Estácio

CURSO: BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

DISCIPLINA: ALGORITMOS E COMPLEXIDADE

DOCENTE: BRENO CAETANO DA SILVA

SEMESTRE LETIVO: 2023.2 TURNO: Manhã

TRABALHO DE ALGORITMOS E COMPLEXIDADE

Integrantes

JOÃO LUCAS BRITO MOURA – 202003270695

VITOR DOS SANTOS BARBOSA PORTELA – 202002121025

SEBASTIÃO FONSECA DA COSTA JUNIOR – 202001602844

Algoritmo De Ordenação: Merge Sort - Exemplo 1

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

void merge_sort(int a[], int length);
void merge_sort_recursion(int a[], int l, int r);
void merge_sorted_arrays(int a[], int l, int m, int r);

int main()  {

  int array[] = { 9, 4, 8, 1, 7, 0, 3, 2, 5, 6};
  int length = 10;

        // Classificando o array usando "Merge Sort"

  merge_sort(array, length);

        //  Imprimindo os elementos da matriz para verificar se eles foram
classificados

  for (int i = 0; i < length; i++)
    printf("%d ", array[i]);
  printf("\n");

  return 0;

        }
```

```
void merge_sort(int a[], int length) {

  merge_sort_recursion(a, 0, length - 1);

        }

void merge_sort_recursion(int a[], int l, int r) {

        // Paramos a recursão quando l >= r

  if (l < r) {

        // Encontar o ponto médio de l e r

   int m = l + (r - l) / 2;

        // Aplicando a função recursivamente nas partes da esquerda e da direita divididas

   merge_sort_recursion(a, l, m);
   merge_sort_recursion(a, m + 1, r);

   // Neste ponto, ambas as partes do array foram classificadas

        // Agora mesclar as partes classificadas do array
```

```
   merge_sorted_arrays(a, l, m, r); }


        }


   // Juntar as duas partes classificadas do array a entre os índices l ... m
e m + 1 ... r


void merge_sorted_arrays(int a[], int l, int m, int r) {


   // Calcular o comprimento das partes esquerda e direita do array


  int left_length = m - l + 1;

  int right_length = r - m;


   // Criar arrays temporários para armazenar essas porções


  int temp_left[left_length];

  int temp_right[right_length];


   // Usando como variáveis de índice/contador para os 3 arrays a,
"temp_left", "temp_right"


  int i, j, k;


   // Copiar a parte esquerda para o array temp_left


  for (int i = 0; i < left_length; i++)
```

```
   temp_left[i] = a[l + i];

   // Copiar a parte direita no array temp_right

  for (int i = 0; i < right_length; i++)
   temp_right[i] = a[m + 1 + i];

   /* Usei "i" para percorrer os índices de "temp_left", o "j" para
percorrer os índices "temp_right"
      e o "k" para percorrer a parte do array */

  for (i = 0, j = 0, k = l; k <= r; k++) {

         /* Contanto que ainda não tenhamos chegado ao final do array
"temp_left"
      com a variável i, então se for o próximo elemento no array left_temp
      é menor */

   /* Ou no caso de se alcançamos o final do array "temp_right", então
armazena
            o próximo elemento de temp_left no próximo elemento na a
matriz "a" */

   if ((i < left_length) &&
       (j >= right_length || temp_left[i] <= temp_right[j]))
   {
     a[k] = temp_left[i];
     i++;
```

```
    } /* Caso contrário, se o próximo elemento em "temp_right" for o
próximo elemento em
       temp_left ou chegando ao final de "temp_left", armazene o próximo
elemento do
            array temp_right para o próximo elemento do array "a" */
    else {
      a[k] = temp_right[j];
      j++;
          }
   }
}
/* Exemplo de visualização do algoritmo de classificação por mesclagem:
          [38, 27, 43, 3, 9, 82, 10]
                  / \
        [38, 27, 43, 3]  [9, 82, 10]
        /       |       |     \
   [38, 27]   [43, 3]  [9, 82]   [10]
    / |     /   |   /   \     |
  [38] [27]  [43]  [3]  [9]   [82]  [10]
    \ /     \  /    \    /    |
   [27, 38]   [3, 43]   [9, 82]    [10]
      \       /       \      /
     [3, 27, 38, 43]       [9, 10, 82]
          \               /
         [3, 9, 10, 27, 38, 43, 82]
*/
```