



SÃO
PAULO
TECH
SCHOOL



ED

Estrutura de Dados e Armazenamento

Ordenação

(Selection Sort, Bubble Sort)

Profa. Célia Taniwaki

Ordenação

- Muitas vezes, é preciso ordenar os dados para poder manipular esses dados de uma forma organizada.
 - Por exemplo:
 - Nomes em ordem alfabética
 - Alunos por ordem do RA ou por Nota
- A busca de um elemento em uma lista ordenada é mais eficiente (Pesquisa binária)

Métodos de ordenação simples

- Esses 3 são algoritmos simples e intuitivos:
 - Selection sort – ordenação por seleção
 - Bubble sort – ordenação por troca
 - Insertion sort – ordenação por inserção
- Características:
 - Fácil implementação

Selection Sort

- Método simples de seleção
 - Percorre uma vez o vetor, procurando o menor valor . Troca o 1º elemento com o menor valor selecionado.
 - Percorre do 2º elemento em diante para procurar o 2º menor valor. Troca o 2º elemento com o 2º menor valor selecionado.
 - E assim sucessivamente.
- Características
 - Ordena através de sucessivas seleções do elemento de menor valor (ou de maior valor) em um segmento não ordenado do vetor e seu posicionamento no final de um segmento ordenado
 - Realiza uma busca sequencial pelo menor valor (ou maior valor) no segmento não ordenado a cada iteração

Selection Sort - Exemplo

Sejam os dados (**ordenados** / **não ordenados**):

4 7 5 2 8 1 6 3

O algoritmo consiste em 2 fors (um dentro do outro)

Vamos supor que o for de fora use a variável i

O for de dentro usa a variável j e começa valendo $i+1$

Dentro do for de dentro, verifica se $v[j] < v[i]$

Se for, troca $v[j]$ com $v[i]$

Para trocar, é preciso 3 instruções (é necessário uma variável aux)

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

```
v[i] = v[j];
```

```
v[j] = aux;
```

Selection Sort - Exemplo

Sejam os dados (**ordenados** / **não ordenados**):

4 7 5 2 8 1 6 3

i começa valendo zero (é o índice do 4)

j começa valendo $i + 1 = 1$ (é o índice do 7)

Verifica se 7 é menor do que 4 \Rightarrow não é, não faz nada

j é incrementado, agora é o índice do 5

Verifica se 5 é menor do que 4 \Rightarrow não é, não faz nada

j é incrementado, agora é o índice do 2

Verifica se 2 é menor do que 4 \Rightarrow é, troca eles de lugar

2 7 5 4 8 1 6 3

Agora i é o índice do 2, e j é o índice do 4

Selection Sort - Exemplo

Sejam os dados (**ordenados** / **não ordenados**):

2 7 5 4 8 1 6 3

j é incrementado, agora é o índice do 8

Verifica se 8 é menor do que 2 \Rightarrow não é, não faz nada

j é incrementado, agora é o índice do 1

Verifica se 1 é menor do que 2 \Rightarrow é, troca eles de lugar

1 7 5 4 8 2 6 3

Agora i é o índice do 1, e j é o índice do 2

j é incrementado, agora é o índice do 6

Verifica se 6 é menor do que 1 \Rightarrow não é, não faz nada

Selection Sort - Exemplo

Sejam os dados (**ordenados** / **não ordenados**):

1 7 5 4 8 2 6 3

j é incrementado, agora é o índice do 3

Verifica se 3 é menor do que 1 \Rightarrow não é, não faz nada

Acabou o for do j e o menor valor do vetor está posicionado na 1ª posição do vetor:

1 7 5 4 8 2 6 3

Agora o for do i incrementa o i

i é o índice do 7

O for do j faz com que j comece valendo 2, é o índice do 5.

E começa tudo de novo.

Selection Sort - Algoritmo

```
selectionSort (int[] v)
início                                     // início é um {
    inteiro i, j;
    para i de 0 enqto i < v.length-1 faça // essa instrução é um for
início
        para j de i+1 enqto j < v.length faça // essa instrução é outro for
início
            se v[j] < v[i]
                então troca(v[i], v[j]);
        fim
    fim
fim                                         // fim é um }

// onde está troca(v[i],v[j]) significa que os valores
// de v[i] e v[j] devem ser trocados (isso pode ser
// feito com 3 instruções e uma variável auxiliar)
```

Selection Sort

- O Selection Sort, na verdade, não fica trocando os valores de lugar toda vez que “encontra” um valor menor do que $v[i]$
- Ele usa uma variável `indMenor` (índice do menor) para guardar o valor do índice do menor elemento do vetor.
- Essa variável começa valendo i
- E se o for do j encontrar um $v[j]$ menor do que $v[indMenor]$, substitui o `indMenor` pelo j .
- Dessa forma, ao terminar o for do j , `indMenor` é o índice do menor valor do vetor.
- Aí então é feita a troca entre $v[i]$ e $v[indMenor]$

Selection Sort - Exemplo

Sejam os dados (**ordenados** / **não ordenados**):

⇒ menor: 1 – troca com o 1^o

⇒ menor: 2 – troca com o 2^o

⇒ menor: 3 – troca com o 3^o

⇒ menor: 4 – troca com o 4^o

⇒ menor: 5 – troca com o 5^o

⇒ menor: 6 – troca com o 6^o

⇒ menor: 7 – já está ok

4 7 5 2 8 1 6 3

1 7 5 2 8 4 6 3

1 2 5 7 8 4 6 3

1 2 3 7 8 4 6 5

1 2 3 4 8 7 6 5

1 2 3 4 5 7 6 8

1 2 3 4 5 6 7 8

1 2 3 4 5 6 7 8

Selection Sort Otimizado - Algoritmo

```
selectionSortOtimizado (int[] v)
início                                     // início é um {
    inteiro i, j, indMenor;
    para i de 0 enqto i < v.length-1 faça //essa instrução é um for
    início
        indMenor ← i;
        para j de i+1 enqto j < v.length faça //essa instrução é outro for
        início
            se v[j] < v[indMenor]
            então início
                indMenor ← j;
            fim
        fim
    fim
    troca(v[i], v[indMenor]);
fim                                         // fim é um }
```

// onde está troca(v[i],v[indMenor]) significa que os valores
// de v[i] e v[indMenor] devem ser trocados (isso pode ser
// feito com 3 instruções e uma variável auxiliar)

Bubble Sort

- Método simples de troca
 - Compara elementos vizinhos do vetor. Se o anterior for maior do que o próximo, troca-os de lugar
- Características
 - Realiza varreduras no vetor, trocando pares adjacentes (vizinhos) de elementos, sempre que o próximo elemento for menor que o anterior
 - Após uma varredura, o maior elemento está corretamente posicionado no vetor e não precisa mais ser comparado. Ordena através de sucessivas trocas entre pares de elementos do vetor

Bubble Sort - Exemplo

Sejam os dados (**desordenados** / **ordenados**):

⇒ $4 > 7$? – não, não troca

⇒ $7 > 5$? – sim, troca

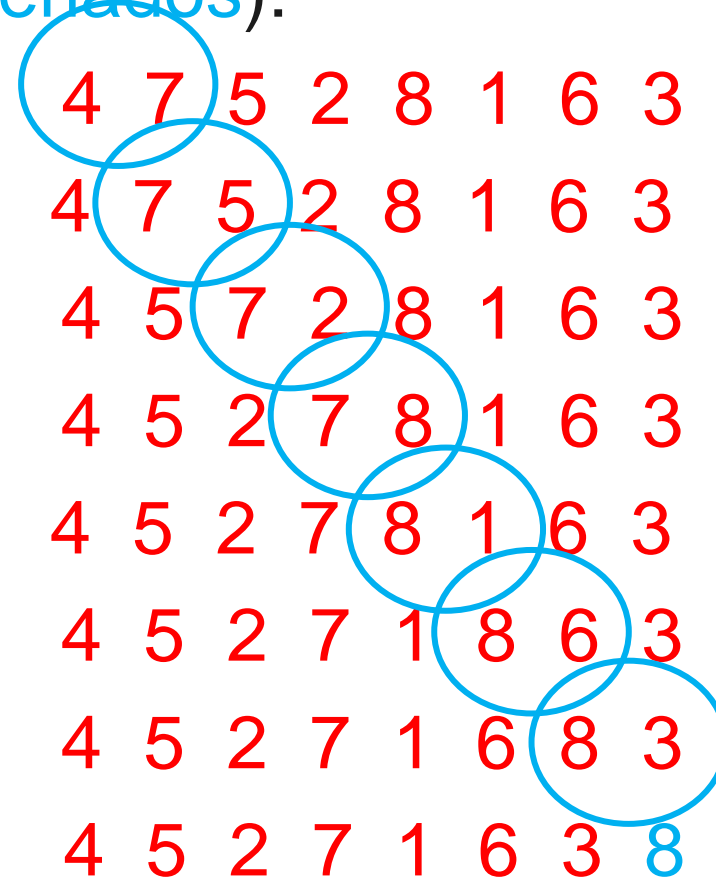
⇒ $7 > 2$? – sim, troca

⇒ $7 > 8$? – não, não troca

⇒ $8 > 1$? – sim, troca

⇒ $8 > 6$? – sim, troca

⇒ $8 > 3$? – sim, troca (8 fica certo)



O maior valor “sobe” como se fosse uma bolha (bubble)

Bubble Sort - Exemplo

Continuando (**desordenados** / **ordenados**):

⇒ $4 > 5$? – não, não troca

⇒ $5 > 2$? – sim, troca

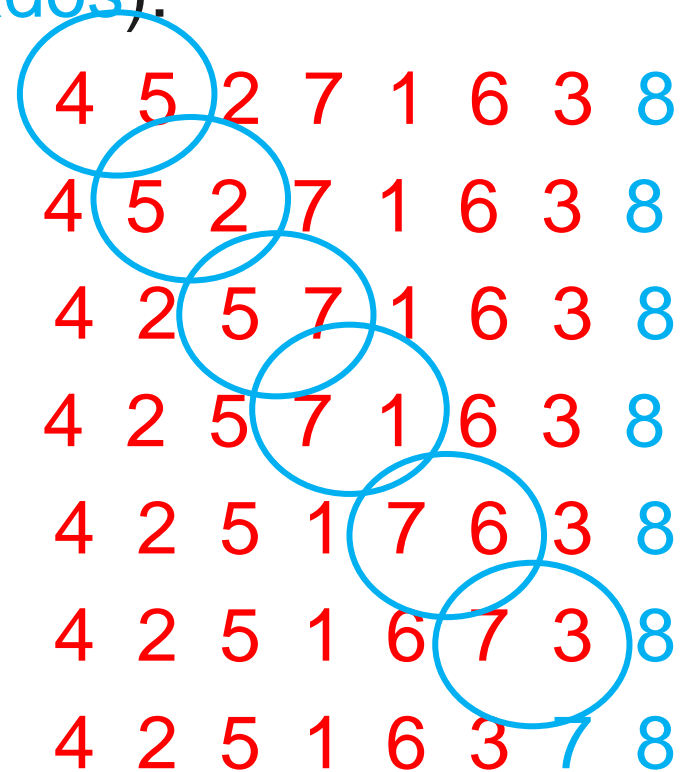
⇒ $5 > 7$? – não, não troca

⇒ $7 > 1$? – sim, troca

⇒ $7 > 6$? – sim, troca

⇒ $7 > 3$? – sim, troca

7 fica no lugar correto ... E assim sucessivamente...



Bubble Sort - Algoritmo

```
bubbleSort (int[] v)
```

```
início
```

```
    inteiro i, j;
```

```
    para i de 0 enquanto i < v.length-1 faça
```

```
        início
```

```
            para j de 1 enquanto j < v.length-i faça
```

```
                início
```

```
                    se v[j-1] > v[j]    // se anterior é menor que atual
```

```
                        então troca (v[j], v[j-1]);
```

```
                fim
```

```
            fim
```

```
    fim
```

Agradeço
a sua atenção!

Célia Taniwaki

celia.taniwaki@sptech.school

SÃO
PAULO
TECH
SCHOOL