ] 0,1

Incorreta

Pergunta 2

0 / 0,5 pts

Um desafio no projeto de algoritmos é a obtenção de um custo computacional reduzido. Para isso, uma habilidade do projetista é contar o número de operações realizadas pelo algoritmo. O trecho de código abaixo realiza algumas operações.

```
for (int i = 0; i < n; i++){
    for (int j = n; j > 1; j /= 2){
        l = a * 2 + b * 5;
    }
}
```

Considerando o código acima, assinale a opção que apresenta a função de complexidade *f(n)* para o melhor e pior caso considerando a operação de multiplicação.

- $f(n) = 2 \times n \times lg(n)$
- $f(n) = 2 \times n \times (n-1)$

- $f(n) = 2 \times n^2$
- $f(n) = 2 \times n \times \lfloor lg(n-1) \rfloor$
- $f(n) = 2 \times n \times \lfloor lg(n) \rfloor$

Pergunta 3

0,5 / 0,5 pts

Um desafio no projeto de algoritmos é a obtenção de um custo computacional reduzido. Para isso, o projetista de algoritmos deve ser capaz de contar o número de operações realizadas em seus algoritmos. O trecho de código abaixo realiza algumas operações.

```
for (int i = 0; i < n ; i++){
    for (int j = 0; j < n - 1; j++){
        l = a * 2 + b * 5;
    }
}
```

Considerando o código acima, assinale a opção que apresenta a função de complexidade *f(n)* para o melhor e pior caso considerando a operação número de multiplicações.

- $f(n) = 2 \times n \times lg(n)$
- $f(n) = 2 \times n \times (n-1)$
- $f(n) = 2 \times (n-2) \times (n-1)$
- $f(n) = 2 \times n^2$
- $f(n) = 2 \times n \times (n+1)$

Incorreta

Pergunta 4

0 / 0,5 pts

A contagem do número de operações realizadas por um algoritmo é uma tarefa fundamental para identificar seu custo computacional. O trecho de código abaixo realiza algumas operações.

```
Random gerador = new Random();
gerador.setSeed(4);

for (int i = 4; i < n; i++){
    if(Math.abs(gerador.nextInt()) % 9 < 4){
        a *= 2; b *= 3; l *= 2;
    } else if (Math.abs(gerador.nextInt()) % 9 == 5) {
        a *= 2; l *= 3;
    } else if (Math.abs(gerador.nextInt()) % 9 > 5) {
        a *= 2;
    }
}
```

Considerando o código acima, marque a opção que apresenta o pior e melhor caso para o número de multiplicações, respectivamente.

- n, n
- (n-4), 0
- (n-4), (n-4)
- 3(n-4), 0
- 3(n-4), (n-4)

O laço é executado (n-4) vezes dado que o contador começa com 4, incrementado de um em um e repetido enquanto for menor que n. O pior caso acontece quando o número aleatório sempre é menor do que 4 fazendo com que a cada passo do laço execute 3 multiplicações. O melhor caso acontece quando o número gerado sempre é 4 fazendo com que nenhuma multiplicação seja efetuada.

Pergunta 5

0,5 / 0,5 pts

Os operadores lógicos *and* e *or* são primordiais na confecção de software. Dadas duas ou mais condições de entrada, a saída do operador *and* é verdadeira quando todas as condições de entrada também são. A saída do operador *or* é verdadeira quando pelo menos uma das entradas é verdadeira. O trecho de código abaixo realiza operações lógicas dentro de uma estrutura condicional.

```
if (n < a + 3 && n > b + 4 && n > c + 1){
    l+= 5;
} else {
    l+= 2; k+=3; m+=7; x += 8;
}

if (n <= b + 4){
    l+= 2; k+=3; m+=7; x += 8;
} else {
    l+= 5;
}
```

Considerando o código acima, marque a opção que apresenta o melhor e pior caso, respectivamente, para o número de adições.

- 6 e 10
- 5 e 9
- 6 e 12
- 4 e 10
- 4 e 9

O pior caso tem 12 adições e acontece quando, apenas a última condição do primeiro if é falsa. Consequentemente, a condição única do segundo if é verdadeira dado que ela é inversa a segunda condição do primeiro if. Dessa forma, o teste do primeiro if realiza 3 adições e sua lista de comandos, quatro. O teste do segundo if realiza uma adição e mais quatro na lista do else.

O melhor tem 6 adições e isso acontece quando todas as condições do primeiro if são verdadeiras. Nesse caso, o teste do primeiro if realiza 3 adições e sua lista de comandos, mais uma. No segundo if, a condição será falsa dado que ela é inversa a segunda condição do primeiro if. Temos portanto uma adição do teste e mais uma da lista do else.

Pergunta 6

0,5 / 0,5 pts

Sejam $T_1(n) = 100n + 15$, $T_2(n) = 10n^2 + 2n$ e $T_3(n) = 0,5n^3 + n^2 + 3$ as equações que descrevem a complexidade de tempo dos algoritmos Alg1, Alg2 e Alg3, respectivamente, para entradas de tamanho n. A respeito da complexidade desses algoritmos, pode-se concluir que (POSCOMP'15 - adaptada):

- Alg2 e Alg3 são $\Theta(n^2)$
- Alg1, Alg2 e Alg3 são, respectivamente, $O(100)$, $O(10)$ e $O(0,5)$
- Alg1, Alg2 e Alg3 são, respectivamente, $O(n)$, $O(n^2)$ e $O(n^3)$.
- Alg1, Alg2 e Alg3 são, respectivamente, em $O(n)$, $O(n^2)$ e $O(n^2)$.
- Alg1 e Alg2 são $\Theta(n^2)$

Realmente, Alg1, Alg2 e Alg3 são, respectivamente, $O(n)$, $O(n^2)$ e $O(n^3)$.

Incorreta

Pergunta 7

0 / 0,5 pts

Um dos problemas mais importantes na Computação é pesquisar a existência de um elemento em um conjunto de dados. Duas técnicas tradicionais de pesquisa são a sequencial e a binária. A respeito dessas técnicas assinale a alternativa correta (POSCOMP'12 - adaptado).

- A pesquisa binária pode ser feita sobre qualquer distribuição dos elementos.
- A pesquisa binária percorre, em média, a metade dos elementos do vetor.
- A pesquisa sequencial exige que os elementos estejam completamente ordenados.
- A pesquisa binária percorre no pior caso $O\left(\lg(n)\right)$ elementos.
- A pesquisa sequencial percorre todos os elementos para encontrar a chave.

Realmente, a pesquisa binária percorre no pior caso $O\left(\lg(n)\right)$ elementos.

Incorreta

Pergunta 8

0 / 0,5 pts

A ordenação interna é um problema clássico na Computação. Considerando-o, avalie as asserções que se seguem:

I. O algoritmo Countingsort ordena um vetor com custo linear.

PORQUE

II. O limite inferior do problema de ordenação interna é $\Theta(n \times lg\, n)$ para a comparação entre registros.

A respeito dessas asserções, assinale a opção correta.

- A asserção I é uma proposição verdadeira, e a asserção II é uma proposição falsa
- A asserção I é uma proposição falsa, e a asserção II é uma proposição verdadeira
- As asserções I e II são proposições verdadeiras, mas a segunda não é uma justificativa correta da primeira
- As asserções I e II são proposições verdadeiras, e a segunda é uma justificativa correta da primeira
- As asserções I e II são proposições falsas

I - CORRETA: O algoritmo Countingsort efetua em tempo linear $\Theta(n)$ a ordenação dos elementos de um vetor. Ele considera três vetores: entrada, contagem e saída. O primeiro passo é em $\Theta(n)$ é criar o vetor de contagem de tal forma que cada posição tenha o número de elementos menores ou iguais aquela posição. O segundo passo é copiar cada elemento do vetor de entrada para o de saída mapeando de tal forma que a posição do elemento no vetor de saída será mapeada a partir do vetor de contagem.

II - CORRETA: É impossível ordenar um vetor com menos do que $\Theta(n \times lg\, n)$ comparações entre os elementos do vetor. O Countingsort não se aplica a tal regra porque ele triplica o espaço de memória e não funciona para qualquer tipo de elemento.

As duas afirmações são independentes.

Pergunta 9

0,5 / 0,5 pts

A primeira fase do heapsort constroi um *heap* com os elementos do vetor. Seja o vetor [20, 10, 5, 30, 50, 45, 35] onde o primeiro elemento (20) está na posição 1, assinale a opção que contém o heap construído no final da fase citada (INMETRO'10, adaptada).

- (●) [50, 30, 45, 10, 20, 5, 35]
- ( ) [5, 10, 20, 30, 35, 45, 50]
- ( ) [50, 20, 45, 30, 10, 5, 35]
- ( ) [20, 10, 30, 5, 15, 45, 50]
- ( ) [50, 45, 35, 30, 20, 15, 10]

Aplicando o algoritmo do Heapsort, temos a sequência de resposta.

Incorreta

Pergunta 10

0 / 0,5 pts

Um algoritmo clássico para a ordenação de elementos de um vetor é o Ordenação por Seleção. Abaixo, temos uma implementação desse algoritmo.

```
for (i = 0; i < (n - 1); i++) {
    indice = i;
    for (j = (i + 1); j < n; j++){
        if (vet[indice] > vet[j]){
            indice = j;
        }
    }
    tmp = vet[indice];
    vet[indice] = vet[i];
    vet[i] = tmp;
}
```

Considerando o código acima e seus conhecimentos sobre algoritmos de ordenação, avalie as asserções que se seguem:

I. O algoritmo de Seleção deve ser considerado como uma opção para ordenação quando tivermos registros grandes".

PORQUE

II. O algoritmo de Seleção possui complexidade quadrática - $O(n^2)$ - para o número de comparações envolvendo os elementos do vetor.

A respeito dessas asserções, assinale a opção correta.

As asserções I e II são proposições verdadeiras, mas a segunda não é uma justificativa correta da primeira.

As asserções I e II são proposições verdadeiras, e a segunda é uma justificativa correta da primeira.

A asserção I é uma proposição verdadeira, e a asserção II é uma proposição falsa.

As asserções I e II são proposições falsas.

A asserção I é uma proposição falsa, e a asserção II é uma proposição verdadeira.

Realmente, as duas afirmações são verdadeiras, contudo, um não justifica a outra.

Pergunta 11

0,5 / 0,5 pts

Sobre a escolha adequada para um algoritmo de ordenação, considere as afirmativas a seguir.

I. Quando os cenários de pior caso for a preocupação, o algoritmo ideal é o Heap Sort.
II. Quando o vetor apresenta a maioria dos elementos ordenados, o algoritmo ideal é o Insertion Sort.
III. Quando o interesse for um bom resultado para o médio caso, o algoritmo ideal é o Quick Sort.
IV. Quando o interesse é o melhor caso e o pior caso de mesma complexidade, o algoritmo ideal é o
Bubble Sort.

Assinale a alternativa correta

Somente as afirmativas II, III e IV são corretas.

Somente as afirmativas I, II e III são corretas.

Somente as afirmativas III e IV são corretas

Somente as afirmativas I e IV são corretas.

Somente as afirmativas I e II são corretas.

(POSCOMP'13)

Incorreta

Pergunta 12

0 / 0,5 pts

Considere o seguinte algoritmo de ordenação de elementos em uma lista (IBGE'13):

I. Escolha um elemento que será chamado o pivot da lista.
II. Reordene a lista de tal forma que os elementos menores que o pivot venham antes dele e os elementos maiores ou iguais ao pivot venham depois dele. Essa operação é chamada de partição, e cria duas sublistas: a de menores que o pivot e a de maiores ou iguais ao pivot.
III. Aplique recursivamente os passos 1 e 2 às sublistas de menores e maiores que o pivot.

O algoritmo acima corresponde ao

Quicksort, e faz, em média, $O\left(n^2\right)$ comparações para ordenar n itens.

Insertionsort, e faz, em média, $O\left(n\right)$ comparações para ordenar n itens.

Quicksort, e faz, em média, $O\left(n \times \lg(n)\right)$ comparações para ordenar n itens.

Insertionsort, e faz, em média, $O\left(n \times \lg(n)\right)$ comparações para ordenar n itens.

Não respondida

Pergunta 13

1 / 2 pts

Encontre a fórmula fechada do somatório $\sum_0^n (2i+5)^2$ e, em seguida, prove a usando indução matemática.

A ser explicada na aula posterior à prova.

A aluna errou na fórmula fechada.

Não respondida

Pergunta 14

0,5 / 2 pts

Uma desvantagem do algoritmo de **Inserção** é que quando ele insere um novo elemento na parte ordenada, ele efetua uma **pesquisa sequencial** para encontrar a posição do novo elemento. Explique e apresente uma proposta de melhoria para o algoritmo. Implemente e apresente a complexidade de sua solução.

A melhoria a ser proposta é a utilização da pesquisa binária para inserir o elemento na posição correta da parte ordenada do vetor. Atenção para o custo da pesquisa binária que é $O(log(n))$.

O enunciado pede: Implemente e apresente a complexidade de sua solução.

Pontuação do teste: **4,5** de 10