

# COM112 - ALGORITMO E ESTRUTURA DE DADOS II<sup>1</sup>

Pedro Henrique Del Bianco Hokama  
UNIFEI

---

<sup>1</sup>Baseado nos slides elaborados por Charles Ornelas Almeida, Israel Guerra e Nivio Ziviani

# Ordenação

# Ordenação

- Ordenar: processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.

# Ordenação

- Ordenar: processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.

# Ordenação

- Ordenar: processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.
  - ▶ Dificuldade de se utilizar um catálogo telefônico se os nomes das pessoas não estivessem listados em ordem alfabética.

# Ordenação

- Ordenar: processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.
  - ▶ Dificuldade de se utilizar um catálogo telefônico se os nomes das pessoas não estivessem listados em ordem alfabética.
- Notação utilizada nos algoritmos:

# Ordenação

- Ordenar: processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.
  - ▶ Dificuldade de se utilizar um catálogo telefônico se os nomes das pessoas não estivessem listados em ordem alfabética.
- Notação utilizada nos algoritmos:
  - ▶ Os algoritmos trabalham sobre os registros de um arquivo ou dados na memória.

# Ordenação

- Ordenar: processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.
  - ▶ Dificuldade de se utilizar um catálogo telefônico se os nomes das pessoas não estivessem listados em ordem alfabética.
- Notação utilizada nos algoritmos:
  - ▶ Os algoritmos trabalham sobre os registros de um arquivo ou dados na memória.
  - ▶ Cada registro possui uma chave utilizada para controlar a ordenação.



# Ordenação

- Ordenar: processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.
  - ▶ Dificuldade de se utilizar um catálogo telefônico se os nomes das pessoas não estivessem listados em ordem alfabética.
- Notação utilizada nos algoritmos:
  - ▶ Os algoritmos trabalham sobre os registros de um arquivo ou dados na memória.
  - ▶ Cada registro possui uma chave utilizada para controlar a ordenação.
  - ▶ Podem existir outros componentes em um registro.

# Ordenação

- Estrutura de um registro:

```
typedef long TipoChave;  
typedef struct Tipoltem{  
    TipoChave Chave;  
    /* Outros componentes */  
} Tipoltem;
```

# Ordenação

- Estrutura de um registro:

```
typedef long TipoChave;  
typedef struct Tipoltem{  
    TipoChave Chave;  
    /* Outros componentes */  
} Tipoltem;
```

- Qualquer tipo de chave sobre o qual exista uma regra de ordenação bem-definida pode ser utilizado.

# Ordenação

- Estrutura de um registro:

```
typedef long TipoChave;  
typedef struct Tipoltem{  
    TipoChave Chave;  
    /* Outros componentes */  
} Tipoltem;
```

- Qualquer tipo de chave sobre o qual exista uma regra de ordenação bem-definida pode ser utilizado.
- Um método de ordenação é estável se a ordem relativa dos itens com chaves iguais não se altera durante a ordenação.

# Ordenação

# Ordenação

- Alguns dos métodos de ordenação mais eficientes não são estáveis.

# Ordenação

- Alguns dos métodos de ordenação mais eficientes não são estáveis.
- A estabilidade pode ser forçada quando o método é não-estável.

# Ordenação

- Alguns dos métodos de ordenação mais eficientes não são estáveis.
- A estabilidade pode ser forçada quando o método é não-estável.
- Sedgewick (1988) sugere agregar um pequeno índice a cada chave antes de ordenar, ou então aumentar a chave de alguma outra forma



# Ordenação

# Ordenação

- Classificação dos métodos de ordenação:

# Ordenação

- Classificação dos métodos de ordenação:
  - ▶ Interna: arquivo a ser ordenado cabe todo na memória principal.

# Ordenação

- Classificação dos métodos de ordenação:
  - ▶ Interna: arquivo a ser ordenado cabe todo na memória principal.
  - ▶ Externa: arquivo a ser ordenado não cabe na memória principal.

# Ordenação

- Classificação dos métodos de ordenação:
  - ▶ Interna: arquivo a ser ordenado cabe todo na memória principal.
  - ▶ Externa: arquivo a ser ordenado não cabe na memória principal.
- Diferenças entre os métodos:

# Ordenação

- Classificação dos métodos de ordenação:
  - ▶ Interna: arquivo a ser ordenado cabe todo na memória principal.
  - ▶ Externa: arquivo a ser ordenado não cabe na memória principal.
- Diferenças entre os métodos:
  - ▶ Em um método de ordenação interna, qualquer registro pode ser imediatamente acessado.

# Ordenação

- Classificação dos métodos de ordenação:
  - ▶ Interna: arquivo a ser ordenado cabe todo na memória principal.
  - ▶ Externa: arquivo a ser ordenado não cabe na memória principal.
- Diferenças entre os métodos:
  - ▶ Em um método de ordenação interna, qualquer registro pode ser imediatamente acessado.
  - ▶ Em um método de ordenação externa, os registros são acessados sequencialmente ou em grandes blocos.

# Ordenação

- Classificação dos métodos de ordenação:
  - ▶ Interna: arquivo a ser ordenado cabe todo na memória principal.
  - ▶ Externa: arquivo a ser ordenado não cabe na memória principal.
- Diferenças entre os métodos:
  - ▶ Em um método de ordenação interna, qualquer registro pode ser imediatamente acessado.
  - ▶ Em um método de ordenação externa, os registros são acessados sequencialmente ou em grandes blocos.
- A maioria dos métodos de ordenação é baseada em comparações das chaves.



# Ordenação

- Classificação dos métodos de ordenação:
  - ▶ Interna: arquivo a ser ordenado cabe todo na memória principal.
  - ▶ Externa: arquivo a ser ordenado não cabe na memória principal.
- Diferenças entre os métodos:
  - ▶ Em um método de ordenação interna, qualquer registro pode ser imediatamente acessado.
  - ▶ Em um método de ordenação externa, os registros são acessados sequencialmente ou em grandes blocos.
- A maioria dos métodos de ordenação é baseada em comparações das chaves.
- Existem métodos de ordenação que utilizam o princípio da distribuição.

# Ordenação por distribuição

# Ordenação por distribuição

- Exemplo de ordenação por distribuição: considere o problema de ordenar um baralho com 52 cartas na ordem:

$\clubsuit < \diamondsuit < \heartsuit < \spadesuit$

desempatando por

$A < 2 < 3 < \dots < 10 < J < Q < K$

# Ordenação por distribuição

- Exemplo de ordenação por distribuição: considere o problema de ordenar um baralho com 52 cartas na ordem:

 <  <  < 

desempatando por

$A < 2 < 3 < \dots < 10 < J < Q < K$

- Algoritmo:
  - 1 Distribuir as cartas em treze montes: ases, dois, três, . . . , reis.
  - 2 Colete os montes na ordem contrária, de forma que o  $K$  fique em cima.
  - 3 Distribua novamente as cartas em quatro montes: paus, ouros, copas e espadas.
  - 4 Colete os montes na ordem especificada.

# Ordenação por distribuição

# Ordenação por distribuição

- Métodos como o ilustrado são também conhecidos como **ordenação digital**, **radixsort** ou **bucketsort**.

# Ordenação por distribuição

- Métodos como o ilustrado são também conhecidos como **ordenação digital**, **radixsort** ou **bucketsort**.
- O método não utiliza comparação entre chaves.

# Ordenação por distribuição

- Métodos como o ilustrado são também conhecidos como **ordenação digital**, **radixsort** ou **bucketsort**.
- O método não utiliza comparação entre chaves.
- Uma das dificuldades de implementar este método está relacionada com o problema de lidar com cada monte.



# Ordenação por distribuição

- Métodos como o ilustrado são também conhecidos como **ordenação digital**, **radixsort** ou **bucketsort**.
- O método não utiliza comparação entre chaves.
- Uma das dificuldades de implementar este método está relacionada com o problema de lidar com cada monte.
- Se para cada monte nós reservarmos uma área, então a demanda por memória extra pode tornar-se proibitiva.

# Ordenação por distribuição

- Métodos como o ilustrado são também conhecidos como **ordenação digital**, **radixsort** ou **bucketsort**.
- O método não utiliza comparação entre chaves.
- Uma das dificuldades de implementar este método está relacionada com o problema de lidar com cada monte.
- Se para cada monte nós reservarmos uma área, então a demanda por memória extra pode tornar-se proibitiva.
- Sabendo a priori a distribuição das cartas o custo para ordenar um arquivo com  $n$  elementos é da ordem de  $O(n)$ .

# Ordenação Interna

# Ordenação Interna

- Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.

# Ordenação Interna

- Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.
- Sendo  $n$  o número registros no arquivo, as medidas de complexidade relevantes são:

# Ordenação Interna

- Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.
- Sendo  $n$  o número registros no arquivo, as medidas de complexidade relevantes são:
  - ▶ Número de comparações  $C(n)$  entre chaves.

# Ordenação Interna

- Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.
- Sendo  $n$  o número registros no arquivo, as medidas de complexidade relevantes são:
  - ▶ Número de comparações  $C(n)$  entre chaves.
  - ▶ Número de movimentações  $M(n)$  de itens do arquivo.

# Ordenação Interna

- Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.
- Sendo  $n$  o número registros no arquivo, as medidas de complexidade relevantes são:
  - ▶ Número de comparações  $C(n)$  entre chaves.
  - ▶ Número de movimentações  $M(n)$  de itens do arquivo.
- O uso econômico da memória disponível é um requisito primordial na ordenação interna.



# Ordenação Interna

- Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.
- Sendo  $n$  o número registros no arquivo, as medidas de complexidade relevantes são:
  - ▶ Número de comparações  $C(n)$  entre chaves.
  - ▶ Número de movimentações  $M(n)$  de itens do arquivo.
- O uso econômico da memória disponível é um requisito primordial na ordenação interna.
- Métodos de ordenação **in situ** são os preferidos.

# Ordenação Interna

- Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.
- Sendo  $n$  o número registros no arquivo, as medidas de complexidade relevantes são:
  - ▶ Número de comparações  $C(n)$  entre chaves.
  - ▶ Número de movimentações  $M(n)$  de itens do arquivo.
- O uso econômico da memória disponível é um requisito primordial na ordenação interna.
- Métodos de ordenação **in situ** são os preferidos.
- Métodos que utilizam listas encadeadas não são muito utilizados.

# Ordenação Interna

- Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.
- Sendo  $n$  o número registros no arquivo, as medidas de complexidade relevantes são:
  - ▶ Número de comparações  $C(n)$  entre chaves.
  - ▶ Número de movimentações  $M(n)$  de itens do arquivo.
- O uso econômico da memória disponível é um requisito primordial na ordenação interna.
- Métodos de ordenação **in situ** são os preferidos.
- Métodos que utilizam listas encadeadas não são muito utilizados.
- Métodos que fazem cópias dos itens a serem ordenados possuem menor importância.

# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

- Métodos simples:

# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

- Métodos simples:
  - ▶ Adequados para pequenos arquivos.

# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

- Métodos simples:
  - ▶ Adequados para pequenos arquivos.
  - ▶ Requerem  $O(n^2)$  comparações.

# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

- Métodos simples:
  - ▶ Adequados para pequenos arquivos.
  - ▶ Requerem  $O(n^2)$  comparações.
  - ▶ Produzem programas pequenos.



# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

- Métodos simples:
  - ▶ Adequados para pequenos arquivos.
  - ▶ Requerem  $O(n^2)$  comparações.
  - ▶ Produzem programas pequenos.
- Métodos eficientes:

# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

- Métodos simples:
  - ▶ Adequados para pequenos arquivos.
  - ▶ Requerem  $O(n^2)$  comparações.
  - ▶ Produzem programas pequenos.
- Métodos eficientes:
  - ▶ Adequados para arquivos maiores.

# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

- Métodos simples:
  - ▶ Adequados para pequenos arquivos.
  - ▶ Requerem  $O(n^2)$  comparações.
  - ▶ Produzem programas pequenos.
- Métodos eficientes:
  - ▶ Adequados para arquivos maiores.
  - ▶ Requerem  $O(n \log n)$  comparações.

# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

- Métodos simples:
  - ▶ Adequados para pequenos arquivos.
  - ▶ Requerem  $O(n^2)$  comparações.
  - ▶ Produzem programas pequenos.
- Métodos eficientes:
  - ▶ Adequados para arquivos maiores.
  - ▶ Requerem  $O(n \log n)$  comparações.
  - ▶ Usam menos comparações.

# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

- Métodos simples:
  - ▶ Adequados para pequenos arquivos.
  - ▶ Requerem  $O(n^2)$  comparações.
  - ▶ Produzem programas pequenos.
- Métodos eficientes:
  - ▶ Adequados para arquivos maiores.
  - ▶ Requerem  $O(n \log n)$  comparações.
  - ▶ Usam menos comparações.
  - ▶ As comparações são mais complexas nos detalhes.

# Ordenação Interna

Ziviani classifica os métodos de ordenação interna:

- Métodos simples:
  - ▶ Adequados para pequenos arquivos.
  - ▶ Requerem  $O(n^2)$  comparações.
  - ▶ Produzem programas pequenos.
- Métodos eficientes:
  - ▶ Adequados para arquivos maiores.
  - ▶ Requerem  $O(n \log n)$  comparações.
  - ▶ Usam menos comparações.
  - ▶ As comparações são mais complexas nos detalhes.
  - ▶ Métodos simples são mais eficientes para pequenos arquivos.

# Ordenação Interna

# Ordenação Interna

- Tipos de dados e variáveis utilizados nos algoritmos de ordenação interna:

```
typedef int TipoIndice;  
typedef TipoItem TipoVetor[MAXTAM + 1];  
/* MAXTAM + 1 por causa da sentinela */  
TipoVetor A;
```



# Ordenação Interna

- Tipos de dados e variáveis utilizados nos algoritmos de ordenação interna:

```
typedef int TipoIndice;  
typedef TipoItem TipoVetor[MAXTAM + 1];  
/* MAXTAM + 1 por causa da sentinela */  
TipoVetor A;
```

- O índice do vetor vai de 0 até *MaxTam*, devido às chaves sentinelas.

# Ordenação Interna

- Tipos de dados e variáveis utilizados nos algoritmos de ordenação interna:

```
typedef int TipoIndice;  
typedef TipoItem TipoVetor[MAXTAM + 1];  
/* MAXTAM + 1 por causa da sentinela */  
TipoVetor A;
```

- O índice do vetor vai de 0 até *MaxTam*, devido às chaves sentinelas.
- O vetor a ser ordenado contém chaves nas posições de 1 até *n*.

# Ordenação por Seleção

# Ordenação por Seleção

- Um dos algoritmos mais simples de ordenação.

# Ordenação por Seleção

- Um dos algoritmos mais simples de ordenação.
- Algoritmo:

# Ordenação por Seleção

- Um dos algoritmos mais simples de ordenação.
- Algoritmo:
  - ▶ Selecione o menor item do vetor.

# Ordenação por Seleção

- Um dos algoritmos mais simples de ordenação.
- Algoritmo:
  - ▶ Selecione o menor item do vetor.
  - ▶ Troque-o com o item da primeira posição do vetor.

# Ordenação por Seleção

- Um dos algoritmos mais simples de ordenação.
- Algoritmo:
  - ▶ Selecione o menor item do vetor.
  - ▶ Troque-o com o item da primeira posição do vetor.
  - ▶ Repita essas duas operações com os  $n - 1$  itens restantes, depois com os  $n - 2$  itens, até que reste apenas um elemento.



# Ordenação por Seleção

# Ordenação por Seleção

- O método é ilustrado abaixo:

	1	2	3	4	5	6
Chaves iniciais:	<i>O</i>	<i>R</i>	<i>D</i>	<i>E</i>	<i>N</i>	<i>A</i>
i = 1	<b>A</b>	<i>R</i>	<i>D</i>	<i>E</i>	<i>N</i>	<b>O</b>
i = 2	<i>A</i>	<b>D</b>	<b>R</b>	<i>E</i>	<i>N</i>	<i>O</i>
i = 3	<i>A</i>	<i>D</i>	<b>E</b>	<b>R</b>	<i>N</i>	<i>O</i>
i = 4	<i>A</i>	<i>D</i>	<i>E</i>	<b>N</b>	<b>R</b>	<i>O</i>
i = 5	<i>A</i>	<i>D</i>	<i>E</i>	<i>N</i>	<b>O</b>	<b>R</b>

# Ordenação por Seleção

- O método é ilustrado abaixo:

	1	2	3	4	5	6
Chaves iniciais:	<i>O</i>	<i>R</i>	<i>D</i>	<i>E</i>	<i>N</i>	<i>A</i>
i = 1	<b>A</b>	<i>R</i>	<i>D</i>	<i>E</i>	<i>N</i>	<b>O</b>
i = 2	<i>A</i>	<b>D</b>	<b>R</b>	<i>E</i>	<i>N</i>	<i>O</i>
i = 3	<i>A</i>	<i>D</i>	<b>E</b>	<b>R</b>	<i>N</i>	<i>O</i>
i = 4	<i>A</i>	<i>D</i>	<i>E</i>	<b>N</b>	<b>R</b>	<i>O</i>
i = 5	<i>A</i>	<i>D</i>	<i>E</i>	<i>N</i>	<b>O</b>	<b>R</b>

- As chaves em negrito sofreram uma troca entre si.

# Ordenação por Seleção

```
1 void Selecao(Tipoltem *A, Tipolndice n){
2     Tipolndice i, j, Min;
3     Tipoltem x;
4     for (i = 0; i < n - 1; i++){
5         Min = i;
6         for (j = i + 1; j < n; j++){
7             if (A[j].Chave < A[Min].Chave){
8                 Min = j;
9             }
10        }
11        x = A[Min];
12        A[Min] = A[i];
13        A[i] = x;
14    }
15 }
```

# Ordenação por Seleção

# Ordenação por Seleção

- $C(n) = \frac{n^2}{2} - \frac{n}{2}$

# Ordenação por Seleção

- $C(n) = \frac{n^2}{2} - \frac{n}{2}$
- $M(n) = 3(n - 1)$

# Ordenação por Seleção

- $C(n) = \frac{n^2}{2} - \frac{n}{2}$
- $M(n) = 3(n - 1)$
- A atribuição  $Min = j$  da linha 8 é executada em média  $n \log n$  vezes, Knuth (1973)



# Ordenação por Seleção

- $C(n) = \frac{n^2}{2} - \frac{n}{2}$
- $M(n) = 3(n - 1)$
- A atribuição  $Min = j$  da linha 8 é executada em média  $n \log n$  vezes, Knuth (1973)
- A complexidade do algoritmo de ordenação por seleção é portanto  $O(n^2)$

# Ordenação por Seleção

Vantagens:

# Ordenação por Seleção

Vantagens:

- Custo linear para o número de movimentos de registros.

# Ordenação por Seleção

## Vantagens:

- Custo linear para o número de movimentos de registros.
- É o algoritmo a ser utilizado para arquivos com registros muito grandes.

# Ordenação por Seleção

## Vantagens:

- Custo linear para o número de movimentos de registros.
- É o algoritmo a ser utilizado para arquivos com registros muito grandes.
- É muito interessante para arquivos pequenos.

## Desvantagens:

# Ordenação por Seleção

## Vantagens:

- Custo linear para o número de movimentos de registros.
- É o algoritmo a ser utilizado para arquivos com registros muito grandes.
- É muito interessante para arquivos pequenos.

## Desvantagens:

- O fato de o arquivo já estar ordenado não ajuda em nada, pois o custo continua quadrático.

# Ordenação por Seleção

## Vantagens:

- Custo linear para o número de movimentos de registros.
- É o algoritmo a ser utilizado para arquivos com registros muito grandes.
- É muito interessante para arquivos pequenos.

## Desvantagens:

- O fato de o arquivo já estar ordenado não ajuda em nada, pois o custo continua quadrático.
- O algoritmo não é **estável**.