Prova I

Entrega 28 de set de 2020 em 10:40 **Pontos** 10,4 **Perguntas** 13
Disponível 28 de set de 2020 em 8:50 - 28 de set de 2020 em 10:40 aproximadamente 2 horas
Limite de tempo 110 Minutos

Instruções

Nossa primeira prova de AEDs II tem duas partes: teórica e prática. Cada uma vale 10 pontos. A prova teórica será realizada no Canvas e a prática, no Verde. A teórica será neste momento. A parte prática será realizada na aula de laboratório desta semana.

A prova teórica terá 11 questões fechadas no valor de 0,4 pontos e, em seguida, duas abertas no valor de 3 pontos cada. No total, distribuímos 0,4 pontos a mais. Após terminar uma questão, o aluno não terá a opção de retornar à mesma. No caso das questões abertas, o aluno poderá enviar um arquivo contendo sua resposta. Essa resposta pode ser uma imagem (foto de boa resolução) ou um código fonte. No horário da aula, você pode acessar a Prova I através do menu "Testes / Prova I".

Após a prova, para cada questão aberta, o aluno deve fazer um vídeo de no MÁXIMO 2 minutos explicando sua resposta. Vídeos com mais de 2 minutos serão penalizados. Cada vídeo deverá ser postado no YouTube e seu link submetido no Canvas até às 23:59 no dia 28/9. Vídeos alterados após essa data/horário serão zerados. Questões sem o vídeo serão penalizadas. Você pode acessar a submissão dos vídeos através do menu "Testes / Prova I - submissão de vídeos".

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

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	Tentativa 1	107 minutos	3,9 de 10,4

ⓘ As respostas corretas não estão mais disponíveis.

Pontuação deste teste: **3,9** de 10,4
Enviado 28 de set de 2020 em 10:39
Esta tentativa levou 107 minutos.

Pergunta 1

0,4 / 0,4 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.

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

Pergunta 2

0,4 / 0,4 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.

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

Pergunta 3

0,4 / 0,4 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), 1
- n, n
- 3(n-4), 1
- 3(n-5), (n-5)

Incorreta

Pergunta 4

0 / 0,4 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 9
- (selected) 4 e 12
- 4 e 9
- 6 e 12
- 5 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 5

0 / 0,4 pts

Uma das principais estruturas de dados é a matriz que possui diversas aplicações na Computação como, por exemplo, em processamento de imagens e vídeos, otimização de sistemas e teoria dos grafos. O código abaixo realiza a multiplicação entre duas matrizes quadradas.

```
for (i = 0; i < n; i++)
   for (j = 0; j < n; j++)
      for (k = 0; k < n; k++)
         c[i][j] += a[i][k] * b[k][j];
```

Sobre o código acima é correto afirmar:

I. Se alterarmos o primeiro laço para *for(i = n - 1; i >= 5; i--)*, a função de complexidade do algoritmo será mantida.
II. Se alterarmos o primeiro laço para *for(i = n - 1; i >= 5; i--)*, a ordem de complexidade do algoritmo será mantida.
III. Sua função de complexidade para o número de multiplicações envolvendo os elementos do vetor é $f(n) = n^3$.

É correto o que se afirma em

- III, apenas.

I, apenas.

II e III, apenas.

I, II e III.

I e II, apenas

A função de complexidade para o número de multiplicações envolvendo os elementos do vetor é $f(n) = n^3 = O(n^3)$.

I. ERRADA - A alteração fará com que a função de complexidade seja $f(n) = (n-5)n^2$

II. CORRETA - A alteração fará com que a função de complexidade seja $f(n) = (n-5)n^2 = O(n^3)$

III. CORRETA.

Pergunta 6

0,4 / 0,4 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 sequencial percorre todos os elementos para encontrar a chave.

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

A pesquisa sequencial exige que os elementos estejam completamente ordenados.

A pesquisa binária percorre, em média, a metade dos elementos do vetor.

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

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

Incorreta

Pergunta 7

0 / 0,4 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.

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

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

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 8

0 / 0,4 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).

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

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

Pergunta 9

0,4 / 0,4 pts

O algoritmo a seguir recebe um vetor v de números inteiros e rearranja esse vetor de tal forma que seus elementos, ao final, estejam ordenados de forma crescente.

```
void ordena(int *v, int n)

{
  int i, j, chave;
  for(i = 1; i < n; i++)
  {
    chave = v[i];
    j = i - 1;
    while(j >= 0 && v[j] < chave)
    {
      v[j-1] = v[j];
```

```
      j = j - 1;
    }
    v[j+1] = chave;
  }
}
```

Considerando que nesse algoritmo há erros de lógica que devem ser corrigidos para que os elementos sejam ordenados de forma crescente, assinale a opção correta no que se refere as correções adequadas (ENADE'17).

- No answer text provided.
- A linha 7 deve ser corrigida da seguinte forma: *j = i + 1;* e a linha 8, do seguinte modo: *while(j >= 0 && v[j] > chave)*.
- **A linha 8 deve ser corrigida da seguinte forma: *while(j >= 0 && v[j] > chave)* e a linha 10, do seguinte modo: *v[j+1] = v[j];*.**
- A linha 10 deve ser corrigida da seguinte forma: *v[j+1] = v[j];* e a linha 13, do seguinte modo: *v[j-1] = chave;*.
- A linha 4 deve ser corrigida da seguinte forma: *for(i = 1; i < n-1; i++)* e a linha 13, do seguinte modo *v[j-1] = chave;*.

O algoritmo em questão é o Inserção e os erros apresentados estão no laço interno. Na versão correta, o elemento a ser inserido na parte ordenada (chave) é comparado com os já ordenados (posições 0 à i). Enquanto o elemento chave é maior que os já ordenados, deslocamos cada elemento já comparado uma posição a mais do que sua posição atual, permitindo a inserção do elemento chave em sua posição correta. Essa inserção acontece após o laço interno.

Pergunta 10

0,4 / 0,4 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 I e IV são corretas.
- Somente as afirmativas I, II e III são corretas.
- Somente as afirmativas II, III e IV são corretas.
- Somente as afirmativas III e IV são corretas
- Somente as afirmativas I e II são corretas.

(POSCOMP’13)

Pergunta 11

0,4 / 0,4 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.
- 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.
- Insertionsort, e faz, em média, $O\left(n\right)$ comparações para ordenar n itens.

Pergunta 12

1 / 3 pts

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

somatorioprova.txt (https://pucminas.instructure.com/files/1955089/download)

A ser explicada na aula posterior à prova.

Enunciado não atendido.

Pergunta 13

0,1 / 3 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.

Nesta questão você deve apresentar o código fonte da solução e explicá-lo via vídeo ou áudio. Ao final, junte os dois arquivos em um zip e entregue fazendo upload.

provainsertionsort.txt (https://pucminas.instructure.com/files/1955203/download)

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 $O(log(n))$.

Enunciado não atendido.

Pontuação do teste: **3,9** de 10,4