

COMENTÁRIOS	ARQUIVOS	RUBRICA
III	0	<





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.

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.

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

$$\bigcirc f(n) = \lfloor lg(n) \rfloor + 1$$

$$f(n) = n$$

$$f(n) = n + 1$$



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

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.

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

$$f(n) = 2 \times n^2$$

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

$$\bigcirc f(n) = 2 \times n \times (n+1)$$

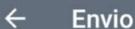
COMENTÁRIOS

ARQUIVOS

RUBRICA

Ш





0 / 0,3 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 = 0; i < n-4; 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.

O n, n

3(n-4), n-4



```
9 == 5) {
      a *= 2; 1 *= 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
- 3(n-4), n-4
- n-4, n-4
- 3(n-4), n
- 3(n-4), 0

Pergunta 5

0 / 0,5 pts

Os operadores lógicos and e or são primordiais na confecção de software. Dadas duas ou mais

COMENTÁRIOS

ARQUIVOS

RUBRICA

Ш



<







0 / 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.

COMENTÁRIOS	ARQUIVOS	RUBRICA
III	0	<



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 12
- 6 e 10
- 0 4 e 12
- 0 4e10
- 0 5 e 12



- 0 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.

COMENTÁRIOS	ARQUIVOS	RUBRICA
III	0	<



0,5 / 0,5 pts

Dois vetores ordenados, contendo, cada um deles, n números inteiros, precisam ser unidos em outro vetor maior, que conterá os 2n números, que também serão armazenados de forma ordenada. A complexidade de tempo de melhor caso desse processo será, então (TRANSPETRO'11),

 $O\left(1\right)$, pois se precisa fazer apenas uma cópia simples de cada um dos elementos originais.

 $O(\lg(n))$, pois se usa a busca binária para determinar qual será o próximo elemento copiado para o vetor de destino.

 $O(n \times \lg(n))$, pois se precisa fazer uma busca de cada elemento para depois inseri-lo no vetor de destino.

 $O\left(n^2\right)$, pois, como há dois vetores, precisa-se fazer dois laços de forma aninhada (um dentro do outro), gerando uma multiplicação das

quantidador do alamantas

COMENTÁRIOS	ARQUIVOS	RUBRICA
III	0	<

copiado para o vetor de destino.

 $O(n \times \lg(n))$, pois se precisa fazer uma busca de cada elemento para depois inseri-lo no vetor de destino.

 $O\left(n^2\right)$, pois, como há dois vetores, precisa-se fazer dois laços de forma aninhada (um dentro do outro), gerando uma multiplicação das quantidades de elementos.

O(n), pois se precisa fazer uma cópia de cada um dos elementos originais, o que implica uma varredura completa de cada vetor de origem.

 $O\left(n\right)$, pois se precisa fazer uma cópia de cada um dos elementos originais, o que implica uma varredura completa de cada vetor de origem.

Pergunta 7

0 / 0,5 pts

COMENTÁRIOS

ARQUIVOS

RUBRICA

III







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?

- 0 1
- 0 16
- 32
- 0 6
- 0 5

Ш

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

COMENTÁRIOS ARQUIVOS RUBRICA

última varifica a única anção rectanta





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.

Pergunta 8

0,5 / 0,5 pts

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

 O algoritmo Countingsort ordena um vetor com custo linear.

PORQUE

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

COMENTÁRIOS

ARQUIVOS



Envio

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 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

I - CORRETA: O algoritmo Countingsort efetua em tempo linear $\Theta\left(n\right)$ a ordenação dos elementos de um vetor. Ele considera três vetores: entrada, contagem e saída. O primeiro passo é em $\Theta\left(n\right)$ 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

COMENTÁRIOS

ARQUIVOS

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.

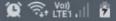
As duas afirmações são independentes.

Pergunta 9

0 / 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).

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







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).

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

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

COMENTÁRIOS

ARQUIVOS

RUBRICA

Ш











0 / 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



Envio

PORQUE

II. O algoritmo de Inserção apresenta complexidade linear - $O\left(n\right)$ - 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 verdadeiras, e a segunda é uma justificativa correta da primeira.

- 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.

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

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

COMENTÁRIOS	ARQUIVOS	RUBRICA
III	0	<

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

- 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 II são corretas.

COMENTÁRIOS

ARQUIVOS



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.

Assina	P 2	al	terr	12thV2	CO	rreta

Comonto	as after	anti-inc	I o II	-50	corretas
Somente	as anni	lativas	ren	540	corretas.

Somente as afirmativas I, II e III são corretas.

Somente as afirmativas I e IV são corretas.

Somente as afirmativas III e IV são corretas



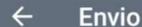
Somente as afirmativas II, III e IV são corretas.

(POSCOMP'13)

COMENTÁRIOS

ARQUIVOS







0 / 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

COMENTÁRIOS	ARQUIVOS	RUBRICA
111	0	<



III

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, 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.

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.

Realmente, as duas afirmações são

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 falsas.

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

Pergunta 13

0 / 2 pts

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

↓ questao13.pdf

COMENTÁRIOS	ARQUIVOS	RUBRICA	
III	0	<	

Encontre a fórmula fechada do somatório $\sum_{0}^{n}(3i-5)^{2}$ e, em seguida, prove a usando indução matemática.



A ser explicada na aula posterior à prova.

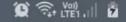


0 / 2 pts

Suponha a classe **Lista** 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 lista correspondem aos n primeiros termos da sequência de Fibonacci. Observe que seu código não deve conter os comandos de repetição for, while e dowhile.

COMENTÁRIOS

ARQUIVOS





^{ldida} Pergunta 14

0 / 2 pts

Suponha a classe **Lista** 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 lista correspondem aos n primeiros termos da sequência de Fibonacci. Observe que seu código não deve conter os comandos de repetição for, while e dowhile.

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: 1,5 de 10