

# Algoritmos de Ordenação (Parte 1)

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados I (AE22CP)  
Engenharia de Computação  
Departamento Acadêmico de Informática (Dainf)  
Universidade Tecnológica Federal do Paraná (UTFPR)  
Campus Pato Branco

- Método da bolha (*bubble sort*)
- Ordenação por seleção (*selection sort*)
- Ordenação por inserção (*insertion sort*)

- Ordenação (ou classificação): Tornar mais simples, rápida e viável a recuperação de uma determinada informação, em um conjunto grande de informações
- Terminologia básica:
  - Arquivo de tamanho  $n$  é uma sequência de  $n$  itens  $(X_1, X_2, \dots, X_n)$
  - O  $i$ -ésimo componente do arquivo é chamado de item
  - Uma chave é associada a cada registro
  - Ordenação pela chave

# Introdução

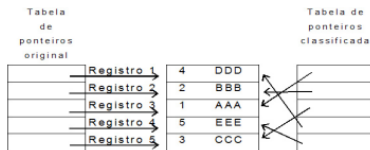
- Terminologia básica:
  - Ordenação interna
  - Ordenação externa
  - Ordenação estável
  - Ordenação pode ocorrer sobre os próprios registros ou sobre uma tabela auxiliar de ponteiros

Registro 1	4	DDD
Registro 2	2	BBB
Registro 3	1	AAA
Registro 4	5	EEE
Registro 5	3	CCC

Arquivo Original

1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Arquivo classificado



- Devido à relação entre a ordenação e a busca, surge uma pergunta: o arquivo deve ser classificado ou não?
- Eficiência de métodos de ordenação
  - Tempo para a execução do método
  - Espaço de memória necessário
- Normalmente o tempo gasto é medido pelo número de operações críticas
  - Comparação de chaves
  - Movimento de registros

- O resultado da mensuração do desempenho é uma fórmula em função de  $n$
- Existem vários tipos de métodos de ordenação (e.g., troca, seleção, inserção)
- Não existe um método de ordenação considerado superior aos outros

- Exemplos de algoritmos de ordenação:
  - Método da bolha (*bubble sort*)
  - Ordenação por seleção (*selection sort*)
  - Ordenação por inserção (*insertion sort*)
  - *Quicksort*
  - *Heapsort*
  - *Shell sort*
  - Ordenação por intercalação (*mergesort*)
  - Ordenação por contagem (*counting sort*)

## Método da bolha (*bubble sort*)



# Método da bolha (*bubble sort*)

- Ordenação por troca: Em cada comparação pode haver troca de posições entre itens comparados
- Percorre o arquivo sequencialmente várias vezes, na qual cada elemento é comparado com o seu sucessor
- Fácil compreensão e implementação
- Um dos métodos de ordenação menos eficiente
- O método é estável

# Método da bolha (*bubble sort*)

- Implementação

```
void bubblesort(int v[], int n){  
    int i, j, x;  
  
    for (i = 0; i < n - 1; i++)  
        for (j = 0; j < n - i - 1; j++)  
            if (v[j] > v[j + 1]){  
                x = v[j];  
                v[j] = v[j + 1];  
                v[j + 1] = x;  
            }  
}
```

# Método da bolha (*bubble sort*)

- Implementação (com uma melhoria)

```
void bubblesort(int v[], int n){
    int i, j, x, troca = 1;

    for (i = 0; (i < n - 1) && troca; i++){
        troca = 0;

        for (j = 0; j < n - i - 1; j++){
            if (v[j] > v[j + 1]){
                x = v[j];
                v[j] = v[j + 1];
                v[j + 1] = x;
                troca = 1;
            }
        }
    }
}
```

# Método da bolha (*bubble sort*)

- Conjunto completo de iterações  $i$  (em verde, estão os elementos que foram ordenados em cada iteração)

iteração	x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
–	25	57	48	37	12	92	86	33
0	25	48	37	12	57	86	33	92
1	25	37	12	48	57	33	86	92
2	25	12	37	48	33	57	86	92
3	12	25	37	33	48	57	86	92
4	12	25	33	37	48	57	86	92
5	12	25	33	37	48	57	86	92

- Na quinta iteração, os elementos estão completamente ordenados

# Método da bolha (*bubble sort*)

- Eficiência do *bubble sort* com melhorias
  - O número de trocas não é maior que o número de comparações
  - Vantajoso em conjuntos pré-ordenados
  - Redução do número de comparações em cada passagem:
    - 1a. passagem:  $n - 1$  comparações
    - 2a. passagem:  $n - 2$  comparações
    - ...
    - $(n - 1)$ a. passagem: 1 comparação
  - Total:  $\frac{(n^2+n)}{2}$
  - $O(n^2)$

# Método da bolha (*bubble sort*)

- Links interessantes:
  - Dança húngara:  
<https://www.youtube.com/watch?v=lyZQPjUT5B4>
  - Simulador gráfico do *bubble sort*:  
<https://visualgo.net/bn/sorting>

## Ordenação por seleção (*selection sort*)

# Ordenação por seleção (*selection sort*)

- Ordenação por seleção: escolhe o maior ou o menor elemento do conjunto para cada iteração para colocá-lo em sua devida posição
- Ideia básica
  - 1 Selecionar o maior elemento do conjunto
  - 2 Trocá-lo com o último elemento
  - 3 Repetir os dois passos anteriores com os  $n - 1$  elementos restantes, após com os  $n - 2$  e assim por diante até sobrar o primeiro elemento que será o menor do conjunto



# Ordenação por seleção (*selection sort*)

- Implementação

```
void selectsort(int v[], int n){
    int i, j, p, aux;

    for (i = n - 1; i >= 1; i--){
        p = i;

        for (j = 0; j < i; j++)
            if (v[j] > v[p])
                p = j;

        aux = v[i];
        v[i] = v[p];
        v[p] = aux;
    }
}
```

# Ordenação por seleção (*selection sort*)

- Exemplo

i	p	X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]
-	-	25	57	48	37	12	92	86	33
7	5	25	57	48	37	12	33	86	92
6	6	25	57	48	37	12	33	86	92
5	1	25	33	48	37	12	57	86	92
4	2	25	33	12	37	48	57	86	92
3	3	25	33	12	37	48	57	86	92
2	1	25	12	33	37	48	57	86	92
1	0	12	25	33	37	48	57	86	92

# Ordenação por seleção (*selection sort*)

- Complexidade do método:  $O(n^2)$
- Esse método de ordenação é estável
- Simples implementação
- Um dos algoritmos mais rápidos para a ordenação de vetores pequenos
- Em vetores grandes, esse algoritmo é um dos mais lentos

# Ordenação por seleção (*selection sort*)

- Links interessantes:
  - Dança cigana:  
<https://www.youtube.com/watch?v=Ns4TPTC8whw>
  - Simulador gráfico do *select sort*:  
<https://visualgo.net/bn/sorting>

Ordenação por inserção (*insertion sort*)

# Ordenação por inserção (*insertion sort*)

- Ordenação por inserção: classifica um conjunto de registros inserindo registros em um arquivo classificado existente
- É o método que consiste em inserir informações num conjunto já ordenado
- Algoritmo utilizado pelo jogador de cartas
  - As cartas são mantidas ordenadas nas mãos dos jogadores
  - Quando o jogador compra ou recebe uma nova carta, ele procura a posição que a nova carta deverá ocupar
- Bastante eficiente para pequenas entradas

# Ordenação por inserção (*insertion sort*)

- Implementação

```
void insertsort(int v[], int n){  
    int i, x;  
  
    for (i = 1; i < n; i++){  
        x = v[i];  
  
        for (j = i - 1; (j >= 0) && (x < v[j]); j--)  
            v[j + 1] = v[j];  
  
        v[j + 1] = x;  
    }  
}
```

# Ordenação por inserção (*insertion sort*)

- Exemplo

x	i	j	v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	v[6]	v[7]
-	-	-	25	57	48	37	12	92	86	33
57	1	0	25	57	48	37	12	92	86	33
48	2	1	25	57	57	37	12	92	86	33
48	2	0	25	48	57	37	12	92	86	33
37	3	2	25	48	57	57	12	92	86	33
37	3	1	25	48	48	57	12	92	86	33
37	3	0	25	37	48	57	12	92	86	33
12	4	3	25	37	48	57	57	92	86	33
12	4	2	25	37	48	48	57	92	86	33
12	4	1	25	37	37	48	57	92	86	33
12	4	0	25	25	37	48	57	92	86	33
12	4	-1	12	25	37	48	57	92	86	33
92	5	4	12	25	37	48	57	92	86	33



# Ordenação por inserção (*insertion sort*)

- Exemplo

x	i	j	v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	v[6]	v[7]
86	6	5	12	25	37	48	57	92	92	33
86	6	4	12	25	37	48	57	86	92	33
33	7	6	12	25	37	48	57	86	92	92
33	7	5	12	25	37	48	57	86	86	92
33	7	4	12	25	37	48	57	57	86	92
33	7	3	12	25	37	48	48	57	86	92
33	7	2	12	25	37	37	48	57	86	92
33	7	1	12	25	33	37	48	57	86	92

# Ordenação por inserção (*insertion sort*)

- Desempenho do método
  - Melhor caso:  $O(n)$
  - Médio caso:  $O(n^2)$
  - Pior caso:  $O(n^2)$
- Deve ser utilizado quando o arquivo está "quase" ordenado
- Desvantagem: alto custo de movimentação de elementos

# Ordenação por inserção (*insertion sort*)

- Links interessantes:
  - Dança romana:  
<https://www.youtube.com/watch?v=R0a1U37913U>
  - Simulador gráfico do *insertion sort*:  
<https://visualgo.net/bn/sorting>



Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.  
*Introduction to Algorithms.*  
Third edition, The MIT Press, 2009.



Horowitz, E., Sahni, S. Rajasekaran, S.  
*Computer Algorithms.*  
Computer Science Press, 1998.



Rosa, J. L. G.  
Métodos de Ordenação. SCE-181 – Introdução à Ciência da  
Computação II.  
*Slides.* Ciência de Computação. ICMC/USP, 2018.



Ziviani, N.  
*Projeto de Algoritmos - com implementações em Java e C++.*  
Thomson, 2007.