

# Selection Sort e Insertion Sort

Ciência da Computação  
Laboratório de Ordenação e Pesquisa  
Prof. M.Sc. Elias Gonçalves

# Selection Sort

# Ideia

- Percorrer a lista de dados (vetor) e executar os seguintes passos:
  - Encontrar a posição do menor (ou maior) elemento e trocar o elemento com o da primeira posição.
  - Encontrar a posição do segundo menor (ou maior) elemento e trocar o elemento com o da segunda posição...
  - Repetir os passos até que o vetor esteja ordenado.

# Dinâmica

→ Ordenar crescente o vetor:

23	17	10	19	5
0	1	2	3	4

→ Na primeira iteração ( $i=0$ ) procura-se o menor valor entre os índices 0 e 4. O menor valor é encontrado no índice 4. Trocam-se os valores dos índices 4 e 0.

$23 < 17$  menor = 17

$17 < 10$  menor = 10

$10 < 19$  menor = 10

$10 < 5$  menor = 5

Troca 5 (índice 4) com 23 (índice 0)

5	17	10	19	23
0	1	2	3	4

# Dinâmica

→ Ordenar crescente o vetor:

5	17	10	19	23
0	1	2	3	4

→ Na segunda iteração ( $i=1$ ) procura-se o menor valor entre os índices 1 e 4. O menor valor é encontrado no índice 2. Trocam-se os valores dos índices 2 e 1.

$17 < 10$  menor = 10

$10 < 19$  menor = 10

$10 < 23$  menor = 10

Troca 10 (índice 2) com 17 (índice 1)

5	10	17	19	23
0	1	2	3	4

# Dinâmica

→ Ordenar crescente o vetor:

5	10	17	19	23
0	1	2	3	4

→ Na terceira iteração ( $i=2$ ) procura-se o menor valor entre os índices 2 e 4. O menor valor é encontrado no índice 2. Mantém o valor na posição, pois já é o menor.

$17 < 19$  menor = 17

$17 < 23$  menor = 17

Mantém o 17 (índice 2) no mesmo lugar

5	10	17	19	23
0	1	2	3	4

# Dinâmica

→ Ordenar crescente o vetor:

5	10	17	19	23
0	1	2	3	4

→ Na quarta iteração ( $i=3$ ) procura-se o menor valor entre os índices 3 e 4. O menor valor é encontrado no índice 3. Mantém o valor na posição, pois já é o menor.

$19 < 23$  menor = 19

Mantém o 19 (índice 3) no mesmo lugar

5	10	17	19	23
0	1	2	3	4

# Dinâmica

→ Ordenar crescente o vetor:

5	10	17	19	23
0	1	2	3	4

→ Na quinta iteração ( $i=4$ ) o vetor já está ordenado, pois é a última iteração. O menor valor é encontrado no índice 4. Mantém o valor na posição, pois já é o menor.

menor = 23

Mantém o 23 (índice 4) no mesmo lugar

5	10	17	19	23
0	1	2	3	4



# Código

```
void selection_sort(int v[], int tam){
    int i;
    for(i=0; i<tam; i++){
        int min = minimo_indice(v, i, tam-1);
        troca_elemento(v, i, min);
    }
}
```

```
void troca_elemento(int v[], int i, int min){
    int aux = v[i];
    v[i] = v[min];
    v[min] = aux;
}
```

```
int minimo_indice(int v[], int i, int tam){
    int min = i;
    int aux;

    for(aux = i+1; aux <= tam; aux++){
        if(v[min] > v[aux])
            min = aux;
    }

    return min;
}
```

```
void imprimir_vetor(int v[], int tam) {
    int i;
    for (i=0; i<tam; i++)
        printf("%d ", v[i]);
    printf("\n");
}
```

# Insertion Sort

# Ideia

- Divide a lista de dados de entrada em duas partições, uma ordenada (esquerda) e outra desordenada (direita). Inicialmente a lista ordenada só terá 1 elemento (índice 0).
- Inserindo elemento na partição ordenada:
  - Procurar a posição em que será inserido;
  - Deslocar o elemento da partição esquerda para a direita deixando a posição de inserção livre para inserir o elemento ordenado.

# Dinâmica

→ Ordenar crescente o vetor:

**$i = 1$ ;  $aux = 0$ ; elemento = 17**

- O que se deseja é inserir o elemento na partição ordenada (esquerda);
- Como  $23 > 17$ , desloca-se o 23 para a direita;
- Insere o 17 no índice 0 e mantém a partição ordenada.

23	17	10	19	5
0	1	2	3	4

23	17	10	19	5
0	1	2	3	4



23	23	10	19	5
0	1	2	3	4

17	23	10	19	5
0	1	2	3	4

# Dinâmica

→ Ordenar crescente o vetor:

**$i = 2$ ;  $aux = 1$ ; elemento = 10**

- O que se deseja é inserir o elemento na partição ordenada (esquerda);
- Como  $23 > 10$ , desloca-se o 23 para a direita;
- Insere o 10 no índice 1;
- Como  $17 > 10$ , desloca-se o 17 para a direita;
- Insere o 10 no índice 0 e mantém a partição ordenada.

17	23	10	19	5
0	1	2	3	4

17	23	10	19	5
0	1	2	3	4



17	23	23	19	5
0	1	2	3	4

17	10	23	19	5
0	1	2	3	4

17	17	23	19	5
0	1	2	3	4

10	17	23	19	5
0	1	2	3	4

# Dinâmica

→ Ordenar crescente o vetor:

**$i = 3$ ;  $aux = 2$ ; elemento = 19**

- O que se deseja é inserir o elemento na partição ordenada (esquerda);
- Como  $23 > 19$ , desloca-se o 23 para a direita;
- Insere o 19 no índice 2 e mantém a partição ordenada.

10	17	23	19	5
0	1	2	3	4

10	17	23	19	5
0	1	2	3	4



10	17	23	23	5
0	1	2	3	4

10	17	19	23	5
0	1	2	3	4

# Dinâmica


→ Ordenar crescente o vetor:

**$i = 4$ ;  $aux = 3$ ; elemento = 5**

- O que se deseja é inserir o elemento na partição ordenada (esquerda);
- Como  $23 > 5$ , desloca-se o 23 para a direita;
- Insere o 5 no índice 3;
- Como  $19 > 5$ , desloca-se o 19 para a direita;
- Insere o 5 no índice 2;

10	17	19	23	5
0	1	2	3	4

10	17	19	23	5
0	1	2	3	4



10	17	19	23	23
0	1	2	3	4

10	17	19	5	23
0	1	2	3	4

10	17	19	19	23
0	1	2	3	4

10	17	5	19	23
0	1	2	3	4

# Dinâmica

## (Continuação)

- Como  $17 > 5$ , desloca-se o 17 para a direita;
- Insere o 5 no índice 1;
- Como  $10 > 5$ , desloca-se o 10 para a direita;
- Insere o 5 no índice 0 e mantém a partição ordenada.

10	17	5	19	23
0	1	2	3	4

10	17	17	19	23
0	1	2	3	4

10	5	17	19	23
0	1	2	3	4

10	10	17	19	23
0	1	2	3	4

5	10	17	19	23
0	1	2	3	4



# Código

```
void insertion_sort(int v[], int tam){
    int i;
    for(i=1; i<tam; i++){
        int elemento = v[i];
        int aux = i-1;
        while(aux >= 0 && v[aux] > elemento){
            v[aux+1] = v[aux];
            aux = aux-1;
        }
        v[aux+1] = elemento;
    }
}
```

```
void imprimir_vetor(int v[], int tam) {
    int i;
    for (i=0; i<tam; i++)
        printf("%d ", v[i]);
    printf("\n");
}
```

# Atividades

- Utilizando **selection sort** e **Insertion sort**:
  - Preencha um vetor com o nome de 10 pessoas. Crie um programa para ordená-lo e mostrá-lo em ordem crescente e decrescente;
  - Preencha um vetor com salários de 10 pessoas. Ordene-o e mostre-o em ordem crescente e decrescente;

# Biblioteca Virtual

CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas.  
**Introdução a Estruturas de Dados com Técnicas de Programação em C** (Capítulo 11);

DROZDEK, Adam. **Estrutura de dados e algoritmos em c++**  
(Capítulo 9);

MARKENZON, Lilian; SZWARCFITER, Jorge Luiz. **Estruturas de Dados e seus Algoritmos** (Capítulos: 7, 11 e 12);

PINTO, Rafael Albuquerque. **Estrutura de Dados** (Páginas 155 a 177).