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	93 minutos	5,6 de 10

ⓘ As respostas corretas não estão mais disponíveis.

Pontuação deste teste: **5,6** de 10
Enviado 18 abr em 10:26
Esta tentativa levou 93 minutos.

Parcial

Pergunta 1

0,5 / 0,5 pts

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

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

Pergunta 2

0,5 / 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.

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

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

Incorreta

Pergunta 3

0 / 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; 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.

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

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);
int x = Math.abs(gerador.nextInt());

for (int i = 0; i <= n-5; 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), (n-4)
- 3(n-5), (n-5)
- 3(n-5), 1
- n, n
- 3(n-4), 1

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 >= a + 3){
    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
- ( ) 6 e 12
- ( ) 4 e 12
- ( ) 4 e 10
- ( ) 5 e 12

O pior caso tem 10 adições e acontece quando a primeira condição do primeiro if é falsa e, consequentemente, a condição única do segundo if é verdadeira dado que ela é inversa a primeira condição do primeiro if. Dessa forma, o teste do primeiro if realiza 1 adição 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 as três condições do primeiro if são verdadeiras. Nesse caso, o teste do primeiro if realiza três adições e sua lista de comandos, uma. No segundo if, temos 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):

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

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 percorre, em média, a metade dos elementos do vetor.

A pesquisa binária percorre no pior caso $O(\lg(n))$ elementos.

A pesquisa sequencial percorre todos os elementos para encontrar a chave.

A pesquisa sequencial exige que os elementos estejam completamente ordenados.

A pesquisa binária pode ser feita sobre qualquer distribuição dos elementos.

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

- [x] 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
- [ ] 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

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

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

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

Pergunta 10

0,5 / 0,5 pts

Considere o algoritmo de ordenação abaixo (TRT-SP'13, adaptada):

```
for(int i = 0; i < n-1; i++){
   int pos = i;
   for(int j = i + 1; j < n; j++){
      if(v[j] < v[pos){
         pos = j;
      }
   }
   int aux = v[pos];
   v[pos] = v[j];
   v[j] = aux;
}
```

Sendo **n** o número de elementos do vetor, qual é o algoritmo

apresentado, quantas comparações e movimentações entre elementos do vetor ele realiza?

- (●) Algoritmo de Seleção, $O\left(n^2\right)$ comparações e $O\left(n\right)$ movimentações.
- ( ) Algoritmo de Seleção, $O\left(n^2\right)$ comparações e $O\left(n^2\right)$ movimentações.
- ( ) Algoritmo de Inserção, $O\left(n^2\right)$ comparações e $O\left(n^2\right)$ movimentações
- ( ) Algoritmo de Inserção, $O\left(n^2\right)$ comparações e $O\left(n\right)$ movimentações.
- ( ) Algoritmo de Inserção, $O\left(n\ \times \lg(n)\right)$ comparações e $O\left(n\ \times \lg(n)\right)$ movimentações.

O algoritmo em questão é o Seleção que realiza $f\left(n\right)\ =\ \frac{\{n^2\}}{2}\ +\ \frac{\{n\}}{2}$ comparações entre elementos do vetor e $f\left(n\right)\ =\ n\ -1$ movimentações.

Pergunta 11

0,5 / 0,5 pts

Quais destes algoritmos de ordenação têm a classe de complexidade assintótica, no pior caso, em $O\left(n\ \times \lg(n)\right)$.

- ( ) QuickSort, MergeSort e SelectionSort
- ( ) QuickSort e BubbleSort
- (●) MergeSort e HeapSort
- ( ) QuickSort e SelectionSort

No answer text provided.

QuickSort, MergeSort e HeapSort

O pior caso do Quicksort, Selectionsort e Bubblesort são $O\left(n^2\right)$ comparações entre registros. Por outro lado, no caso do Quicksort, seu melhor e caso médio fazem $O\left(n \times \lg(n)\right)$ comparações entre registros (POSCOMP'15).

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

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

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.

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

2 / 2 pts

Encontre a fórmula fechada do somatório $\sum_1^n (3i+2)^2$ e, em seguida, prove a usando indução matemática

A ser explicada na aula posterior à prova.

ok

Pergunta 14

0,1 / 2 pts

Suponha a classe **Pilha** vista na sala e crie o método **R.E.C.U.R.S.I.V.O** *boolean isFibonacciRecursivo(int i)* que recebe como parâmetro um contador *i* e retorna *true* quando os *n* elementos existentes na pilha correspondem aos *n* primeiros termos da sequência de Fibonacci. Observe que seu código não deve conter os comandos de repetição *for*, *while* e *do-while*.

prova.txt
(https://pucminas.instructure.com/files/5867884/download)

Considerando a estrutura definida, o aluno deve implementar o mostrar com a verificação se os dois primeiros são valores **1** e se os demais correspondem a soma dos dois anteriores.

Considerando a estrutura definida, o aluno deve implementar o mostrar com a verificação se os dois primeiros são valores 1 e se os demais correspondem a soma dos dois anteriores.

Pontuação do teste: **5,6** de 10