Métodos de Ordenação: Selection, Insertion, Bubble, Merge (Sort)

Hebert Coelho e Nádia Félix

Ordenação

É a operação de rearranjar os dados em uma determinada ordem.

Problema da ordenação - Definição formal [1]

Entrada: Uma sequência de n números $\langle a_1, a_2, \ldots, a_n \rangle$.
Saída: Uma permutação (reordenada) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ da sequência de entrada tal que $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

Ordenação

- A eficiência no manuseio de dados muitas vezes pode ser aumentada se os dados estiverem dispostos de acordo com algum critério de ordem
  - Exemplo: Uma lista telefônica sem ordem.

POLICIA LOCAL ORIHUELA COSTA	966 760 000
GUARDIA CIVIL	966 796 143
AMBULANCIA SAMUR	112
BOMBEROS	965 300 080
HOSPITAL VEGA BAJA	965 367 054
CLINICA INTERNATIONAL (PRIVADA)	966 765 138
FARMACIA LA FUENTE	965 300 099
AYUNTAMIENTO ORIHUELA COSTA	966 760 000
OFICINA TURISMO ORIHUELA COSTA	966 760 000
AEROPUERTO - ALICANTE: EL ALTET	966 919 100
AEROPUERTO - MURCIA: SAN JAVIER	968 172 025
VICTIMAS VIOLENCIA DOMESTICA	016

Ordenação

- Do ponto da memória do computador, os algoritmos de ordenação podem ser classificados em:
  1. **Ordenação Interna** (quando os dados a serem ordenados estão na memória principal).

Ordenação

- Do ponto da memória do computador, os algoritmos de ordenação podem ser classificados em:
  1. **Ordenação Interna** (quando os dados a serem ordenados estão na memória principal).
  2. **Ordenação Externa** (quando os dados a serem ordenados necessitam de armazenamento em memória auxiliar como por exemplo o disco HD).

Nesta disciplina estamos interessados em métodos de ordenação interna.

Ordenação Interna

Na escolha de um algoritmo de ordenação interna deve ser considerado principalmente:

- O tempo gasto pela ordenação;
- O uso econômico da memória disponível;

Ordenação Interna

- A maioria dos métodos de ordenação é baseado em **comparação de chaves**;
  - Exemplos: insertionsort, selectionsort, etc;
- Existem métodos de ordenação que utilizam o princípio da **distribuição**;
  - Exemplos: radixsort, bucketsort, etc;

Ordenação Interna

Algoritmos de Ordenação

Algoritmos de ordenação podem ser aplicados a diversos tipos de estrutura, tais como:

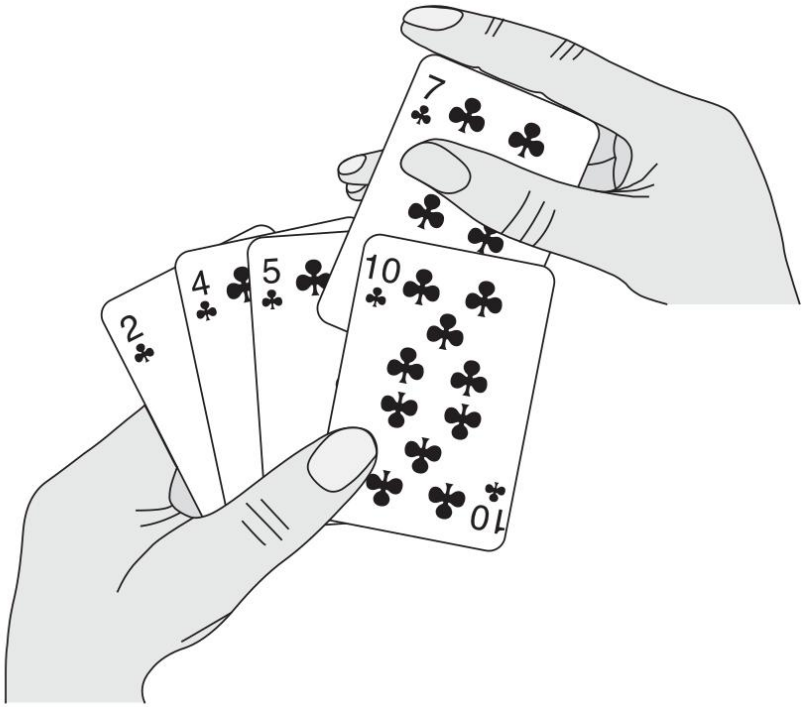
- Vetores;
- Matrizes;
- Estruturas dinâmicas.

Classificação

Um algoritmo de ordenação é **estável** se preserva ordem relativa de itens com valores idênticos.

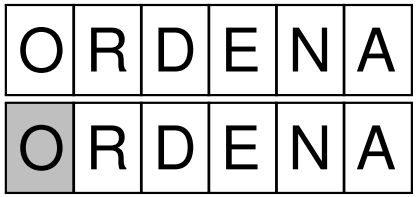
Ordenação por inserção (InsertionSort)

A idéia da ordenação por inserção é dividir os elementos em duas subestruturas, uma com os elementos já ordenados e outra com elementos ainda por ordenar.

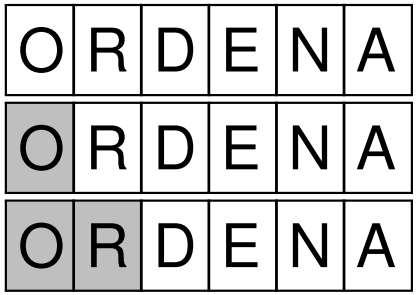


Ordenação por inserção (InsertionSort) - Exemplo

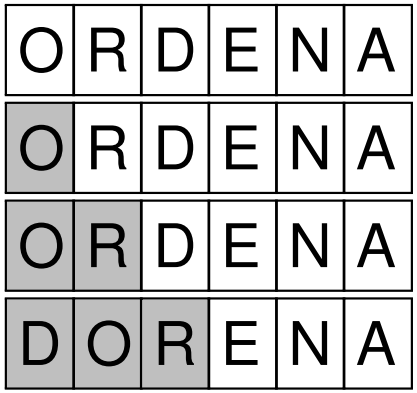
Ordenação por inserção (InsertionSort) - Exemplo



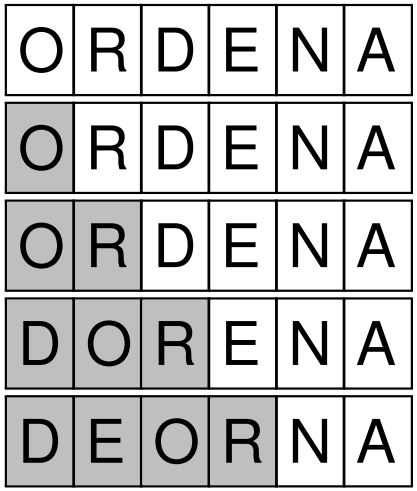
Ordenação por inserção (InsertionSort) - Exemplo



Ordenação por inserção (InsertionSort) - Exemplo



Ordenação por inserção (InsertionSort) - Exemplo



Ordenação por inserção (InsertionSort) - Exemplo

O	R	D	E	N	A
O	R	D	E	N	A
O	R	D	E	N	A
D	O	R	E	N	A
D	E	O	R	N	A
D	E	N	O	R	A

Ordenação por inserção (InsertionSort) - Exemplo

O	R	D	E	N	A
O	R	D	E	N	A
O	R	D	E	N	A
D	O	R	E	N	A
D	E	O	R	N	A
D	E	N	O	R	A
A	D	E	N	O	R

Intercultural Computer Science Education
https://www.youtube.com/watch?v=ROaIU379l3U

Ordenação por inserção (InsertionSort) - código em C

```
1  void insertionSort( int* v, int n ) //n é tamanho do vetor
2  {
3    int i= 0;
4    int j= 1;
5    int aux = 0;
6    while (j < n)
7    {
8      aux = v[j];
9      i = j - 1;
10     while ((i >= 0) && (v[i] > aux))
11     {
12       v[i + 1] = v[i];
13       i = i - 1;
14     }
15     v[i + 1] = aux;
16     j = j + 1;
17   }
18 }
```

Ordenação por inserção (InsertionSort) - Exercícios

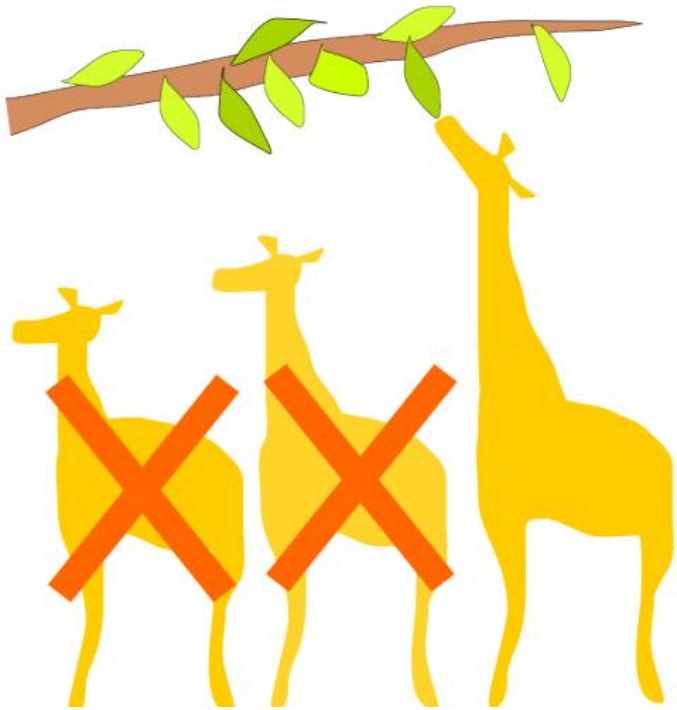
1. Executar a função insertionSort para um vetor de tamanho 5 em ordem crescente e contar o número de vezes que a linha 12 é executada;
2. Executar a função insertionSort para um vetor de tamanho 5 em ordem decrescente e contar o número de vezes que a linha 12 é executada;
3. Qual é o pior caso e o melhor caso para este algoritmo?
4. A ordenação por inserção é estável?

Ordenação por inserção (InsertionSort)

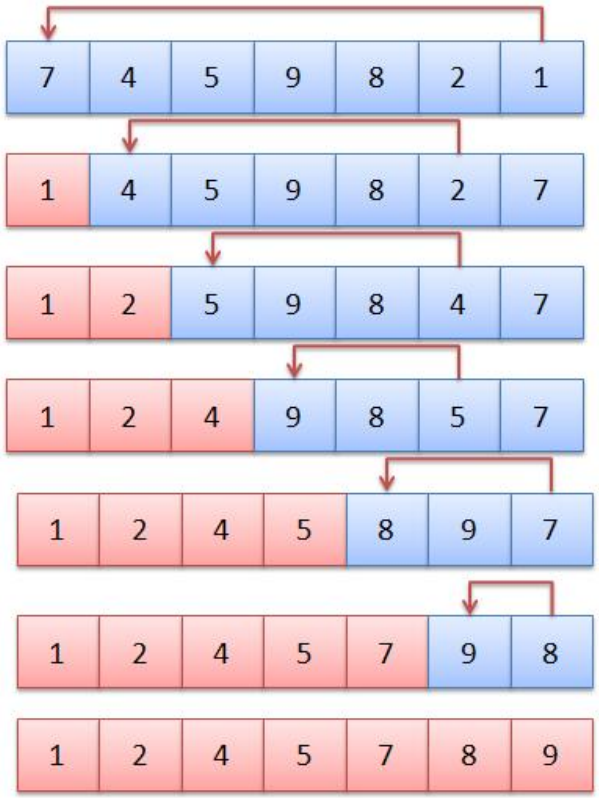
- O número mínimo de comparações e movimentos ocorre quando os itens estão originalmente em ordem.
- O número máximo de comparações e movimentos ocorre quando os itens estão originalmente em ordem reversa.
- Bom método a ser usado quando a sequência esta quase ordenada, ou quando se deseja adicionar poucos itens a uma sequência já ordenada.
- O algoritmo de ordenação por inserção é estável.

Ordenação por seleção (SelectionSort)

- A idéia da ordenação por seleção é procurar o menor elemento do vetor (ou maior) e movimentá-lo para a primeira (última) posição do vetor.
- Repetir para os *n* elementos do vetor.



Ordenação por seleção (SelectionSort) - Exemplo



Intercultural Computer Science Education
https://www.youtube.com/watch?v=Ns4TPTC8whw

SelectionSort - código em C

```
1  void selection_sort(int* v, int n) { //n é tamanho do vetor
2    int i, j, min, aux;
3    for (i = 0; i < (n-1); i++)
4    {
5        min = i;
6        for (j = (i+1); j < n; j++) {
7          if(v[j] < v[min])
8            min = j;
9        }
10       if (v[i] != v[min]) {
11         aux = v[i];
12         v[i] = v[min];
13         v[min] = aux;
14       }
15   }
16 }
```

SelectionSort - Exercícios

1. Determine o melhor e o pior caso para o SelectionSort?
2. A ordenação por seleção é estável?

SelectionSort - Vantagens e Desvantagens

1. Uma vantagem do Selection Sort é que entre os algoritmos de ordenação ele apresenta uma das menores quantidades de movimentos entre os elementos, assim pode haver algum ganho quando se necessita ordenar estruturas complexas.
2. Uma desvantagem é que o número de comparações é igual para o melhor caso, caso médio e o pior caso. Assim, mesmo que o vetor esteja ordenado o custo continua quadrático (n^2).
3. Não é estável (depende da implementação).

Ordenação por bolhas (BubbleSort)

- A idéia da ordenação por bolhas é flutuar o maior elemento para o fim.
- Repita *n* vezes a flutuação.

Ordenação por bolhas (BubbleSort) - Exemplo

Usando “visualgo” para rodar exemplos

Vamos rodar um exemplo com:
https://visualgo.net/en/sorting

Intercultural Computer Science Education
https://www.youtube.com/watch?v=IyZQPjUT5B4

BubbleSort - código em C

```
1  void BubbleSort(int* v, int n) { //n é tamanho do vetor
2      int i, fim, aux;
3      for (fim = n-1; fim > 0; --fim) {
4          for (i = 0; i < fim; ++i) {
5              if (v[i] > v[i+1]) {
6                  aux = v[i];
7                  v[i] = v[i+1];
8                  v[i+1] = aux;
9              }
10         }
11     }
12 }
```

Exercícios

1. Determine o melhor e o pior caso para o BubbleSort?
2. Como o algoritmo BubbleSort apresentado pode ser melhorado?
3. O BubbleSort é estável?

BubbleSort - Vantagens e Desvantagens

1. Simples de entender e implementar.
2. Uma desvantagem é que na prática ele tem execução lenta mesmo quando comparado a outros algoritmos quadráticos (n^2).
3. Tem um número muito grande de movimentação de elementos, assim não deve ser usado se a estrutura a ser ordenada for complexa.

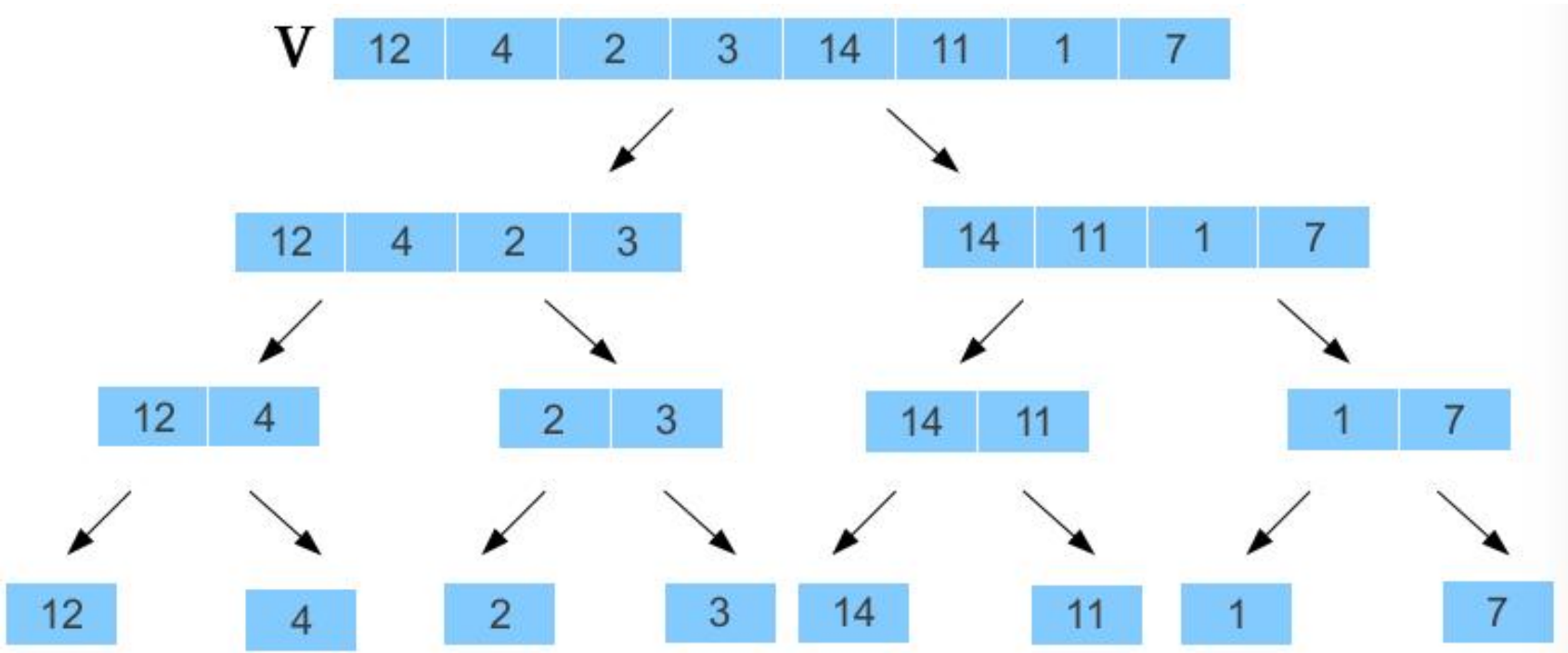
MergeSort (intercalação direta) - Ideia

- divida o vetor em 2 subvetores (ao meio) recursivamente até ele ter o tamanho 1.
- Intercale pares de elementos adjacentes. Repita esse processo até restar apenas um arquivo de tamanho n.

MergeSort (intercalação direta) - Ideia

- Dividir e Conquistar;
- Divide, recursivamente, o conjunto de dados até que o subconjunto possua 1 elemento
- Combina 2 subconjuntos de forma a obter 1 conjunto maior e ordenado
- Esse processo se repete até que exista apenas 1 conjunto.

MergeSort (intercalação direta) - Exemplo



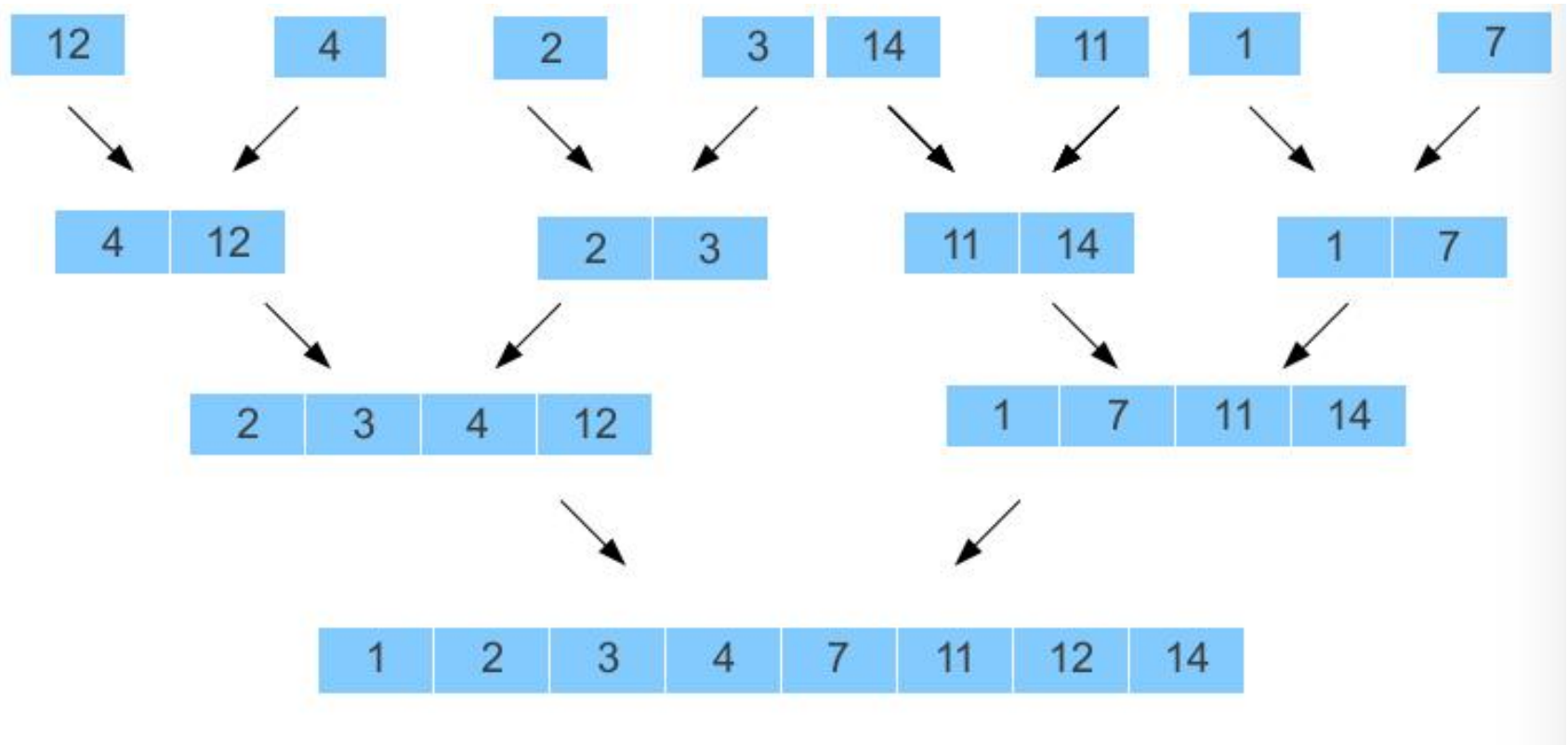
MergeSort - código em C

```
1 void mergeSort(int* vetor, int inicio, int fim){
2     if (inicio < fim) {
3         int meio = (inicio+fim)/2;
4         mergeSort(vetor, inicio, meio);
5         mergeSort(vetor, meio+1, fim);
6         merge(vetor, inicio, meio, fim);
7     }
8 }
```

MergeSort (intercalação direta) - Ideia

- divida o vetor em 2 subvetores (ao meio) recursivamente até ele ter o tamanho 1.
- **Intercale pares de elementos adjacentes**. Repita esse processo até restar apenas um arquivo de tamanho n.

MergeSort (intercalação direta) - Exemplo



Intercultural Computer Science Education
https://www.youtube.com/watch?v=XaqR3G_NVoo

MergeSort - código em C

```
1  void merge(int vetor[], int inicio, int meio, int fim) {
2      int com1 = inicio, com2 = meio+1, comAux = 0, vetAux[fim-inicio+1];
3      while(com1<=meio && com2<=fim){
4          if(vetor[com1] <= vetor[com2]){
5              vetAux[comAux] = vetor[com1];
6              com1++;
7          }else{
8              vetAux[comAux] = vetor[com2];
9              com2++; }
10         comAux++; }
11     while(com1<=meio){ //Caso ainda haja elementos na primeira metade
12         vetAux[comAux] = vetor[com1];
13         comAux++;com1++; }
14     while(com2<=fim){ //Caso ainda haja elementos na segunda metade
15         vetAux[comAux] = vetor[com2];
16         comAux++;com2++; }
17     for(comAux=inicio;comAux<=fim;comAux++){ //Move os elementos de volta
           para o vetor original
18         vetor[comAux] = vetAux[comAux-inicio];
19     }
20 }
```

Exercícios

1. Determine o melhor e o pior caso para o MergeSort?
2. O MergeSort é estável?

MergeSort (intercalação direta) - Vantagens e Desvantagens

1. Método de ordenação com tempo:
   1. Melhor caso: $n * log_2 n$;
   2. Pior caso:$n * log_2 n$;
2. Pode ser adaptado para ordenação de arquivos externos (memória secundária).
3. Utiliza mais memória para poder ordenar (vetor auxiliar).
4. O MergeSort é estável: não altera a ordem de dados iguais.

Bibliografia

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L. e STEIN, C. *Introduction to Algorithms*, 3ª edição, MIT Press, 2009.

ZIVIANI, N. Projeto de Algoritmos: com implementações em Pascal e C, 2ª edição, Cengage Learning, 2009.

Slides dos professores Humberto Longo, Márcia Cappelle, Marcelo Quinta e Marcos Roriz

wikipedia.