

Iniciado em terça-feira, 11 fev. 2025, 14:00

Estado Finalizada

Concluída em terça-feira, 11 fev. 2025, 14:35

Tempo empregado 35 minutos

Avaliar 9,00 de um máximo de 10,00(90%)

Questão 1

Correto

Atingiu 1,00 de 1,00

Qual é a complexidade do código abaixo:

```
#include <stdio.h>
int main() {
int main() {
    int n; // Defina o valor de n
    printf("Digite um valor para n: ");
    scanf("%d", &n);
    int teste = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            teste = teste + i * j;
        }
    }
    printf("Resultado: %d\n", teste);
    return 0;
}
```

- ☐ a. $O(1)$
- ☐ b. $O(\log n)$
- ☐ c. $O(n)$
- ☐ d. $O(n \log n)$
- ☒ e. $O(n^2)$ ✓
- ☐ f. $O(2^n)$
- ☐ g. $O(n!)$

Sua resposta está correta.

Um único laço aninhado como este é $O(n^2)$.



Questão 2

Correto

Atingiu 1,00 de 1,00

Qual a complexidade do seguinte algoritmo?

```
int funcao(int L[N][N]) {  
    return L[100][100];  
}
```

- ☒ a. $O(1)$ ✓
- ☐ b. $O(\log_2(n))$
- ☐ c. $O(n)$
- ☐ d. $O(n \cdot \log_2(n))$
- ☐ e. $O(n^2)$
- ☐ f. $O(n!)$

Sua resposta está correta.

A indexação é executada em tempo constante portanto, a complexidade da função é **constante** em relação ao tamanho da entrada.

Questão 3

Incorreto

Atingiu 0,00 de 1,00

Qual a complexidade do seguinte algoritmo:

```
int funcao(int n) {  
    int aux = 0, i = 1;  
    while (i < n + 1) {  
        i = i * 2;  
        aux = aux + i;  
    }  
    return aux;  
}
```

- ☐ a. $O(1)$
- ☐ b. $O(\log_2(n))$
- ☒ c. $O(n)$ ✗
- ☐ d. $O(n \cdot \log_2(n))$
- ☐ e. $O(n^2)$
- ☐ f. $O(n!)$

Sua resposta está incorreta.

O custo de execução é proporcional à quantidade de instruções executadas. Cada iteração do laço interno (em função de **i**) realiza apenas $i/2$ operações. Como a iteração é repetida $\log n$ vezes no laço interno, sua complexidade é **logaritmica** em relação ao tamanho da entrada.

Questão 4

Correto

Atingiu 1,00 de 1,00

Qual a complexidade do seguinte algoritmo:

```
int funcao(int n) {
    int aux = 0;
    for (int i = 1; i <= n; i++) {
        int j = 1;
        while (j < i) {
            j = j * 2;
            aux = aux + i;
        }
    }
    return aux;
}
```

- ☐ a. $O(1)$
- ☐ b. $O(\log_2(n))$
- ☐ c. $O(n)$
- ☒ d. $O(n \cdot \log_2(n))$ ✓
- ☐ e. $O(n^2)$
- ☐ f. $O(n!)$

Sua resposta está correta.

O custo de execução é proporcional à quantidade de instruções executadas. Cada iteração do laço interno (em função de **j**) realiza apenas duas operações, portanto a complexidade da iteração é **constante**. Como a iteração é repetida **log n** vezes no laço interno, sua complexidade é **logaritmica** em relação ao tamanho da entrada.

O laço interno define o comportamento de cada iteração do laço externo (em função de **i**), e viu-se que esta tem complexidade $O(\log n)$. Como a iteração é repetida **n** vezes no laço externo, sua complexidade é **quasilinear** $O(n \log n)$ em relação ao tamanho da entrada.

Questão 5

Correto

Atingiu 1,00 de 1,00

Das afirmações abaixo sobre busca binária e tabelas de dispersão, qual afirmação é a verdadeira?

- ☐ a. A busca binária tem complexidade de $O(\log n)$ para o caso médio, fazendo com que tenha uma performance melhor que a [tabela de dispersão](#).
- ☐ b. A busca binária tem complexidade de $O(\log n)$ para todos os casos.
- ☐ c. A [tabela de dispersão](#) possui complexidade média $O(n)$.
- ☒ d. A [tabela de dispersão](#) é dependente de uma função chamada "Função de espalhamento". ✓
- ☐ e. Uma vantagem da busca binária sobre a [tabela de dispersão](#) é que a primeira é preparada para tratar dados repetidos, enquanto que a segunda não.

Sua resposta está correta.

Questão 6

Correto

Atingiu 1,00 de 1,00

Os grafos e as árvores são estruturas fundamentais na ciência da computação para modelar relações e hierarquias. Um **grafo** é

composto por um conjunto de ✓ e um conjunto de ✓, que

representam as conexões entre os elementos. Se as conexões possuem direção, o grafo é chamado de ✓

✓, caso contrário, é dito ✓.

Uma **árvore** é um tipo especial de grafo que é ✓, ou seja, não contém

✓, e possui um único ✓, a partir do qual todos os demais nós são acessíveis. Em uma árvore binária,

cada nó pode ter no máximo ✓ filhos. Quando os nós são organizados de forma que a altura da árvore

seja minimizada, a estrutura é chamada de ✓.

Sua resposta está correta.

Questão 7

Correto

Atingiu 1,00 de 1,00

Abaixo é apresentado o código do algoritmo de quick sort. Entretanto, o código, da forma como está, apresenta algum(ns) erro(s). Indique qual(is) linha(s) isso ocorre.

```
1. void swap(int *a, int *b) {
2.     int temp = *b;
3.     *a = *b;
4.     *b = temp;
5. }
6.
7. int partition(int arr[], int low, int high) {
8.     int pivot = arr[high];
9.     int i = (low - 1);
10.    for (int j = low; j <= high - 1; j++) {
11.        if (arr[j] <= pivot) {
12.            i++;
13.            swap(&arr[i], &arr[j]);
14.        }
15.    }
16.    swap(&arr[i + 1], &arr[high]);
17.    return (i + 1);
18. }
19.
20. void quickSort(int arr[], int low, int high) {
21.    if (low > high) {
22.        int pi = partition(arr, low, high);
23.        quickSort(arr, low, pi - 1);
24.        quickSort(arr, pi + 1, high);
25.    }
26. }
```

- ☐ a. Linha 1
- ☒ b. Linha 2 ✓
- ☐ c. Linha 3
- ☐ d. Linha 4
- ☐ e. Linha 5
- ☐ f. Linha 6
- ☐ g. Linha 7
- ☐ h. Linha 8
- ☐ i. Linha 9
- ☐ j. Linha 10
- ☐ k. Linha 11
- ☐ l. Linha 12
- ☐ m. Linha 13
- ☐ n. Linha 14
- ☐ o. Linha 15
- ☒ p. Linha 16 ✓
- ☐ q. Linha 17
- ☐ r. Linha 18
- ☐ s. Linha 19
- ☐ t. Linha 20

- ☒ u. Linha 21 ✓
- ☐ v. Linha 22
- ☐ w. Linha 24
- ☐ x. Linha 25
- ☐ y. Linha 26

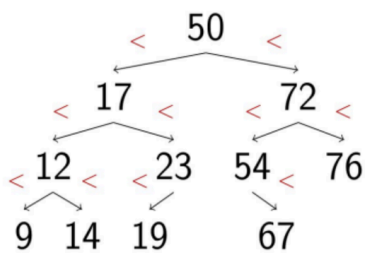
Sua resposta está correta.

Questão 8

Correto

Atingiu 1,00 de 1,00

Seja a seguinte Árvore Binária de Busca:



Informe na caixa de texto abaixo a sequência dos nós visitados ao ser executado o **algoritmo de pos-ordem**. Lembre-se, a correção desta questão é automática, sendo assim, **informe SOMENTE os números separados por vírgula e sem espaço**.

Resposta: 9,14,12,19,23,17,67,54,76,72,50



Questão 9

Correto

Atingiu 2,00 de 2,00

Em uma cidade, há n pessoas identificadas de 1 a n . Há um boato de que uma dessas pessoas é secretamente o juiz da cidade.

Se o juiz da cidade existir, então:

1. O juiz da cidade não confia em ninguém.
2. Todos (exceto o juiz da cidade) confiam no juiz da cidade.
3. Há exatamente uma pessoa que satisfaz as propriedades **1** e **2**.

Você recebe um array `confia` em que `confia[i] = [ai, bi]` representa que a pessoa identificada como a_i confia na pessoa identificada como b_i . Se uma relação de confiança não existir no array `confia`, então essa relação de confiança não existe.

Retorne o identificador do juiz da cidade se o juiz da cidade existir e puder ser identificado ou, caso contrário, retorne -1 .

For example:

Input	Result
2 1 1 2	2
3 2 1 3 2 3	3
3 3 1 3 2 3 3 1	-1

Answer: (penalty regime: 0, 0, 1, 2, ... %)

RESET ANSWER

```
1 | int findJudge(int N, int trust[][2], int trustSize) {
2 |     int confiaEm[N + 1];
3 |     int ehConfiado[N + 1];
4 |
5 |     for (int i = 0; i <= N; i++) {
6 |         confiaEm[i] = 0;
7 |         ehConfiado[i] = 0;
8 |     }
9 |
10 |    for (int i = 0; i < trustSize; i++) {
11 |        int a = trust[i][0];
12 |        int b = trust[i][1];
13 |        confiaEm[a]++;
14 |        ehConfiado[b]++;
15 |    }
16 |
17 |    for (int i = 1; i <= N; i++) {
18 |        if (confiaEm[i] == 0 && ehConfiado[i] == N - 1) {
19 |            return i;
20 |        }
21 |    }
22 |
23 |    return -1;
24 | }
```

	Input	Expected	Got	
✓	2 1 1 2	2	2	✓
✓	3 2 1 3 2 3	3	3	✓
✓	3 3 1 3 2 3 3 1	-1	-1	✓

Passou em todos os teste! ✓

Correto

Notas para este envio: 2,00/2,00.