Ordenação por intercalação (merge sort)

Alunos: Elivander,Marcelo,Lucas

O merge sort ou ordenação por mistura

- O algoritmo MergeSort, é um algoritmo de ordenação recursivo, que utiliza o método “dividir para conquistar

- O algoritmo foi proposto por John Von Neumann em 1945

Descrição da técnica

Esta técnica realiza-se em três fases:

Divisão: o problema maior é dividido em problemas menores

Conquista: o resultado do problema é calculado quando o problema é pequeno o suficiente.

Combinação:

Vantagens

1. **O MergeSort funciona melhor em listas ligadas em comparação que ele faz em matrizes.**

2. **Algoritmos que utilizam o método de partição são caracterizados por serem os mais rápidos dentre os outros algoritmos**

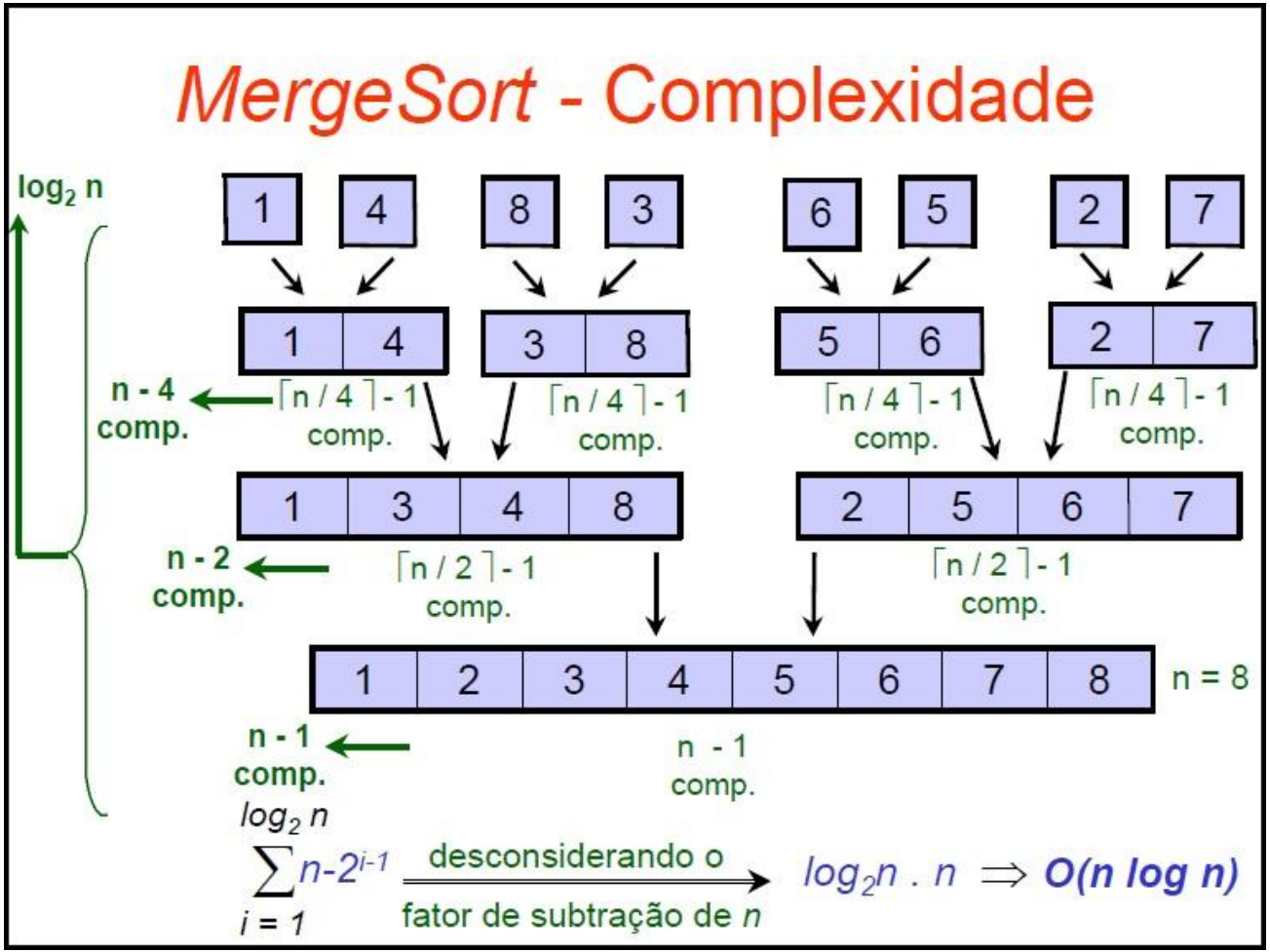
Algoritmo	Tempo		
	Melhor	Médio	Pior
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quick sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$

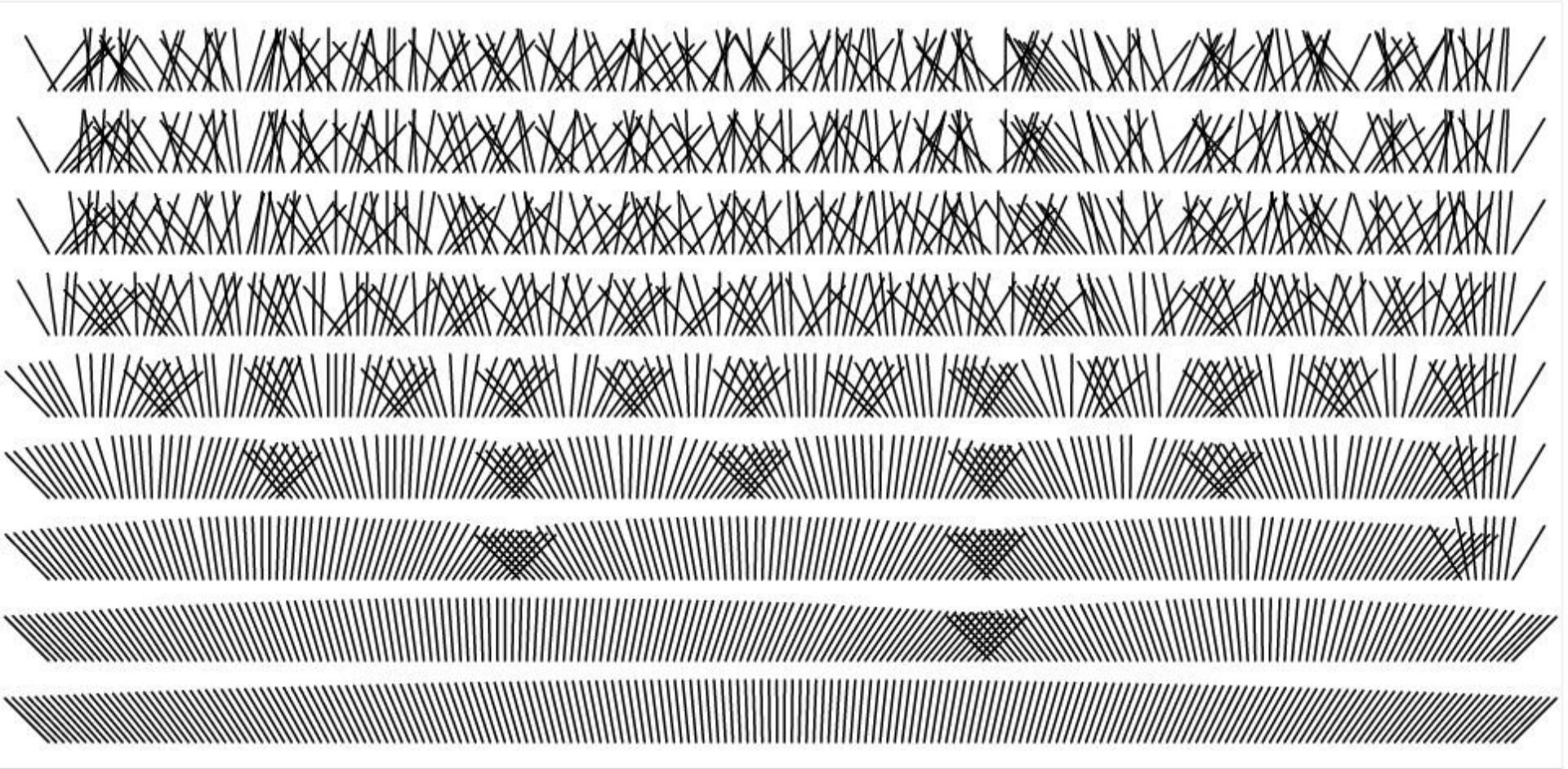
Desvantagens

- **Utiliza funções recursivas;**
- **Gasto extra de memória. O algoritmo cria uma cópia do vetor para cada nível da chamada recursiva, totalizando um uso adicional de memória igual a**

6 5 3 1 8 7 2 4

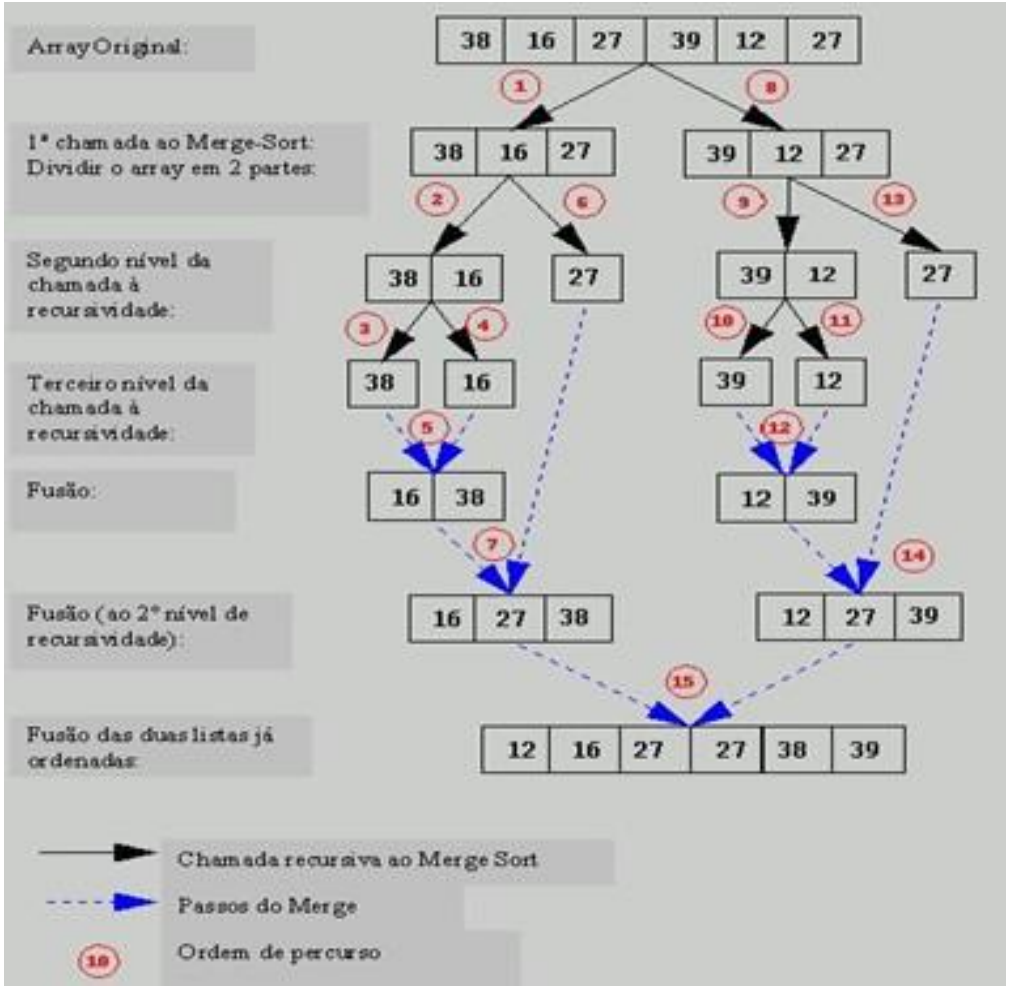
MergeSort - Complexidade





```c
int main (){
    int n, i, *vet;
    int a=1;
      a=a/2;
    printf("%i\n",a);

    printf("defina o tamanho do vetor\n");
    scanf("%d", &n);

    vet = (int *)malloc(n * sizeof (int));

    for(i=0; i<n;i++)
        scanf("%d", &vet[i]);

    mergeSort(vet, 0, n);


    for(i=0; i<n;i++)
        printf("%d ", vet[i]);

}
```

```
//função recursiva que divide o vetor em vetores menores
void mergeSort(int vetor[], int comeco, int fim){
    if (comeco < fim) {
        int meio = (fim+comeco)/2;

        mergeSort(vetor, comeco, meio);
        mergeSort(vetor, meio+1, fim);
        merge(vetor, comeco, meio, fim); //depois de dividir todo o vetor, na volta ele chama está função para ordenar o vetor
    }
}
```


Array Original:
38 16 27 39 12 27
1ª chamada ao Merge-Sort: Dividir o array em 2 partes:
38 16 27
39 12 27
Segundo nível da chamada à recursividade:
38 16
27
39 12
27
Terceiro nível da chamada à recursividade:
38
16
39
12
Fusão:
16 38
12 39
Fusão (ao 2º nível de recursividade):
16 27 38
12 27 39
Fusão das duas listas já ordenadas:
12 16 27 27 38 39
Chamada recursiva ao Merge Sort
Passos do Merge
Ordem de percurso

```c
void merge(int vetor[], int comeco, int meio, int fim) {

    int com1 = comeco;

    int com2 = meio+1;

    int comAux = 0;

    int tam = fim-comeco+1;

    int *vetAux;//criando vetor auxiliar
    vetAux = (int*)malloc(tam * sizeof(int));


    while(com1 <= meio && com2 <= fim){  // ordena e junta os dois vetores em um auxiliar
        if(vetor[com1] < vetor[com2]) {
            vetAux[comAux] = vetor[com1];
            com1++;
        } else {
            vetAux[comAux] = vetor[com2];
            com2++;
        }
        comAux++;
    }
    while(com1 <= meio){  //Caso ainda haja elementos na primeira metade
        vetAux[comAux] = vetor[com1];
        comAux++;
        com1++;
    }
```

```
while(com2 <= fim) {   //Caso ainda haja elementos na segunda metade
    vetAux[comAux] = vetor[com2];
    comAux++;
    com2++;
}

for(comAux = comeco; comAux <= fim; comAux++){    //Move os elementos de volta para o vetor original
    vetor[comAux] = vetAux[comAux-comeco];
}
free(vetAux);
```

1. Felipe, Henrique (1 de julho 2018). «Merge Sort». Blog Cyberini. Consultado em 6 de julho de 2018
2. https://www.interviewbit.com/tutorial/merge-sort-algorithm/
3. https://www.geeksforgeeks.org/merge-sort/
4. https://www.cos.ufrj.br/~rfarias/cos121/aula_07.html