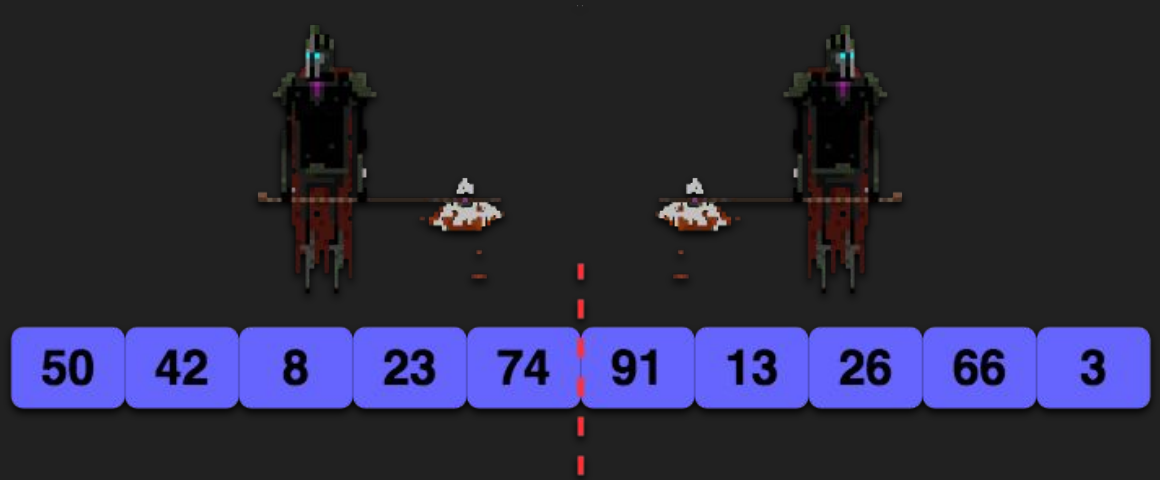
Algoritmos de Ordenação

QuickSort e MergeSort

Equipe:
Armando Luz Borges
Annyel Cordeiro da Silva
João dos Santos Neto
Mayra Caetano de Oliveira
Nara Raquel Dias Andrade
Vinicius da Silva Nunes

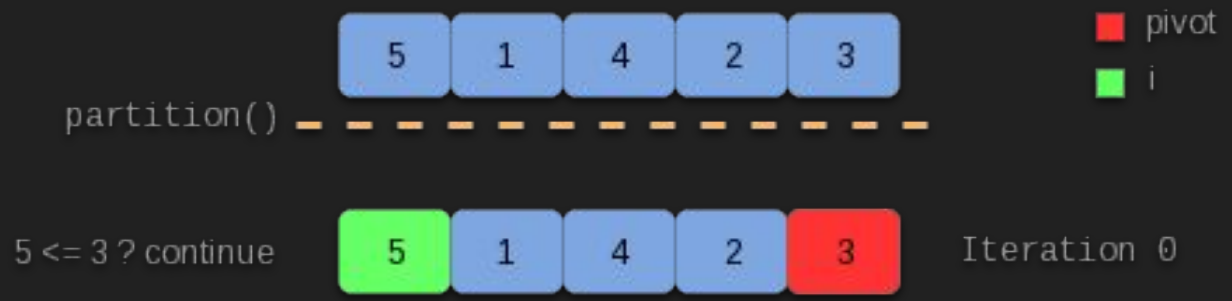
QuickSort - Ordenação por Comparação

Quick Sort é um algoritmo de ordenação eficiente, que tem como base o conceito de divisão e conquista. Apesar de ser complexo como o MergeSort e o HeapSort, na prática, o QuickSort é mais veloz, pois suas constantes são menores.



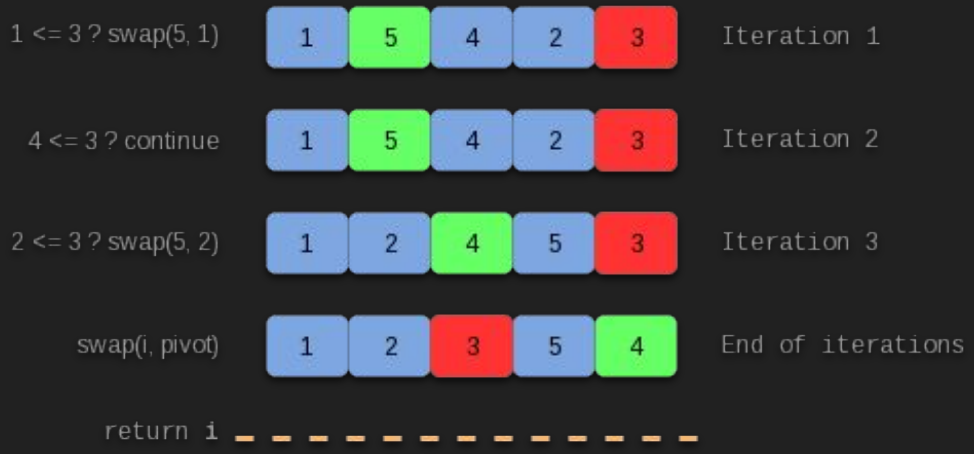
QuickSort - Como funciona?

O Funcionamento do QuickSort é baseado em uma rotina fundamental chamada de **particionamento**, onde um número presente no array será escolhido, sendo nomeado de ***pivô,*** o elemento escolhido será usado como uma espécie de ponto médio dos elementos.

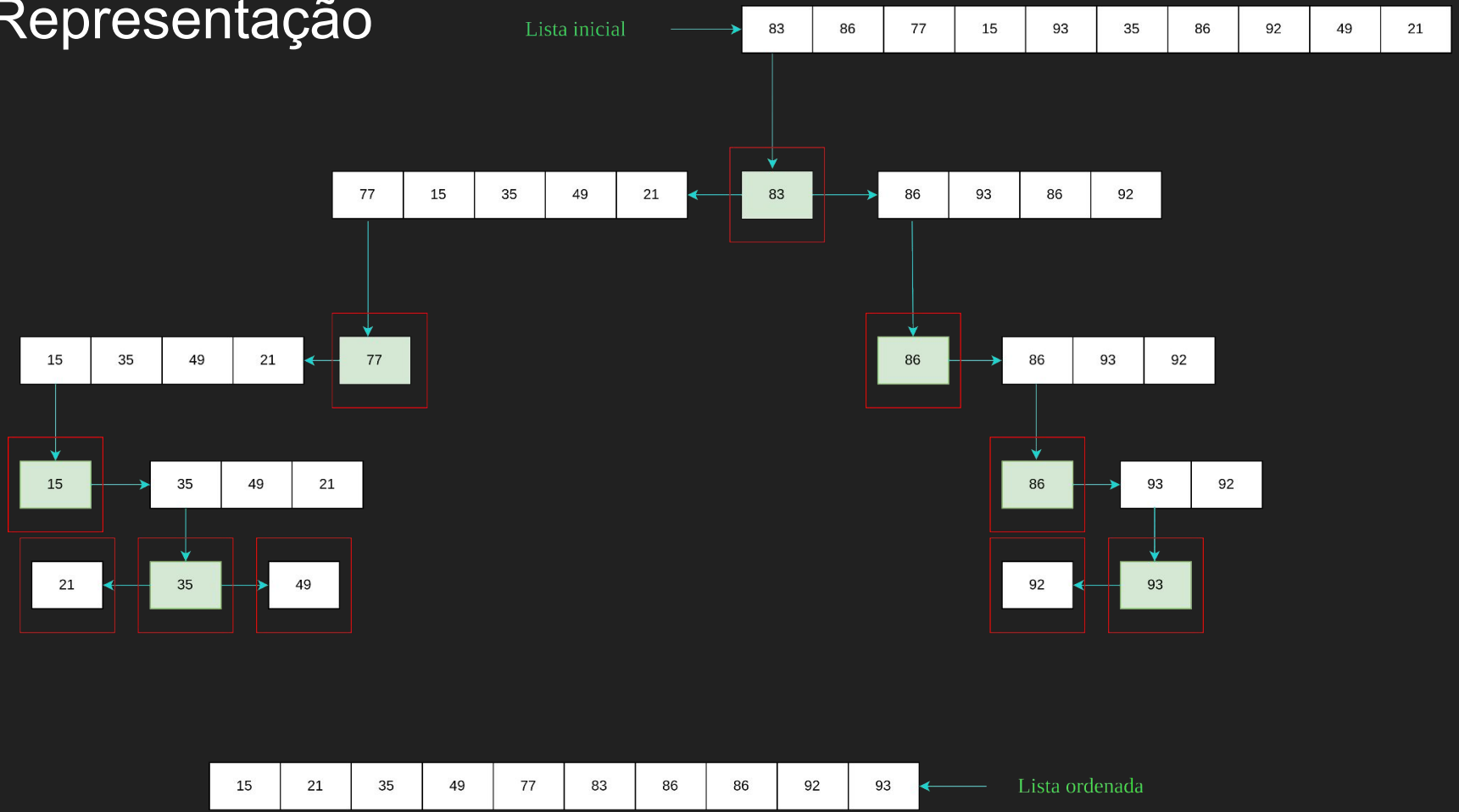


QuickSort - Como funciona?

Os elementos maiores que o pivô são postos à direita, e os menores são postos à esquerda. A partir daí, são formadas sub-listas, nas quais o mesmo processo é repetido, até que se chegue nos átomos da lista. Neste ponto os valores já estarão ordenados.



Representação



Quick Sort - Pior Caso

- No pior caso o Quick Sort é O(n2). Esse caso se manifesta quando o pivot **sempre** divide o array em duas porções de tamanho 0 e n-1, respectivamente.

Ex: values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

- Na primeira execução do particionamento ele já está em sua posição, pois o array está ordenado. Assim, chamamos recursivamente para a esquerda (vazio) e para a direita (2, 3, 4…10).

Quick Sort - Como Evitar o Pior Caso

- Escolher aleatoriamente o pivot é uma boa estratégia para diminuir significativamente a probabilidade de ocorrência do pior caso.

- Escolher o pivot como sendo a mediana entre o primeiro elemento, o elemento central e o último elemento é uma boa estratégia para diminuir significativamente a probabilidade de ocorrência do pior caso.

Ex: array [5, 2, 3, 7, 9] mediana: 5

- A mediana entre esses elementos é o 5. Portanto, essa seria a nossa escolha de pivot.

Quick Sort - Melhor Caso

- No melhor caso o Quick Sort é O(n∗log n). Esse caso se manifesta quando o pivot sempre divide o array em duas porções iguais.

- Se o pivot sempre ficar no meio do array, teremos uma árvore binária na recursão em que a esquerda tem metade do tamanho do array e a direita também tem a metade do tamanho.

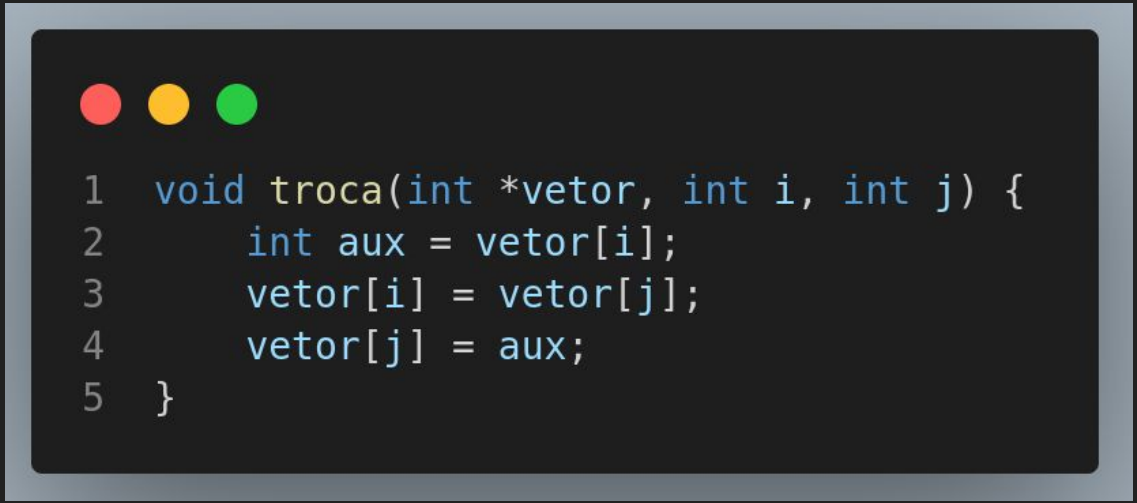
Quick Sort - Implementação

```c
void quickSort(int *vetor, int inicio, int fim) {
    if (inicio < fim) {
        int pivo = particiona(vetor, inicio, fim);
        quickSort(vetor, inicio, pivo - 1);
        quickSort(vetor, pivo + 1, fim);
    }
}
```

Quick Sort - Implementação

```c
int particiona(int *vetor, int inicio, int fim) {
    int pivo = vetor[fim];
    int i = inicio - 1;
    int j;
    for (j = inicio; j < fim; j++) {
        if (vetor[j] <= pivo) {
            i++;
            troca(vetor, i, j);
        }
    }
    troca(vetor, i + 1, fim);
    return i + 1;
}
```

Quick Sort - Implementação



```c
void troca(int *vetor, int i, int j) {
    int aux = vetor[i];
    vetor[i] = vetor[j];
    vetor[j] = aux;
}
```

Merge Sort - O que é?

- Merge Sort é um algoritmo de classificação que divide um array em sub-arrays menores, classificando cada sub-array e, em seguida, mesclando os sub-arrays classificados novamente para formar o array classificado final.

- A classificação por mesclagem é uma escolha popular para classificar grandes conjuntos de dados porque é relativamente eficiente e fácil de implementar. Geralmente é usado em conjunto com outros algoritmos, como o quicksort, para melhorar o desempenho geral de uma rotina de classificação.

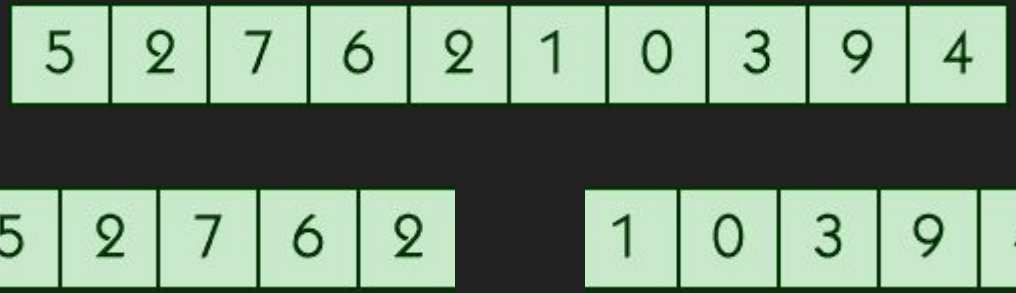
Merge Sort - Como funciona?

- A ideia do Merge Sort é dividir o vetor em dois sub-arrays, cada um com metade dos elementos do array original. Esse procedimento é então aplicado aos dois sub-arrays recursivamente.

- Quando os sub-arrays têm apenas um elemento (caso base), a recursão para. Então, os sub-arrays ordenados são fundidos (ou intercalados) num único array ordenado.

Merge Sort - Funcionamento

Como exemplo, ordenamos o array [5, 2, 7, 6, 2, 1, 0, 3, 9, 4]. Inicialmente, dividimos o array em dois sub-arrays, cada um com metade dos elementos do array original.



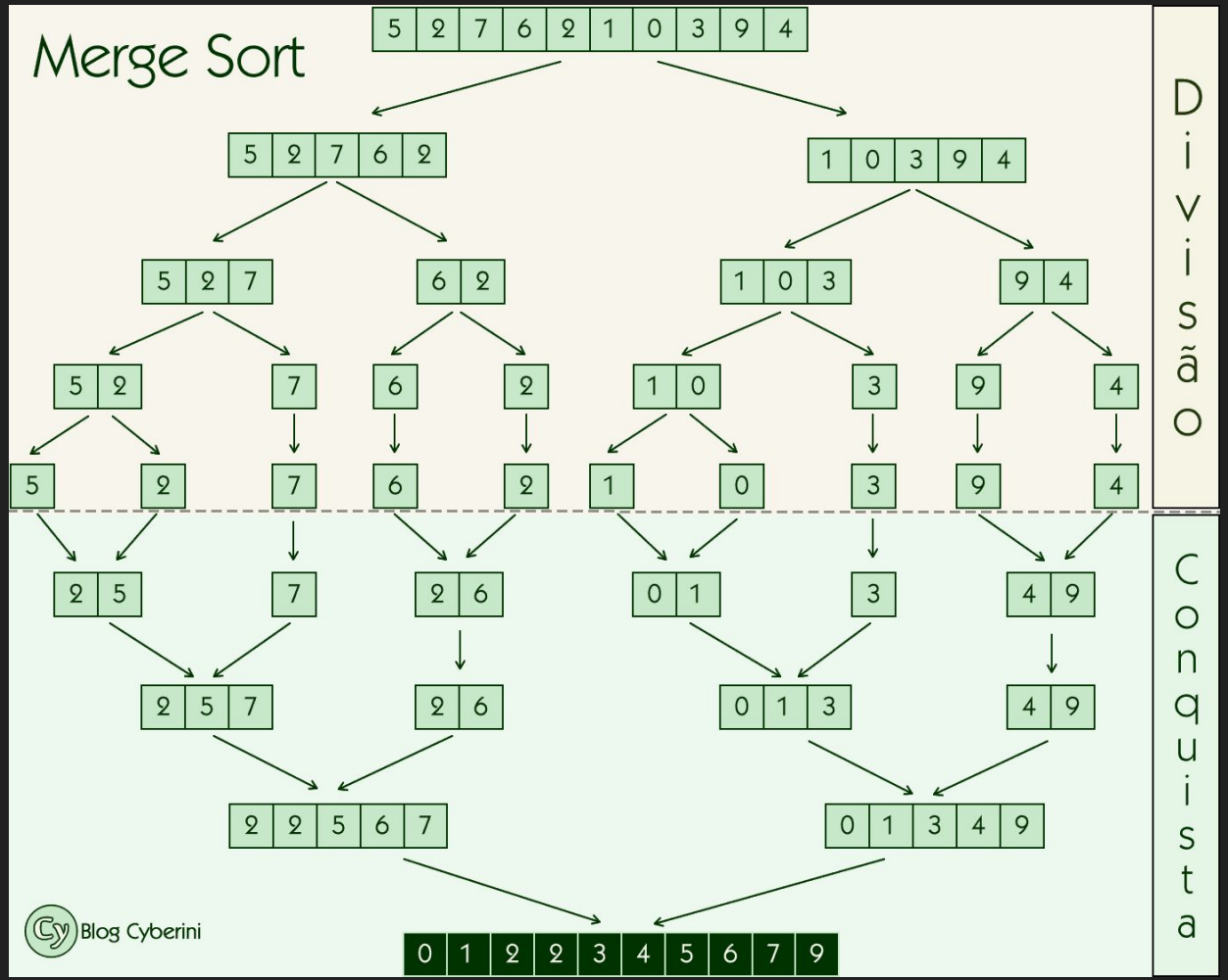
Merge Sort - Funcionamento

Após isso, reaplicamos o método aos dois sub-arrays

De novo,

Merge Sort - Funcionamento

Finalmente, fazemos a fusão dos subvetores


Merge Sort
5 2 7 6 2 1 0 3 9 4
5 2 7 6 2
1 0 3 9 4
5 2 7
6 2
1 0 3
9 4
5 2
7
6
2
1 0
3
9
4
5
2
7
6
2
1
0
3
9
4
2 5
7
2 6
0 1
3
4 9
2 5 7
2 6
0 1 3
4 9
2 2 5 6 7
0 1 3 4 9
0 1 2 2 3 4 5 6 7 9
Divisão
Conquista
Blog Cyberini

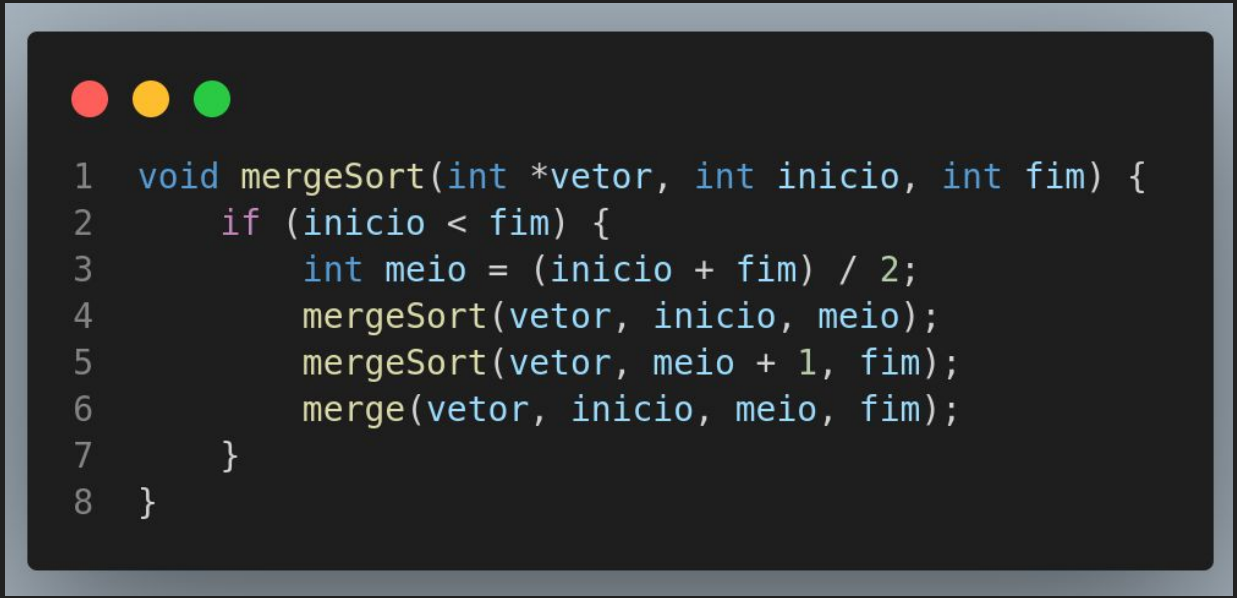
Merge Sort - Custo

- O Merge Sort tem um característica bastante diferente comparado ao Quick Sort, em relação ao custo, pois ele apresenta uma análise assintótica de N Log N no seu pior, médio e melhor caso.

- **Análise do Tempo de Execução:** N Log N

- **Análise do uso de memória:** O(N), em outras palavras, ele NÃO é in-place e é estável.

Merge Sort - Implementação



```c
void mergeSort(int *vetor, int inicio, int fim) {
    if (inicio < fim) {
        int meio = (inicio + fim) / 2;
        mergeSort(vetor, inicio, meio);
        mergeSort(vetor, meio + 1, fim);
        merge(vetor, inicio, meio, fim);
    }
}
```

Merge Sort - Implementação

```c
void merge(int *vetor, int inicio, int meio, int fim) {
    int i, j, k;
    int n1 = meio - inicio + 1;
    int n2 = fim - meio;

    int *L = (int *) malloc(n1 * sizeof(int));
    int *R = (int *) malloc(n2 * sizeof(int));

    for (i = 0; i < n1; i++) {
        L[i] = vetor[inicio + i];
    }
    for (j = 0; j < n2; j++) {
        R[j] = vetor[meio + 1 + j];
    }
```

```c
    i = 0;
    j = 0;
    k = inicio;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            vetor[k] = L[i];
            i++;
        } else {
            vetor[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        vetor[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        vetor[k] = R[j];
        j++;
        k++;
    }
}
```

Referências

https://blog.pantuza.com/artigos/o-algoritmo-de-ordenacao-quicksort

https://www.geeksforgeeks.org/quick-sort/?ref=lbp

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

https://2.bp.blogspot.com/-r6_Xn_OLzMM/WzbMOnybAPI/AAAAAAAAIIU/XWGEsWS-m_c9rby-Q19j3G_2DAoyuInEACLcBGAs/s1600/diagrama-merge-sort.png

https://www.blogcyberini.com/2018/07/merge-sort.html

https://joaoarthurbm.github.io/eda/posts/quick-sort/#:~:text=Ent%C3%A3o%2C%20o%20custo%20do%20melhor,array%20em%20duas%20por%C3%A7%C3%B5es%20iguais

https://www.google.com/url?q=https://joaoarthurbm.github.io/eda/posts/merge-sort/&sa=D&source=editors&ust=1678193467510112&usg=AOvVaw1amlg78sI0cZ4BhtRv_1-k

Obrigada pela atenção!

Link para a questão

https://docs.google.com/document/d/1FJi34qCGXPzJ2tJFRhWyAxsVj5e_W9giby7qPgnMFKs/