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 foi travado 18 abr em 10:30.

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	Tentativa 1	84 minutos	5,9 de 10

ⓘ As respostas corretas não estão mais disponíveis.

Pontuação deste teste: **5,9** de 10
Enviado 18 abr em 10:16
Esta tentativa levou 84 minutos.

Parcial

Pergunta 1

0,5 / 0,5 pts

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

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

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 = 1; i <= n; i *= 2){
  a += 3;
}
```

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 adição.

- $f(n) = \lceil lg(n) \rceil + 1$
- $f(n) = n$
- $f(n) = n - 1$

- [x] $f(n) = lg(n)$
- [ ] $f(n) = \lfloor lg(n) \rfloor + 1$

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 = 2; i < n; i++){
    for (int j = 2; j <= n; 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.

- [ ] $f(n) = 2 \times n^2$
- [ ] $f(n) = 2 \times n \times \lfloor lg(n-1) \rfloor$
- [ ] $f(n) = \ 2 \ \times n \ \times (n+1)$
- [ ] $f(n) = 2 \times (n-2) \times (n-2)$
- [x] $f(n) = 2 \times (n-1) \times (n-2)$

Pergunta 4

0,5 / 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);
int x = Math.abs(gerador.nextInt());

for (int i = 4; i < n; i++){
    if( x % 9 < 4) {
        a *= 2; b *= 3; l *= 2;
    } else if (x % 9 == 5) {
        a *= 2; l *= 3;
    } else  {
        a *= 2;
    }
}
```

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

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

Incorreta

Pergunta 5

0 / 0,5 pts

O comando condicional *if-else* possibilita a escolha de um grupo de ações a serem executadas quando determinadas condições de entrada são ou não satisfeitas. O trecho de código abaixo contém 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 > c + 1){
    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 12
- ( ) 5 e 9
- ( ) 6 e 9
- ( ) 4 e 12
- ( ) 4 e 9

O pior caso tem nove adições e acontece quando as três condições do primeiro if são falsas e, consequentemente, a condição única do segundo if é falsa dado que ela é igual a primeira 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 uma na lista do else.

O melhor tem quatro adições e isso acontece quando a primeira condição é verdadeira e a segunda é falsa. Nesse caso, o teste do primeiro if realiza uma adição e sua lista de comandos, uma. No segundo if, temos uma adição do teste e mais uma da lista do else.

Incorreta

Pergunta 6

0 / 0,5 pts

O desempenho pode ser visto como um conjunto de características, capacidades e rendimentos de um elemento ou grupo comparado com metas, requisitos ou expectativas previamente definidos. Quando desenvolvemos um algoritmo, nós não podemos avaliar seu

desempenho medindo seu tempo de execução dado que essa medida é afetada por diversos fatores tais como linguagem de implementação, arquitetura do computador, compilador e sistema operacional. Uma das formas de efetuar essa avaliação é criando uma função matemática que considera as operações mais relevantes realizadas pelo algoritmo. Dado o algoritmo abaixo, avalie as afirmações sobre as operações realizadas pelo mesmo.

```
int metodo(int[] array, int b){
   int resp = 1, n = array.length;

   if(n > 10) resp = array[0] * b;
   else resp = b;

   for(int i = 0; i < n; i++){
      resp--;
      b-= array[i];
      array[0] *= 2;
   }
   resp += b;
   return resp;
}
```

O algoritmo acima realiza:
I. o mesmo número de multiplicações no melhor e o pior caso.
II. n subtrações no pior caso.
III. $O(n)$ multiplicações no pior caso.
IV. $O(n^2)$ multiplicações no pior caso.
V. $(n+2)$ comparações no pior e melhor caso.

É correto apenas o que se afirma em:

- III, IV e V
- I, III e V
- II e III
- I e V

I e IV

I - ERRADA. No pior caso, quando n > 10, o algoritmo realiza n + 1 multiplicações. No melhor, ele realiza n multiplicações.

II- ERRADA. Em todos os casos, o algoritmo realiza 2n subtrações.

III - CORRETA. No pior caso, o algoritmo realiza (n +1) multiplicações o que é O(n).

IV - CORRETA. No pior caso, o algoritmo realiza (n +1) multiplicações o que é O(n) e, consequentemente, $O\left(n^2\right)$.

V - CORRETA. Em todos os casos, o algoritmo realiza (n+2) comparações. Uma comparação no if, n comparações verdadeiras no for e uma comparação falsa no for.

Incorreta

Pergunta 7

0 / 0,5 pts

Sabendo que 32 times se classificaram para a Copa do Mundo de Futebol e seus nomes foram colocados de forma ordenada em um *array*. Se aplicarmos a busca binária para encontrar um nome, no máximo (pior caso), quantos itens do *array* teremos que examinar?

- 16
- 32
- 5
- 6
- 1

No pior caso, faremos seis comparações. A primeira divide o conjunto em dois de 16. A segunda (em cada partição), em dois de 8. A terceira, em dois de 4. A quarta, em dois de 2. A quinta em dois de 1. A última verifica a única opção restante.

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 falsa, e a asserção II é uma proposição verdadeira
- 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

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

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.

Incorreta

Pergunta 9

0 / 0,5 pts

O Heapsort é um algoritmo de ordenação clássico que utiliza uma estrutura de *heap* invertido para ordenar os elementos do vetor. A primeira etapa desse algoritmo organiza todos os elementos do vetor em um *heap* invertido, estrutura de dados na qual todos os elementos são maiores ou iguais aos seus filhos. A segunda etapa remove a cabeça do *heap* (maior elemento), reduzindo o tamanho do mesmo em uma unidade. Em seguida, a posição que ficou livre recebe o elemento removido. A segunda etapa repete esse processo até que o *heap* tenha somente um elemento, o menor de todos. Considerando a descrição anterior e seus conhecimentos sobre algoritmos de ordenação, avalie as asserções que se seguem:

I. A ordem de complexidade do Heapsort é $O(n \times lg\, n)$ para todos os casos.

PORQUE

II. A ordem de complexidade da primeira etapa é $O(n)$ no melhor caso e $O(n \times lg\, n)$ no pior caso e a da segunda etapa é $O(n \times lg\, n)$ para todos os casos.

A respeito dessas asserções, assinale a opção correta.

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

As duas afirmações são verdadeiras e a segunda, realmente, justifica a primeira.

Pergunta 10

0,5 / 0,5 pts

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

```
for (i = 1; i < n; i++) {
  tmp = vet[i];
  int j = i - 1;

  while ((j >= 0) && (vet[j] > tmp)) {
    vet[j + 1] = vet[j];
    j--;
  }
  vet[j + 1] = tmp;
}
```

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

I. O algoritmo de Inserção deve ser usado quando os elementos do vetor estão ordenados ou quase ordenados.

PORQUE

II. O algoritmo de Inserção apresenta complexidade linear - $O(n)$ - em seu melhor caso dado que cada novo elemento inserido no subconjunto ordenado será comparado apenas com um elemento desse subconjunto.

A respeito dessas asserções, assinale a opção correta.

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

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

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

As duas alternativas estão corretas e a segunda, realmente, justifica a primeira.

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 III e IV são corretas

Somente as afirmativas I e II são corretas.

Somente as afirmativas I e IV são corretas.

Somente as afirmativas II, III e IV são corretas.

Somente as afirmativas I, II e III são corretas.

(POSCOMP'13)

Pergunta 12

0,5 / 0,5 pts

O Quicksort é um dos principais algoritmos de ordenação entre suas vantagens estão sua simplicidade e o fato desse algoritmo realizar para o melhor caso e o caso médio $O\left(n \times \lg(n)\right)$ comparações entre os elementos da lista a ser ordenada. Sobre esse algoritmo, avalie as asserções que se seguem:

I. O Quicksort apresenta ordem de complexidade quadrática - $O\left(n^2\right)$ - para seu pior caso em termos de número de comparações envolvendo os elementos da lista.

PORQUE

II. A eficácia do Quicksort depende da escolha do pivô mais adequado para o conjunto de dados que se deseja ordenar. A pior situação ocorre quando o pivô escolhido corresponde sistematicamente ao maior ou menor elemento do conjunto a ser ordenado.

A respeito dessas asserções, assinale a opção correta

As asserções I e II são proposições falsas.

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

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.

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

Realmente, as duas afirmações são verdadeiras e a segunda justifica a primeira.

io respondida

Pergunta 13

1,5 / 2 pts

Encontre a fórmula fechada do somatório $\sum_0^n (5i - 3)^2$ e, em seguida, prove a usando indução matemática.

A ser explicada na aula posterior à prova.

A aluna errou no quadrado perfeito, contudo, ela sabe fazer indução.

io respondida

Pergunta 14

1,4 / 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 $O(log(n))$.

Recomendo implementar o código para identificar os problemas. A análise de complexidade está errada.

Pontuação do teste: **5,9** de 10