

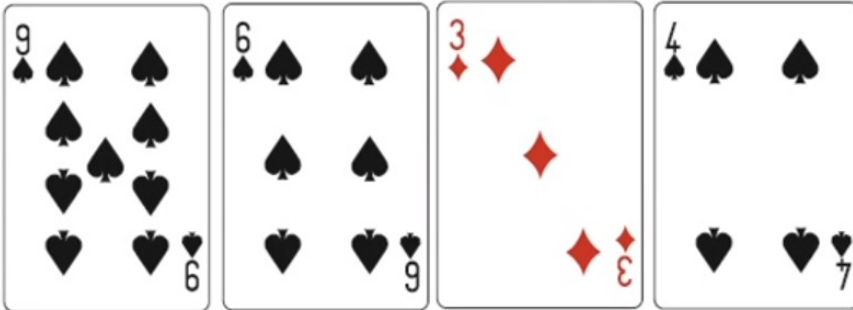
DCC405 – ESTRUTURA DE DADOS II

Aula 15.1 – Selection Sort

Selection Sort

- A ordenação por seleção (*selection sort*) é um algoritmo de ordenação baseado em se passar sempre o menor valor do vetor para a primeira posição (ou o maior dependendo da ordem requerida), depois o de segundo menor valor para a segunda posição, e assim é feito sucessivamente com os ***n-1*** elementos restantes, até os últimos dois elementos.

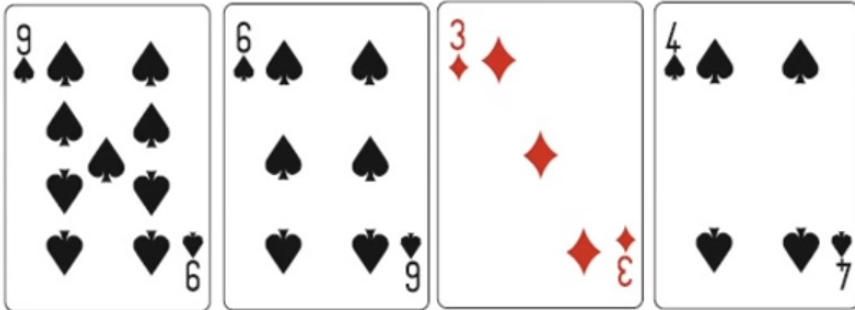
Selection Sort - Exemplo



Selection Sort - Exemplo

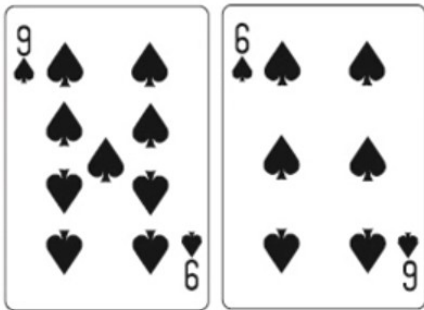
Lado Esquerdo

Lado Direito

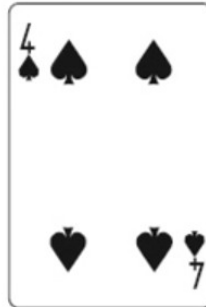


Selection Sort - Exemplo

Lado Esquerdo

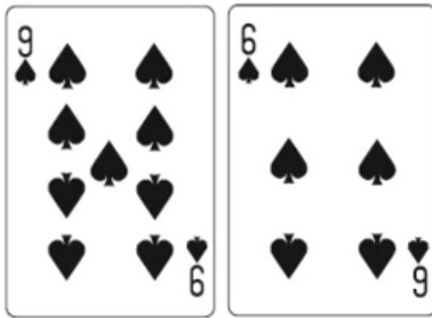


Lado Direito

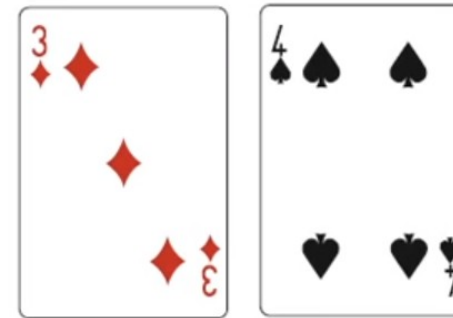


Selection Sort - Exemplo

Lado Esquerdo



Lado Direito

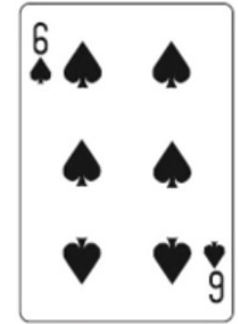
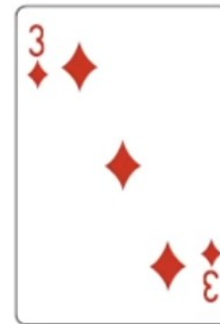


Selection Sort - Exemplo

Lado Esquerdo



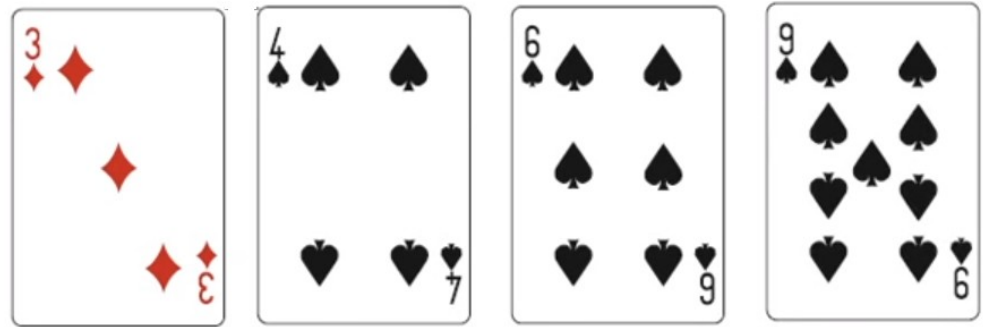
Lado Direito



Selection Sort - Exemplo

Lado Esquerdo
desordenado

Lado Direito
ordenado



Selection Sort - 1ª abordagem

2	7	4	1	5	3
0	1	2	3	4	5

Selection Sort - 1ª abordagem

A

2	7	4	1	5	3
0	1	2	3	4	5

Selection Sort - 1ª abordagem

A

2	7	4	1	5	3
0	1	2	3	4	5

B

0	1	2	3	4	5

Selection Sort - 1ª abordagem

A

2	7	4	1	5	3
0	1	2	3	4	5

B

1					
0	1	2	3	4	5

Selection Sort - 1ª abordagem

A

2	7	4	MAX	5	3
0	1	2	3	4	5

B

1					
0	1	2	3	4	5

$$\text{MAX} = 2^{31} - 1$$

Selection Sort - 1ª abordagem

A

MAX	7	4	MAX	5	3
0	1	2	3	4	5

B

1	2				
0	1	2	3	4	5

$$\text{MAX} = 2^{31} - 1$$

Selection Sort - 1ª abordagem

A

MAX	7	4	MAX	5	MAX
0	1	2	3	4	5

B

1	2	3			
0	1	2	3	4	5

$$\text{MAX} = 2^{31} - 1$$

Selection Sort - 1ª abordagem

A

MAX	MAX	MAX	MAX	MAX	MAX
2	7	4	1	5	3
0	1	2	3	4	5

B

1	2	3	4	5	7
0	1	2	3	4	5

$$\text{MAX} = 2^{31} - 1$$

Selection Sort - 1ª abordagem

A

MAX	MAX	MAX	MAX	MAX	MAX
0	1	2	3	4	5

$$\text{MAX} = 2^{31} - 1$$

B

1	2	3	4	5	7
0	1	2	3	4	5

→ **In-place sorting algorithm** pega um número constante extra de memória

Selection Sort - 2ª abordagem

A

2	7	4	1	5	3
0	1	2	3	4	5

Selection Sort - 2ª abordagem

A

2	7	4	1	5	3
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 0

Selection Sort - 2ª abordagem

A

2	7	4	1	5	3
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 0

Selection Sort - 2ª abordagem

A

1	7	4	2	5	3
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 0

Selection Sort - 2ª abordagem

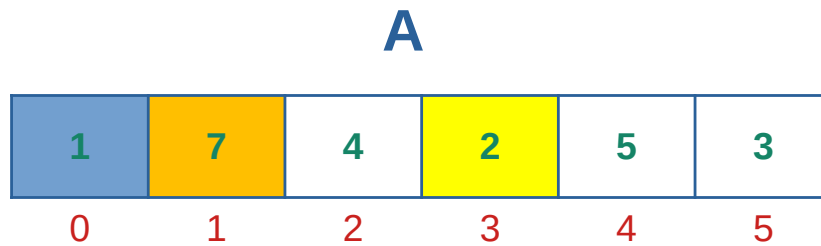
A

1	7	4	2	5	3
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 1

Selection Sort - 2ª abordagem



Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 1

Selection Sort - 2ª abordagem

A

1	2	4	7	5	3
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 1

Selection Sort - 2ª abordagem

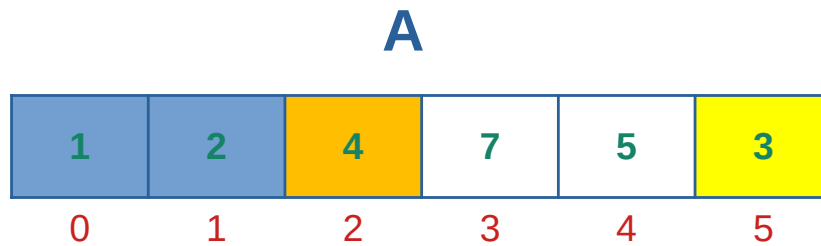
A

1	2	4	7	5	3
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 2

Selection Sort - 2ª abordagem



Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 2

Selection Sort - 2ª abordagem

A

1	2	3	7	5	4
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 2

Selection Sort - 2ª abordagem

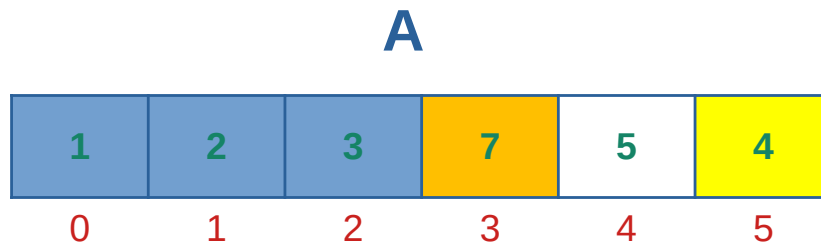
A

1	2	3	7	5	4
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 3

Selection Sort - 2ª abordagem



Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 3

Selection Sort - 2ª abordagem

A

1	2	3	4	5	7
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 3

Selection Sort - 2ª abordagem

A

1	2	3	4	5	7
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 4

Selection Sort - 2ª abordagem

A

1	2	3	4	5	7
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 4

Selection Sort - 2ª abordagem

A

1	2	3	4	5	7
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 4

Selection Sort - 2ª abordagem

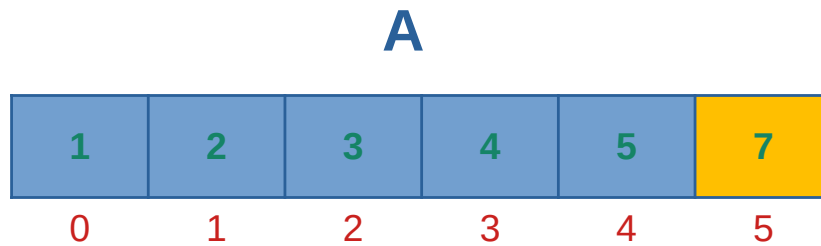
A

1	2	3	4	5	7
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 5

Selection Sort - 2ª abordagem



Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 5

Selection Sort - 2ª abordagem

A

1	2	3	4	5	7
0	1	2	3	4	5

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 5

Selection Sort - 2ª abordagem

A

1	2	3	4	5	7
0	1	2	3	4	5

Ordenado!

Algoritmo:

- 1) Encontra o elemento mínimo
- 2) Troca ele com o elemento do índice 5

Selection Sort - 2ª abordagem

A

2	7	4	1	5	3
0	1	2	3	4	5

Pseudocódigo:

```
selectionSort(Array, n) {  
    for i ← 0 to n-2 {  
        iMin ← i  
        for j ← i+1 to n-1 {  
            if(Array[j] < Array[iMin])  
                iMin ← j  
        }  
        temp ← Array[i]  
        Array[i] ← Array[iMin]  
        Array[iMin] ← temp  
    }  
}
```

Selection Sort - Complexidade

A

1	2	3	4	5	7
0	1	2	3	4	5

Custo computacional: $O(n^2)$

Pseudocódigo:

```
selectionSort(Array, n) {  
    for i ← 0 to n-2 {  
        iMin ← i c1  
        for j ← i+1 to n-1 {  
            c2 | if(Array[j] < Array[iMin])  
                iMin ← j  
        }  
        temp ← Array[i]  
        Array[i] ← Array[iMin] c3  
        Array[iMin] ← temp  
    }  
}
```

Selection Sort - Complexidade

A

1	2	3	4	5	7
0	1	2	3	4	5

Custo computacional: $O(n^2)$

Pseudocódigo:

```
selectionSort(Array, n) {  
    for i ← 0 to n-2 {  
        iMin ← i c1 * n-1 vezes  
        for j ← i+1 to n-1 {  
            c2 | if(Array[j] < Array[iMin])  
                iMin ← j  
        }  
        temp ← Array[i]  
        Array[i] ← Array[iMin] c3 * n-1 vezes  
        Array[iMin] ← temp  
    }  
}
```


Selection Sort - Complexidade

A

1	2	3	4	5	7
0	1	2	3	4	5

Custo computacional: $O(n^2)$

$$(n-1) + (n-2) + (n-3) + \dots + 1 \\ = \\ n(n-1)/2$$

Pseudocódigo:

```
selectionSort(Array, n) {  
    for i ← 0 to n-2 {  
        iMin ← i c1 * n-1 vezes  
        for j ← i+1 to n-1 {  
            c2 | if(Array[j] < Array[iMin])  
                iMin ← j  
        }  
        temp ← Array[i]  
        Array[i] ← Array[iMin] c3 * n-1 vezes  
        Array[iMin] ← temp  
    }  
}
```

Selection Sort - Complexidade

A

1	2	3	4	5	7
0	1	2	3	4	5

Custo computacional: $O(n^2)$

Pseudocódigo:

```
selectionSort(Array, n) {  
    for i ← 0 to n-2 {  
        iMin ← i c1 * n-1 vezes  
        for j ← i+1 to n-1 {  
            c2 | if(Array[j] < Array[iMin])  
                iMin ← j  
        }  
        temp ← Array[i]  
        Array[i] ← Array[iMin] c3 * n-1 vezes  
        Array[iMin] ← temp  
    }  
}
```

$$(n-1) + (n-2) + (n-3) + \dots + 1 \\ = \\ n(n-1)/2$$

$$T(n) = (n-1)*c1 + n(n-1)/2 * c2 \\ + (n-1) * c3$$

Selection Sort - Complexidade

A

1	2	3	4	5	7
0	1	2	3	4	5

Custo computacional: $O(n^2)$

$$(n-1) + (n-2) + (n-3) + \dots + 1 \\ = \\ n(n-1)/2$$



$$T(n) = (n-1)*c_1 + n(n-1)/2 * c_2 \\ + (n-1) * c_3$$



$$T(n) = an^2 + bn + c$$



$$O(n^2)$$

Pseudocódigo:

```
selectionSort(Array, n) {  
    for i ← 0 to n-2 {  
        iMin ← i c1 * n-1 vezes  
        for j ← i+1 to n-1 {  
            c2 | if(Array[j] < Array[iMin])  
                iMin ← j  
        }  
        temp ← Array[i]  
        Array[i] ← Array[iMin] c3 * n-1 vezes  
        Array[iMin] ← temp  
    }  
}
```

Selection Sort - Complexidade

