Quiz 02 - Noções de Complexidade

- Entrega 10 fev em 9:21
- Pontos 100
- Perguntas 11
- Disponível 10 fev em 8:50 - 10 fev em 9:21 31 minutos
- Limite de tempo Nenhum

Instruções

Este quiz aborda noções de complexidade. Ele tem 9 questões de múltipla escolha e 2 de verdadeiro ou falso. Após o preenchimento de uma questão, o aluno não tem a opção de retorno à mesma. Este trabalho deve ser efetuado sem consulta.

Este teste foi travado 10 fev em 9:21.

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	Tentativa 1	11 minutos	60 de 100

Pontuação deste teste: 60 de 100
Enviado 10 fev em 9:19
Esta tentativa levou 11 minutos.

Pergunta 1
10 / 10 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-1 ; 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.

Correto!

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

Pergunta 2
10 / 10 pts

Dois vetores ordenados com n números em cada um deles, devem ser unidos e formarão um outro vetor maior com 2n números. Todos eles estarão ordenados no vetor maior.

O custo do tempo de execução do processo de união destes vetores será, então:

Correto!

Custo de 2n operações. Pois se precisa fazer uma cópia de cada um dos elementos originais, o que implica uma varredura completa de cada vetor de origem.

Custo de 1 operação. Pois será necessário apenas uma cópia simples de cada um dos elementos originais.

Custo de n × lg(n) operações. Pois se precisa fazer uma busca de cada elemento para depois inseri-lo no vetor de destino.

Custo de n^2 operações. Pois, como há dois vetores, precisa-se fazer dois laços de forma aninhada (um dentro do outro), gerando uma multiplicação das quantidades de elementos.

Custo de 2n operações. Pois se precisa fazer uma cópia de cada um dos elementos originais, o que implica uma varredura completa de cada vetor de origem.

Pergunta 3
10 / 10 pts

Um dos algoritmos clássicos na computação é fazer a verificação de uma string é ou não palindromo conforme apresentado no código fonte abaixo:

```
int tam = palavra.Length;
for (int i = 0; i < tam / 2; i++){
    if (palavra[i] != palavra[tam - i - 1])
        return false;
}
return true;
```

Pode-se dizer que o custo de comparações deste algoritmo é:

- n comparações.
- 2n comparações.

Correto!

- n/2 comparações.
- n^2 comparações.

O custo em comparações para o algoritmo apresentado é n/2. A palavra é verificada dos extremos até a metade.

Pergunta 4
0 / 10 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-3; 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.

Resposta correta

- 3(n-2), 0
- 3(n-3), 0

Você respondeu

- 3(n-2), n-2
- 3(n-3), n-3
- n-3, n-3

Pergunta 5

0 / 10 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.

Resposta correta

- 3(n-4), 0

Você respondeu

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

Pergunta 6

0 / 10 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 < 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.

Resposta correta

- 5 e 9
- 4 e 12
- 5 e 12

Você respondeu

- 4 e 9
- 6 e 12

O pior caso tem 12 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 o laço.O melhor tem 4 adições e isso acontece quando a primeira condição é falsa e a segunda verdadeira. Nesse caso, o teste do primeiro if realiza duas adições e sua lista de comandos, uma. No segundo if, temos uma adição do teste e mais uma da lista do else.

Pergunta 7

0 / 10 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.

Resposta correta

- 6 e 10
- 4 e 12

Você respondeu

- 6 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 8
10 / 10 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 = n; i >= 1; i >>= 1){
  a *= 2;
}
```

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.

Correto!

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

Pergunta 9
10 / 10 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 <<= 1){
  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.

Correto!

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

Pergunta 10
5 / 5 pts

Algoritmos de busca binária têm complexidade de tempo Θ(lg n), tornando-os eficientes que a busca linear para grandes conjuntos de dados ordenados.
Correto!

- (•) Verdadeiro
- ( ) Falso

Verdadeira. Algoritmos de busca binária têm complexidade de tempo Θ(lg n), tornando-os eficientes que a busca linear para grandes conjuntos de dados ordenados.

Pergunta 11
5 / 5 pts
Um algoritmo com complexidade de espaço Θ(1) significa que sua utilização de memória é constante, independentemente do tamanho da entrada.
Correto!

- (•) Verdadeiro
- ( ) Falso

Verdadeira. A complexidade de espaço Θ(1) indica que o algoritmo utiliza uma quantidade fixa de memória, independentemente do tamanho da entrada.

Pontuação do teste: 60 de 100