INSTITUTO FEDERAL
Ceará

Nome: Ana Vitória Cabral Duarte
Matrícula: 20212013020205

Implementar a compilação e execução do algoritmo de classificação usando o QuickSort e o Merge Sort.
Responda:
1) No caso do Quick Sort :
Qual foi a estratégia usada para escolher o Pivot?
No exemplo dado quantos testes foram executados?

https://www.geeksforgeeks.org/quick-sort/

Implementação:

```
#include <stdio.h>

// Função para fundir dois subarrays
void merge(int arr[], int low, int mid, int high) {
    int n1 = mid - low + 1;
    int n2 = high - mid;

    int left[n1], right[n2];

    for (int i = 0; i < n1; i++) {
        left[i] = arr[low + i];
    }
    for (int i = 0; i < n2; i++) {
        right[i] = arr[mid + 1 + i];
    }

    int i = 0, j = 0, k = low;

    while (i < n1 && j < n2) {
        if (left[i] <= right[j]) {
            arr[k] = left[i];
            i++;
        } else {
            arr[k] = right[j];
            j++;
        }
```

```
        k++;
    }

    while (i < n1) {
        arr[k] = left[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = right[j];
        j++;
        k++;
    }
}

// Implementação da função Merge Sort
void mergeSort(int arr[], int low, int high) {
    if (low < high) {
        int mid = low + (high - low) / 2;
        mergeSort(arr, low, mid);
        mergeSort(arr, mid + 1, high);
        merge(arr, low, mid, high);
    }
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high; j++) {
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return (i + 1);
}

// Implementação da função Quick Sort
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
```

```
}

int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int n = sizeof(arr) / sizeof(arr[0]);

    // Escolha aqui se deseja usar o Quick Sort ou Merge Sort
    // Exemplo: quickSort(arr, 0, n - 1);
    mergeSort(arr, 0, n - 1);

    printf("Array Ordenado\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

- **Qual foi a estratégia usada para escolher o Pivot?**

O elemento do pivô é escolhido como o último elemento do array (arr[high]).
A estratégia usada para escolher o pivô é selecionar o último elemento do array como o pivô.

- **No exemplo dado quantos testes foram executados?**

No exemplo dado com o array {10, 7, 8, 9, 1, 5}, o Quick Sort realizará um total de 5 testes (partições) durante a execução:

A primeira chamada ordena a partição {1, 7, 8, 9, 5} com o pivô 5.
A segunda chamada ordena a partição {1, 7, 8, 9} com o pivô 7.
A terceira chamada ordena a partição {1, 5, 8, 9} com o pivô 5.
A quarta chamada ordena a partição {1, 5} com o pivô 1.
A quinta chamada ordena a partição {8, 9} com o pivô 9

https://www.geeksforgeeks.org/merge-sort/

Implementação:

```
#include <stdio.h>

// Função para realizar a partição
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
```

```
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return (i + 1);
}

// Implementação da função Quick Sort
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

// Função para imprimir um array
void imprimirArray(int arr[], int tamanho) {
    for (int i = 0; i < tamanho; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Código principal
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int tamanho = sizeof(arr) / sizeof(arr[0]);

    printf("Array original:\n");
    imprimirArray(arr, tamanho);

    quickSort(arr, 0, tamanho - 1);

    printf("\nArray ordenado:\n");
    imprimirArray(arr, tamanho);

    return 0;
}
```

- **Qual foi a estratégia usada para escolher o Pivot?**

O elemento do pivô é escolhido como o último elemento do subarray a ser ordenado (arr[high]).

- **No exemplo dado quantos testes foram executados?**

No exemplo dado, que usa o array {12, 11, 13, 5, 6, 7}, o pivô será escolhido como 7, que é o último elemento do subarray a ser ordenado.

o Quick Sort realizará um total de 5 testes (partições) durante a execução:

A primeira chamada ordena a partição {5, 6, 11} com o pivô 7.
A segunda chamada ordena a partição {5, 6} com o pivô 6.
A terceira chamada ordena a partição {5} com o pivô 5.
A quarta chamada ordena a partição {11} com o pivô 11.
A quinta chamada ordena a partição {13} com o pivô 13.