

Lista2_Algoritmos

Questão 1. Submeta o vetor [33 44 55 77 95 99 22 25 41 66 88 89] ao algoritmo PARTICIONA do QuickSort. Qual o estado final do vetor? Que índice o algoritmo devolve?

Fazendo algumas alterações, temos:

```
#include <string.h>
```

```
#include <math.h>
```

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

```
// A utility function to print an array of size n
```

```
void printArray(int vetor[], int tamanhoOriginal) {
```

```
    for (int i = 0; i < tamanhoOriginal; i++) {
```

```
        if(i == 0) {
```

```
            printf("[%d, ", vetor[i]);
```

```
        }
```

```
        if(i > 0 && i < (tamanhoOriginal - 1)) {
```

```
            printf("%d, ", vetor[i]);
```

```
        }
```

```
        if(i == (tamanhoOriginal - 1)) {
```

```
            printf("%d]\n", vetor[i]);
```

```
        }
```

```
    }
```

```
}
```

```
//Procedimento de troca.
```

```
void trocar(int vetor[], int first, int second) {
```

```
    int guarda = vetor[first];
```

```
    vetor[first] = vetor[second];
```

```
    vetor[second] = guarda;
```

```
}
```

```
//Procedimento que realiza a partição.
```

```
int particionar(int vetor[], int quickInit, int tamanhoOriginal) {
```

```
    //Define auxiliar como último elemento do vetor.
```

```
    int tamanhoCorreto = tamanhoOriginal - 1;
```

```
    int pivo = vetor[tamanhoCorreto];
```

```
    printf ("pivo: %d\n", pivo);
```

```
    //Define i como tamanho da partição.
```

```
    int i = quickInit - 1;
```

```
    int j;
```

```
    //Implementação do laço para realizar a ordenação.
```

```
    for (j = quickInit; j <= (tamanhoCorreto - 1); j++) {
```

```
        //Implementação da condição para realizar a troca.
```

```
        if (vetor[j] < pivo) {
```

```
            i++;
```

```
            trocar (vetor, i, j);
```

```
            printf ("i: %d\n", i);
```

```
            printf ("j: %d\n", j);
```

```
            printf ("vetor[i]: %d\n", vetor[i]);
```

```
            printf ("vetor[j]: %d\n", vetor[j]);
```

```
            printArray(vetor, tamanhoOriginal);
```

```
        }
```

```
    }
```

```
    trocar (vetor, (i + 1), tamanhoCorreto);
```

```
    printf ("i + 1: %d\n", i + 1);
```

```
    printf ("tamanhoCorreto: %d\n", tamanhoCorreto);
```

```
    printf ("vetor[i + 1]: %d\n", vetor[i + 1]);
```

```
    printf ("vetor[tamanhoCorreto]: %d\n", vetor[tamanhoCorreto]);
```

```
    //printArray(vetor, tamanhoOriginal);
```

```

    //printf ("retorno: %d", i + 1);

    return (i + 1);
}

/* Driver program to test particionar */

int main()
{
    int vetor[] = {33, 44, 55, 77, 95, 99, 22, 25, 41, 66, 88, 89};

    int tamanhoOriginal = 12;

    int quickInit = 0;

    printf ("Tam: %d\n", tamanhoOriginal);

    printArray(vetor, tamanhoOriginal);

    particionar(vetor, quickInit, tamanhoOriginal);

    printArray(vetor, tamanhoOriginal);

    return 0;
}

```

Resposta do algoritmo:

1. Vetor devolvido: [33, 44, 55, 77, 22, 25, 41, 66, 88, 89, 95, 99]

2. Índice: 9, onde o índice varia no intervalo: [0, 11].

Questão 2 - 5

Questão 2. Suponha que todos os elementos do vetor $A[p \dots r]$ são iguais entre si. Quantas vezes a linha 4 do algoritmo *Divide* é executada? Qual o valor do índice que o algoritmo devolve? Qual o valor do índice que o algoritmo devolve quando o vetor é crescente? E quando o vetor é decrescente?

Resposta: Para o problema apresentado na questão 1, a quantidade de vezes que a linha 4 (`for (j = quickInit; j <= (tamanhoCorreto - 1); j++)`) é executada: 11 vezes.

Esse tamanho é igual ao retorno do algoritmo *Particionar* (9) + 2.

Para elementos iguais (12 elementos): **retorno do índice: 0 e número de execução: 11.**

Para elementos crescentes (12 elementos): **retorno do índice: 11 e número de execução: 11**

Para elementos decrescentes (12 elementos): **retorno do índice: 0 e número de execução: 11**

Questão 3. Reescreva o Algoritmo *Divide* de modo a usar o valor original (da origem) de $A[p]$ como pivô.

```
#include <string.h>
```

```
#include <math.h>
```

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

```
// A utility function to print an array of size n
```

```
void printArray(int vetor[], int tamanhoOriginal) {
```

```
    for (int i = 0; i < tamanhoOriginal; i++) {
```

```
        if(i == 0) {
```

```
            printf ("%d, ", vetor[i]);
```

```
        }
```

```
        if(i > 0 && i < (tamanhoOriginal - 1)) {
```

```
            printf ("%d, ", vetor[i]);
```

```
        }
```

```
        if(i == (tamanhoOriginal - 1)) {
```

```
            printf ("%d\\n", vetor[i]);
```

```
        }
```

```
    }
```

```
}
```

```
//Procedimento de troca.
```

```
void trocar(int vetor[], int first, int second) {
```

```
    int guarda = vetor[first];
```

```
    vetor[first] = vetor[second];
```

```
    vetor[second] = guarda;
```

```
}
```

```
//Procedimento que realiza a partição.
```

```
int particionar(int vetor[], int quickInit, int tamanhoOriginal) {
```

```
    //Define auxiliar como último elemento do vetor.
```

```
    int contadorLinha4 = 0;
```

```
int tamanhoCorreto = tamanhoOriginal - 1;

int pivo = vetor[0];

printf ("pivo: %d\n", pivo);

//Define i como tamanho da partição.

int i = quickInit - 1;

int j;

//Implementação do laço para realizar a ordenação.

for (j = quickInit; j <= (tamanhoCorreto - 1); j++) {

    contadorLinha4++;

    printf ("jant: %d\n", j);

    //Implementação da condição para realizar a troca.

    if (vetor[j] < pivo) {

        i++;

        trocar (vetor, i, j);

        printf ("i: %d\n", i);

        printf ("j: %d\n", j);

        printf ("vetor[i]: %d\n", vetor[i]);

        printf ("vetor[j]: %d\n", vetor[j]);

        printArray(vetor, tamanhoOriginal);

    }

}

trocar (vetor, (i + 1), tamanhoCorreto);

printf ("i + 1: %d\n", i + 1);

printf ("tamanhoCorreto: %d\n", tamanhoCorreto);

printf ("vetor[i + 1]: %d\n", vetor[i + 1]);

printf ("vetor[tamanhoCorreto]: %d\n", vetor[tamanhoCorreto]);

//printArray(vetor, tamanhoOriginal);

printf ("retorno: %d\n", i + 1);

printf ("contadorLinha4: %d\n", contadorLinha4);

return (i + 1);
```

```

}

/* Driver program to test Particionar*/

int main()
{
    int vetor[] = {33, 44, 55, 77, 95, 99, 22, 25, 41, 66, 88, 89};

    //int vetor[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

    //int vetor[] = {12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

    int tamanhoOriginal = 12;

    int quickInit = 0;

    printf ("Tam: %d\n", tamanhoOriginal);

    printArray(vetor, tamanhoOriginal);

    particionar(vetor, quickInit, tamanhoOriginal);

    printArray(vetor, tamanhoOriginal);

    return 0;
}

```

Questão 4. **Aplicação direta do algoritmo.**

Questão 5.

```

#include <string.h>

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

```

// A utility function to print an array of size n

```

void printArray(int vetor[], int tamanhoOriginal) {
    for (int i = 0; i < tamanhoOriginal; i++) {
        if(i == 0) {
            printf ("%d, ", vetor[i]);
        }

        if(i > 0 && i < (tamanhoOriginal - 1)) {
            printf ("%d, ", vetor[i]);
        }
    }
}

```

```

        if(i == (tamanhoOriginal - 1)) {
            printf ("%d", vetor[i]);
        }
    }
}

```

/ Function to sort an array using insertion sort */*

```

void insertionSort(int vetor[], int i, int tamanhoOriginal) {

```

```

    int key, j;

```

```

    key = vetor[i];

```

```

    j = i - 1;

```

```

    /* Move elements of vetor[0..i-1], that are
       greater than key, to one position ahead
       of their current position */

```

```

    while (j >= 0 && vetor[j] > key) {

```

```

        //Troca das posições (j + 1 = i).

```

```

        vetor[j + 1] = vetor[j];

```

```

        //Decremento de j.

```

```

        j = j - 1;

```

```

    }

```

```

    vetor[j + 1] = key;

```

```

    if ((i + 1) <= tamanhoOriginal) {

```

```

        insertionSort (vetor, (i + 1), tamanhoOriginal);

```

```

    }

```

```

}

```

/ Driver program to test insertion sort */*

```

int main() {

```

```

    //int vetor[] = { 12, 13, 14, 15, 16};

```

```

    int vetor[] = { 22, 20, 3, 2, 1};

```

```
// int tamanho = sizeof(vetor) / sizeof(vetor[0]);
```

```
int tamanhoOriginal = 5;
```

```
/*printf("Enter the size of array: ");
```

```
scanf("%d", &tamanhoOriginal);
```

```
int vetor[tamanhoOriginal];
```

```
for (int i = 0; i < tamanhoOriginal; i++) {
```

```
    vetor[i] = rand() % 100;
```

```
}*/
```

```
printArray(vetor, tamanhoOriginal);
```

```
insertionSort(vetor, 1, tamanhoOriginal);
```

```
printArray(vetor, tamanhoOriginal);
```

```
return 0;
```

Questão 6. Vetor com todos os elementos em ordem decrescente. O while é percorrido várias vezes em ordem aritmética com o aumento do índice i $[(1 + 2 + 3 + 4 + + (i - 1))]$

```
}
```