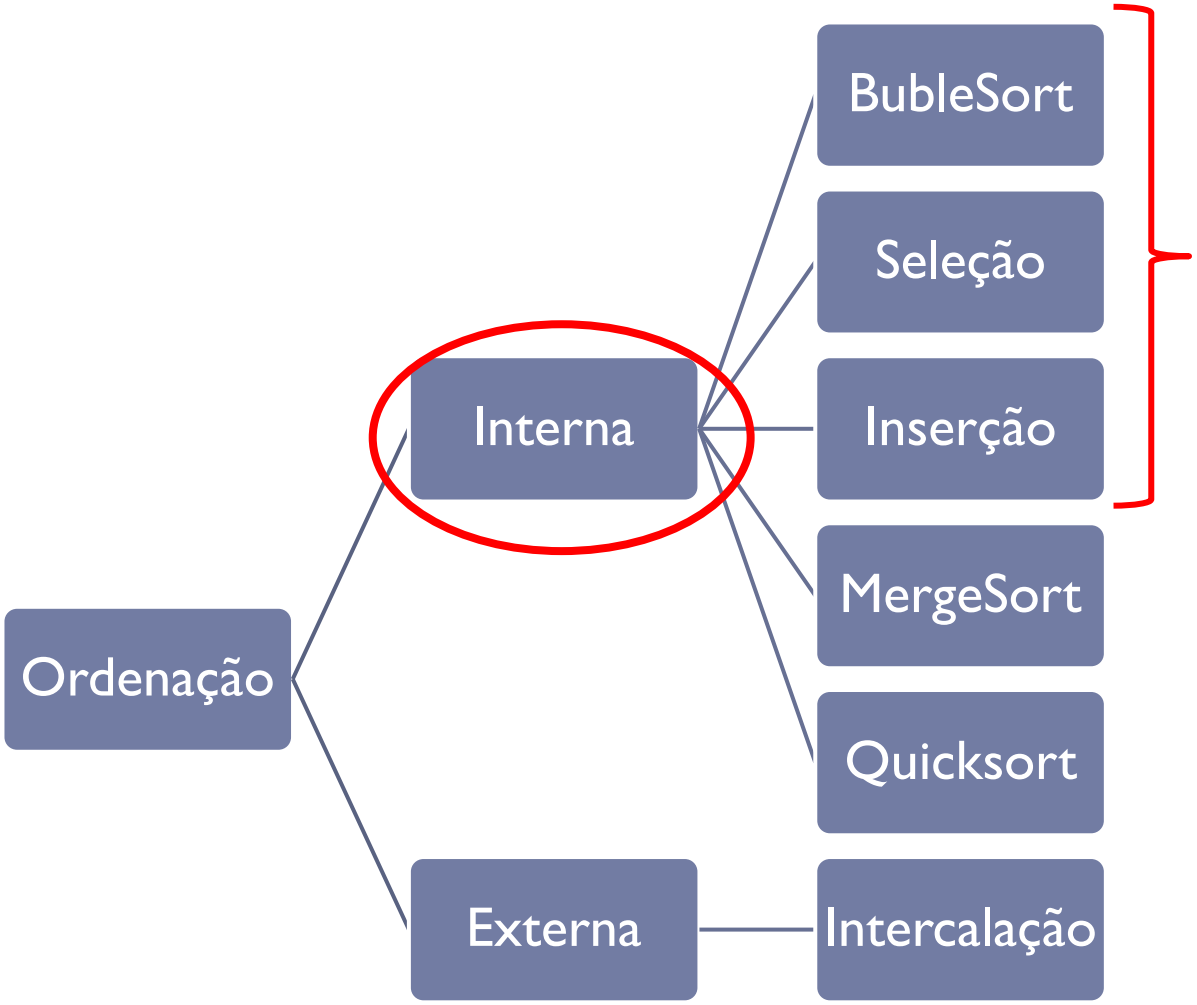
Algoritmo e Estrutura de Dados II
COM-112

Vanessa Souza

Universidade Federal de Itajubá – Instituto de Matemática e Computação

Ordenação

Classificação dos Métodos de Ordenação



Comparação entre os métodos

- Existe uma outra gama de algoritmos de ordenação mais eficientes ($O(n\log_2 n)$).
  - mergeSort
  - quickSort

- Esses algoritmos baseiam-se na estratégia de DIVIDIR PARA CONQUISTAR

- Implementações mais difíceis – estritamente recursivos

RECURSÃO

Revisão

FONTE : Ziviani

Algoritmo Recursivo

- Um método que chama a sim mesmo, direta ou indiretamente, é dito recursivo.

- O uso da recursividade geralmente permite uma descrição mais clara e concisa dos algoritmos, especialmente quando o problema a ser resolvido é recursivo por natureza ou utiliza estruturas recursivas, tais como as árvores.

Algoritmo Recursivo

- Um compilador implementa um método recursivo por meio de uma **pilha**, na qual são armazenados os dados usados em cada chamada de um método que ainda não terminou de processar.

- Todos os dados não globais vão para a pilha, pois o estado corrente da computação deve ser registrado para que possa ser recuperado de uma nova ativação de um método recursivo, quando a ativação anterior deverá prosseguir.

Algoritmo Recursivo

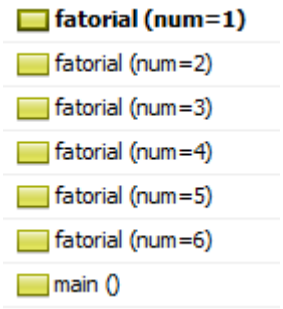
- Quando alcança sua condição de parada, o método retorna para quem chamou, utilizando o endereço de retorno que está no topo da pilha.

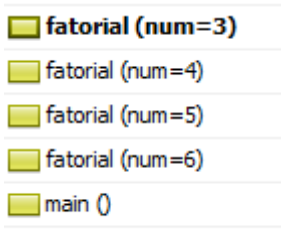
Algoritmo Recursivo

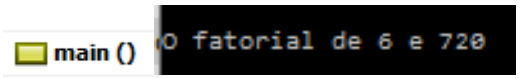
- Exemplo: Cálculo do Fatorial

```
int fatorial(int num)
{
    int fat;
    if (num <= 1)
        return 1;
    else
        fat = num * fatorial (num - 1);
    return fat;
}
```

```
Digite um numero para saber seu fatorial : 6
```







Algoritmo Recursivo

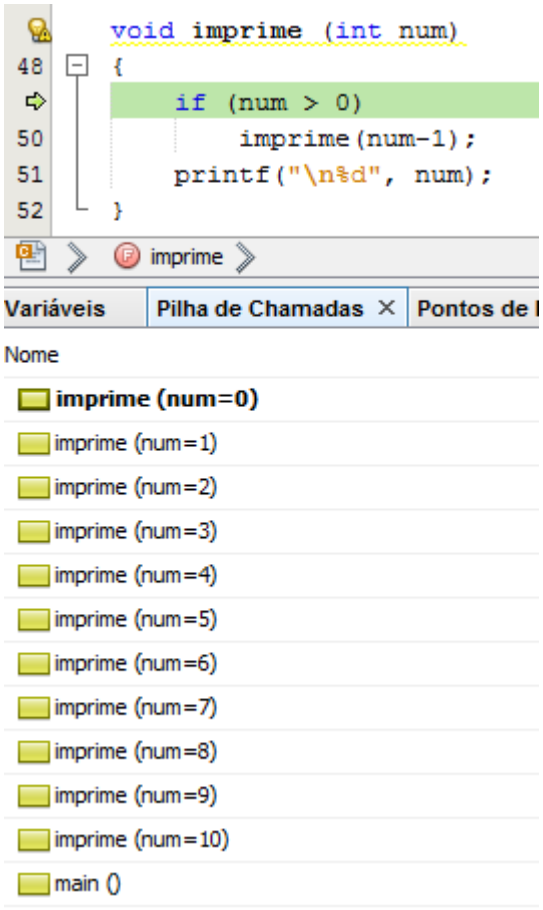
- Exemplo: Imprime recursivo

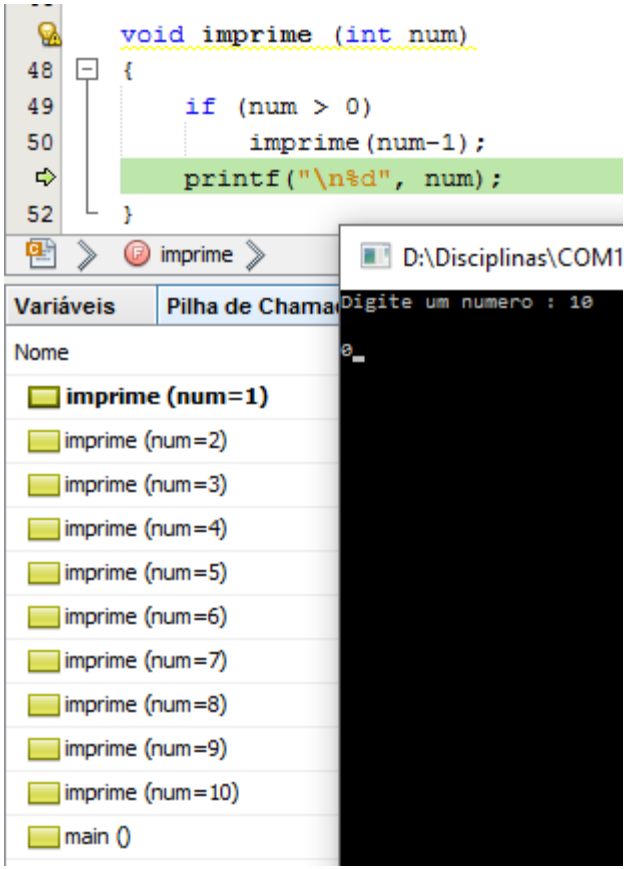
```
void imprime (int num)
{
    if (num > 0)
        imprime(num-1);
    printf("\n%d", num);
}
```

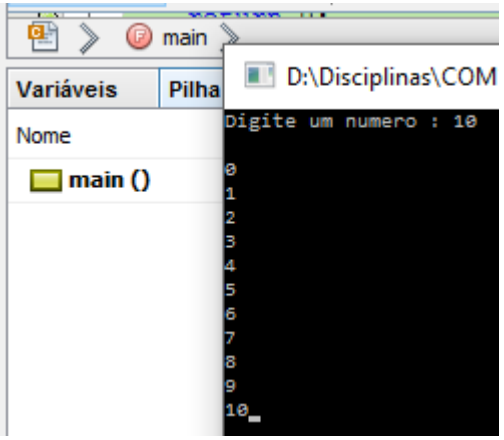
```
Digite um numero : 10
```

Algoritmo Recursivo

- Exemplo: Imprime recursivo







Recursividade

- Vantagens
  - Simplifica a solução de alguns problemas
  - Algoritmos recursivos são mais compactos para alguns tipos de algoritmo, mais legíveis e mais fáceis de ser compreendidos e implementados.

- Desvantagens
  - Por usarem intensamente a pilha de execução, os algoritmos recursivos tendem a ser mais lentos e a consumir mais memória que os iterativos, porém pode valer a pena sacrificar a eficiência em benefício da clareza.
  - Erros de implementação podem levar a estoure de pilha.

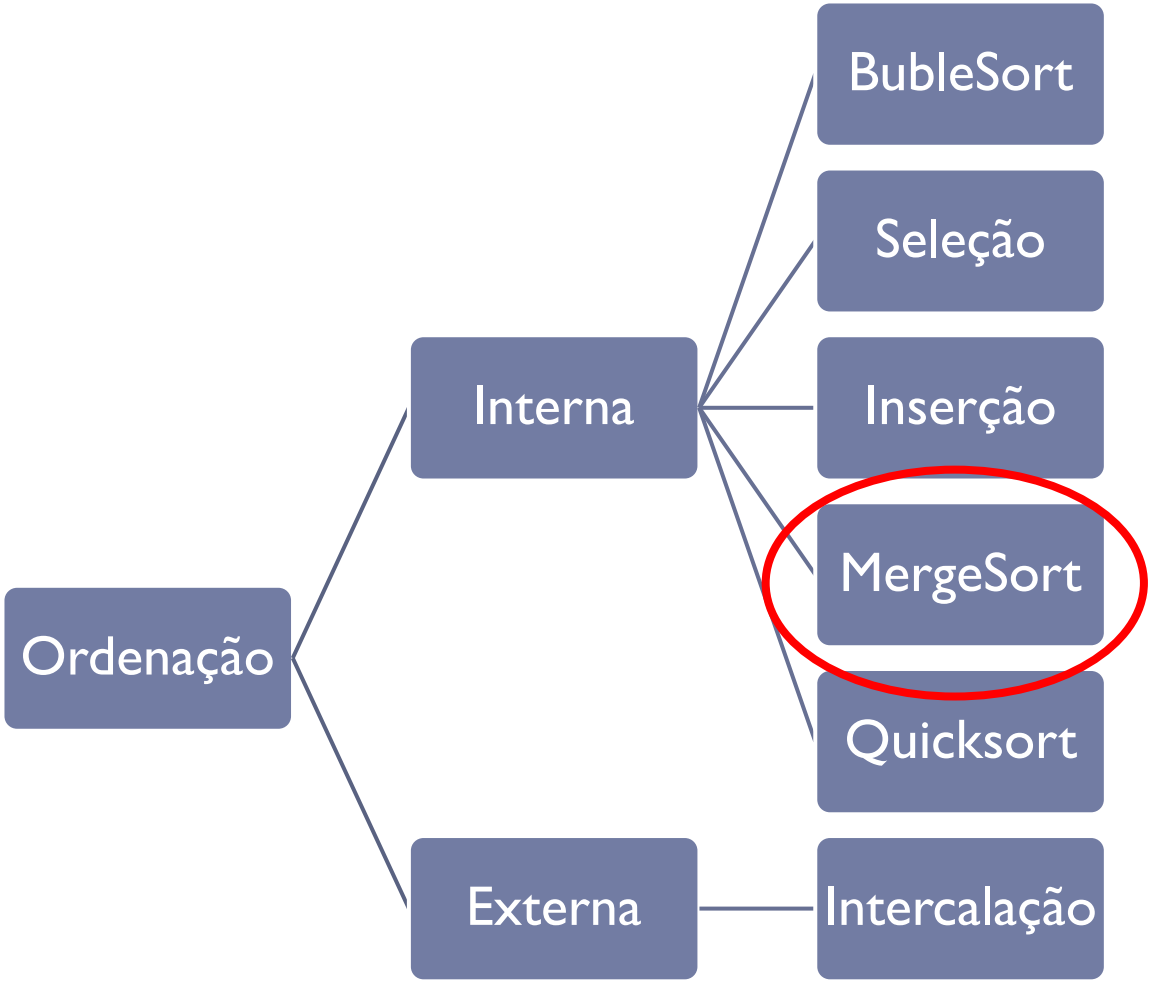
Recursividade

- Quadro comparativo da execução de algoritmos para o cálculo do Fibonacci.

n	10	20	30	50	100
Recursivo	8 ms	1 s	2 min	21 dias	10^9 anos
Iterativo	0.17 ms	0.33 ms	0.5 ms	0.75 ms	1.5 ms

- Evitar uso de recursividade quando existe uma solução óbvia por iteração.

Classificação dos Métodos de Ordenação



Ordenação por Fusão ou Intercalação

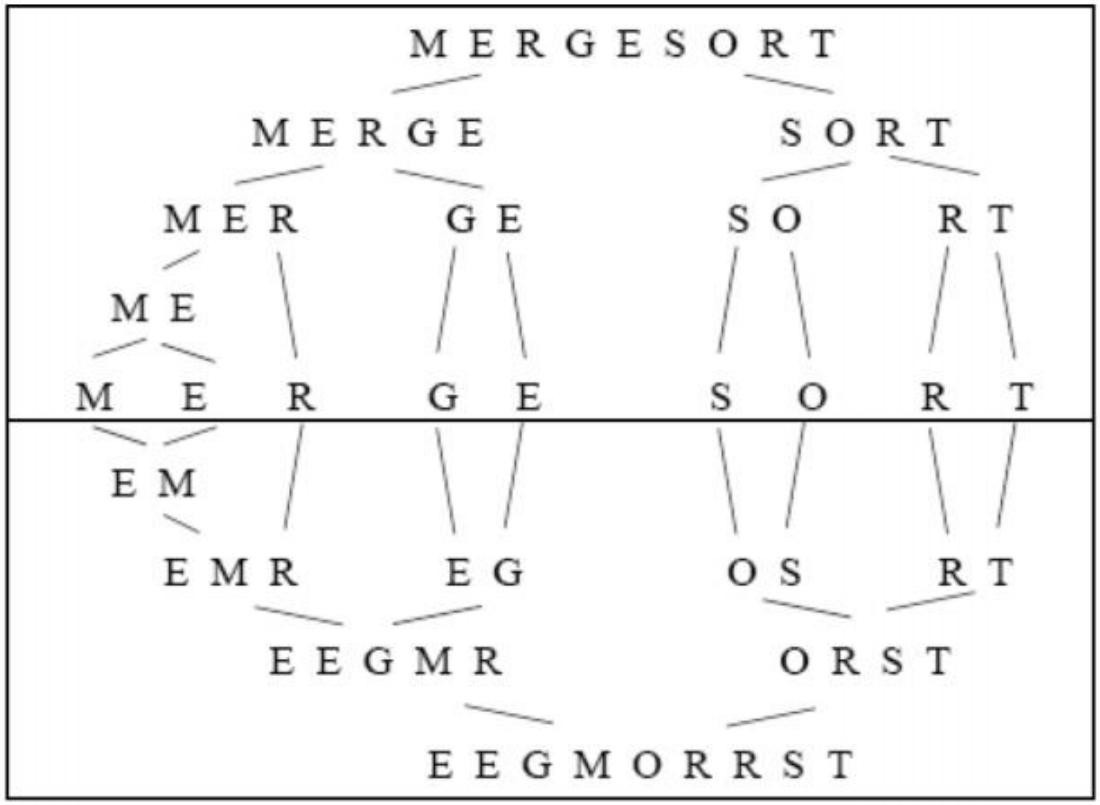
MergeSort

MergeSort

- **Ideia :** reduzir um problema em problemas menores, resolver cada um destes subproblemas e combinar as soluções parciais para obter a solução do problema original.

- É composto de duas fases:
  - **Divisão**
    - Divide o vetor original recursivamente em vetores menores
    - A divisão é feita sempre no meio do vetor
    - A divisão é feita até o vetor ficar com tamanho 1
  - **Intercalação ou Merge**
    - Intercala os elementos dos dois vetores ordenados para obter a ordenação total.

MergeSort



Fase de divisão

Fase de intercalação (merge)

MergeSort

- O mergeSort não é paralelo!

input	M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E
sort left half	E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E
sort right half	E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X
merge results	A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

MergeSort

5	3	7	6	2	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

5	3	7	6	2
0	1	2	3	4

5	3	7
0	1	2

5	3
0	1

5
0

3
1

FASE DA DIVISÃO

Lado esquerdo do vetor

MergeSort

5	3	7	6	2	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

5	3	7	6	2
0	1	2	3	4

5	3	7
0	1	2

5	3
0	1

5
0

3
1

INTERCALAÇÃO

MergeSort

5	3	7	6	2	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

5	3	7	6	2
0	1	2	3	4

5	3	7
0	1	2

5	3
0	1

5
0

3
1

INTERCALAÇÃO

Vetor Auxiliar

5	3
0	1

3	5
0	1

MergeSort

3	5	7	6	2	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

5	3	7	6	2
0	1	2	3	4

5	3	7
0	1	2

5
0

3
1

3	5
0	1

O vetor auxiliar é copiado para o vetor original nas suas devidas posições.

MergeSort

3	5	7	6	2	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

5	3	7	6	2
0	1	2	3	4

5	3	7
0	1	2

O algoritmo volta para a fase de divisão.

MergeSort

3	5	7	6	2	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

5	3	7	6	2
0	1	2	3	4

5	3	7
0	1	2

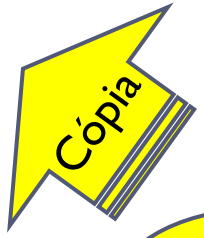
3	5
0	1

7
2

MergeSort

3	5	7	6	2	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

3	5	7	6	2
0	1	2	3	4



Vetor Auxiliar

3	5
0	1

7
2

3	5	7
0	1	2

INTERCALAÇÃO

MergeSort

3	5	7	6	2	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

5	3	7	6	2
0	1	2	3	4

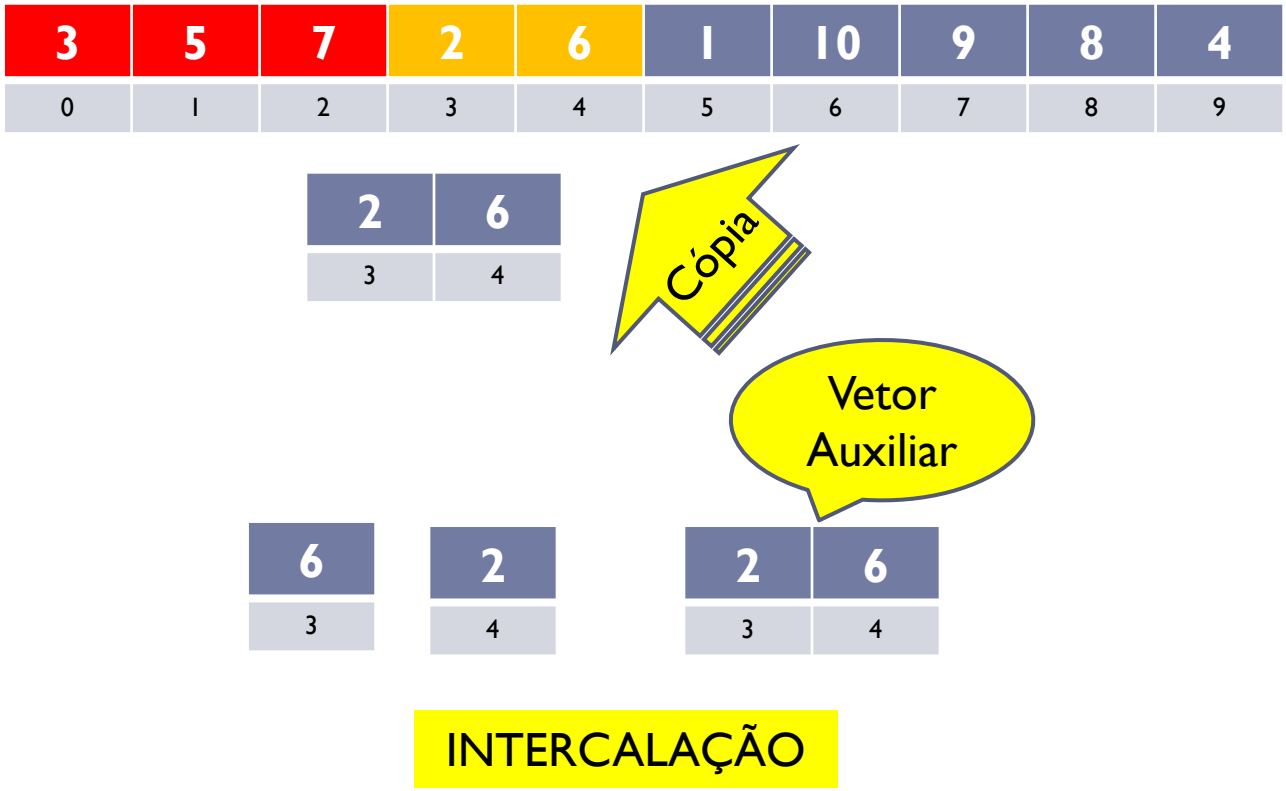
6	2
3	4

6
3

2
4

O algoritmo volta para a fase de divisão.

MergeSort


3 5 7 2 6 1 10 9 8 4
0 1 2 3 4 5 6 7 8 9
2 6
3 4
Cópia
Vetor Auxiliar
6 2 2 6
3 4 3 4
INTERCALAÇÃO

MergeSort

3	5	7	2	6	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

3	5	7	2	6
0	1	2	3	4

Vetor Auxiliar

3	5	7
0	1	2

2	6
3	4

2	3	5	6	7
0	1	2	3	4

INTERCALAÇÃO

Cópia

2	3	5	6	7	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

MergeSort

2	3	5	6	7	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

1	10	9	8	4
5	6	7	8	9

1	10	9
5	6	7

1	10
5	6

1
5

10
6

FASE DA DIVISÃO

Lado direito do vetor

MergeSort

2	3	5	6	7	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

1	10	9	8	4
5	6	7	8	9

1	10	9
5	6	7

Vetor Auxiliar

1
5

10
6

1	10
5	6

INTERCALAÇÃO

MergeSort

2	3	5	6	7	1	10	9	8	4
0	1	2	3	4	5	6	7	8	9

1	10	9	8	4
5	6	7	8	9

1	10	9
5	6	7

1	10
5	6

9
7

MergeSort

2	3	5	6	7	1	9	10	8	4
0	1	2	3	4	5	6	7	8	9

1	10	9	8	4
5	6	7	8	9

1	10	9
5	6	7

Vetor Auxiliar

1	10
5	6

9
7

1	9	10
5	6	7

INTERCALAÇÃO

MergeSort

2	3	5	6	7	1	9	10	8	4
0	1	2	3	4	5	6	7	8	9

1	9	10	8	4
5	6	7	8	9

8	4
8	9

8
8

4
9

MergeSort

2	3	5	6	7	1	9	10	4	8
0	1	2	3	4	5	6	7	8	9

1	9	10	8	4
5	6	7	8	9

Vetor Auxiliar

8
8

4
9

4	8
8	9

INTERCALAÇÃO

MergeSort

2	3	5	6	7	1	4	8	9	10
0	1	2	3	4	5	6	7	8	9

1	9	10	8	4
5	6	7	8	9

Vetor Auxiliar

1	9	10
5	6	7

4	8
8	9

1	4	8	9	10
5	6	7	8	9

INTERCALAÇÃO

MergeSort

2	3	5	6	7	1	4	8	9	10
0	1	2	3	4	5	6	7	8	9

2	3	5	6	7
0	1	2	3	4

1	4	8	9	10
5	6	7	8	9

INTERCALAÇÃO

Vetor Auxiliar

1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9

MergeSort

1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9

COPIA TODO O VETOR AUXILIAR PARA O VETOR PRINCIPAL

MergeSort

Algoritmo: ORDENAÇÃO - MERGESORT

```
1  void MergeSort(vet, inicio, fim):
2      int meio;
3      se (inicio < fim) então
4          meio ← ⌊(inicio+fim)/2⌋
5          MergeSort(V, inicio, meio)
6          MergeSort(V, meio+1, fim)
7          Merge(V, inicio, meio, fim)
8      fim
9  fim

10 void Merge(vet, inicio, meio, fim):
11     int marcadorV1 ← inicio; // Controla as posições do vetor lógico 1.
12     int marcadorV2 ← meio + 1; // Controla as posições do vetor lógico 2.
13     int vetAux[(fim-inicio)+1]; // Vetor auxiliar.
14     int i ← 0; // Contador do vetor auxiliar.
15     int k; // Contador de vet para cópia final.
16     enquanto (marcadorV1 ≤ meio) E (marcadorV2 ≤ fim) faça
17         se vet[marcadorV1] < vet[marcadorV2] então
18             vetAux[i] = vet[marcadorV1];
19             marcadorV1 + +;
20         senão
21             vetAux[i] = vet[marcadorV2];
22             marcadorV2 + +;
23         fim
24         i + +;
25     fim
26     Copia o resto do vetor 1 ou o resto de vetor 2 para o vetor auxiliar
27     Copia o vetor auxiliar para o vetor original
28 fim
```

MergeSort

- Exercício
  - Executar a ordenação do vetor abaixo usando o MergeSort. Ao final você deve apresentar a pilha de execução.

18	7	1	3	8	6

MergeSort

- A eficiência do algoritmo depende de quão eficientemente será a intercalação dos dois vetores (ordenados) em um único vetor ordenado.

- A intercalação pode ser feita fazendo-se, no máximo, (n – 1) comparações, onde *n* é o número total de elementos dos dois vetores originais, ou seja, o algoritmo de intercalação é O(n).

- Como o número de elementos do vetor é reduzido à metade em cada chamada do mergesort, o número total de "rodadas" é $\log_2 n$.

- Assim, a complexidade assintótica do MergeSort é O(nlogn)

MergeSort

- O MergeSort é considerado um algoritmo ótimo, uma vez que ele tem sempre a mesma complexidade.

- No entanto, o MergeSort ocupa mais espaço na memória para fazer a intercalação dos sub-vetores.

MergeSort

- Exercício
  - Executar a ordenação do vetor abaixo usando o MergeSort. Ao final você deve apresentar a pilha de execução.

22	33	55	77	99	11	44	66	88

MergeSort

- https://visualgo.net/en/sorting
- https://www.hackerearth.com/practice/algorithms/sorting/selection-sort/visualize/

Trabalho Prático

- Dividir a turma
  - 1 dupla
  - 8 grupos de 5 pessoas