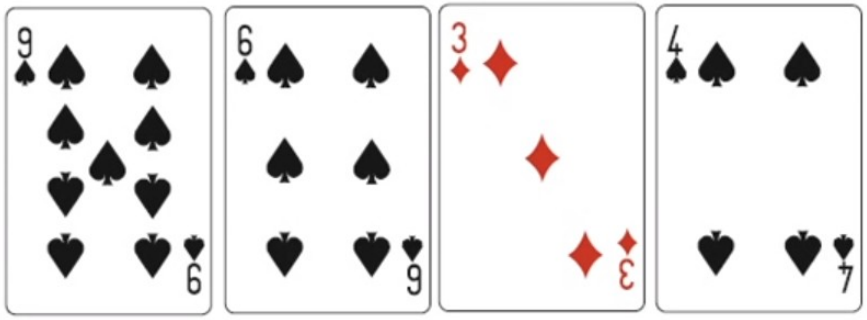
DCC405 – ESTRUTURA DE DADOS II

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.

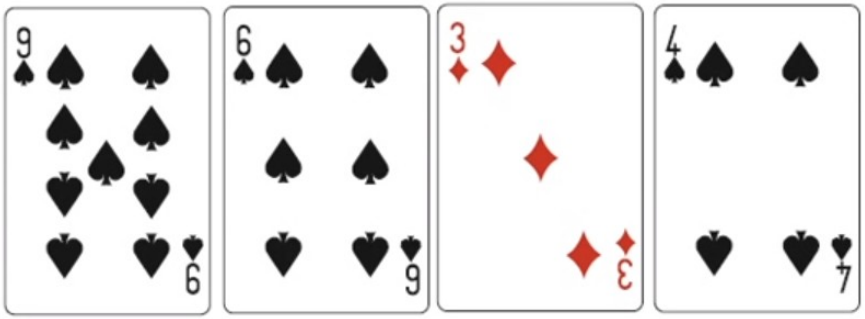
Selection Sort - Exemplo



Selection Sort - Exemplo

Lado Esquerdo

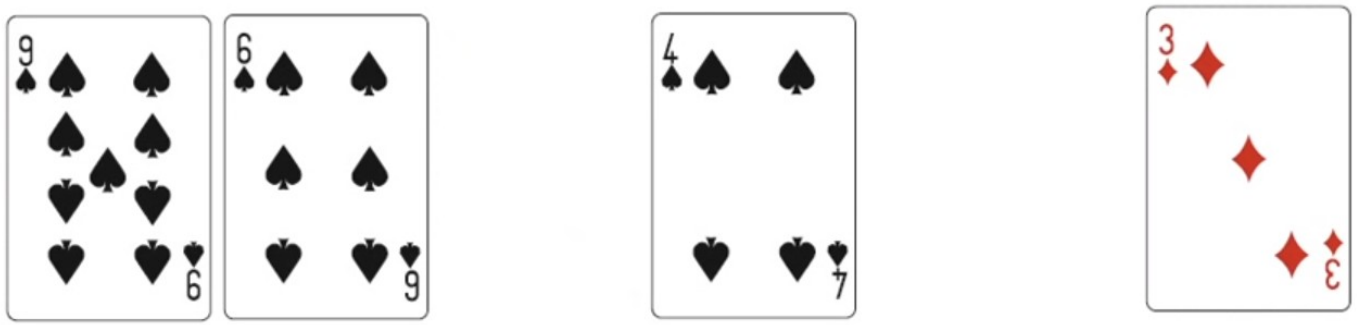
Lado Direito



Selection Sort - Exemplo

Lado Esquerdo

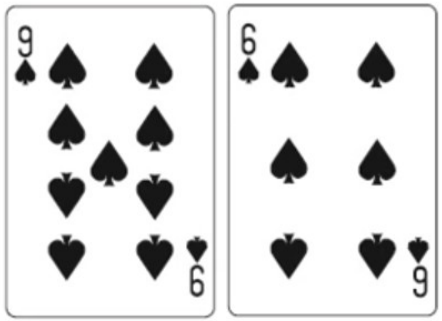
Lado Direito

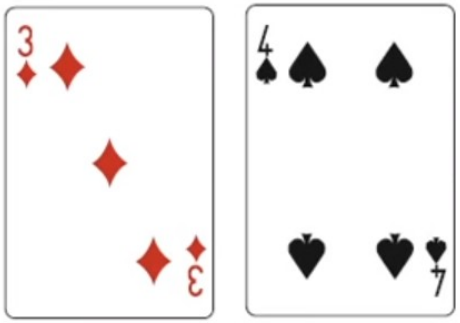


Selection Sort - Exemplo

Lado Esquerdo

Lado Direito

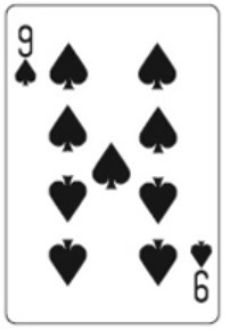


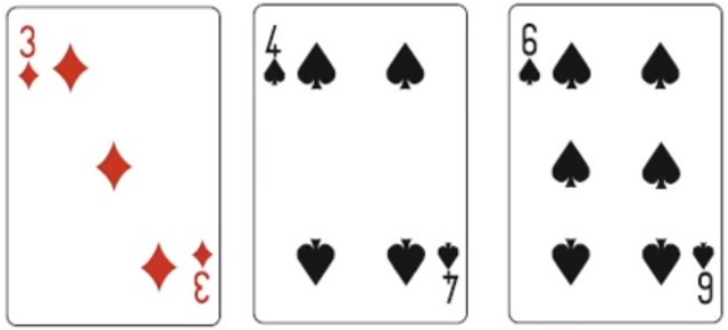


Selection Sort - Exemplo

Lado Esquerdo

Lado Direito

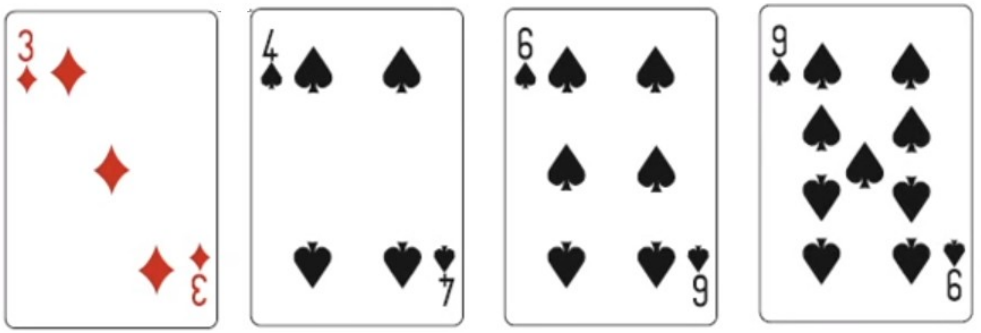




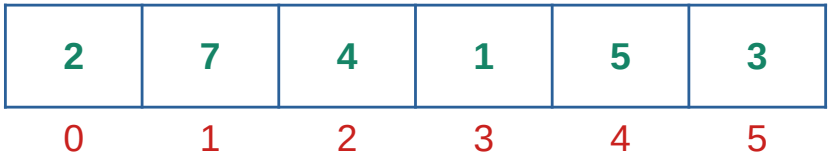
Selection Sort - Exemplo

Lado Esquerdo
desordenado

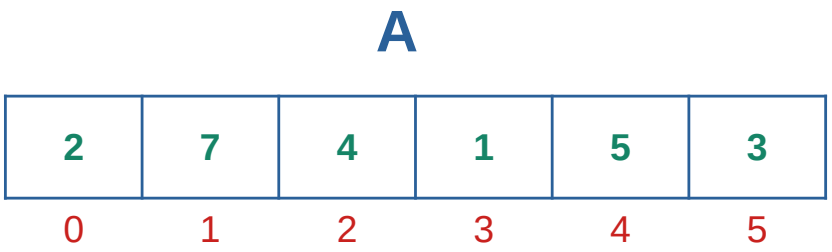
Lado Direito
ordenado



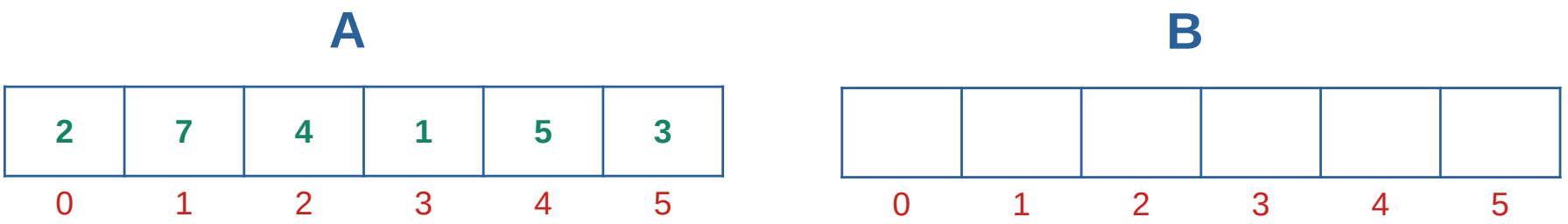
Selection Sort - 1ª abordagem



Selection Sort - 1ª abordagem



Selection Sort - 1ª abordagem



Selection Sort - 1ª abordagem

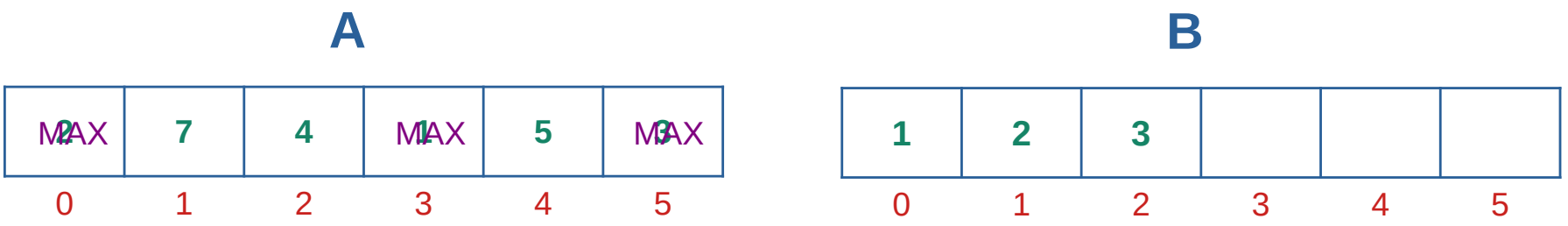
Selection Sort - 1ª abordagem

$MAX = 2^{31} - 1$

Selection Sort - 1ª abordagem

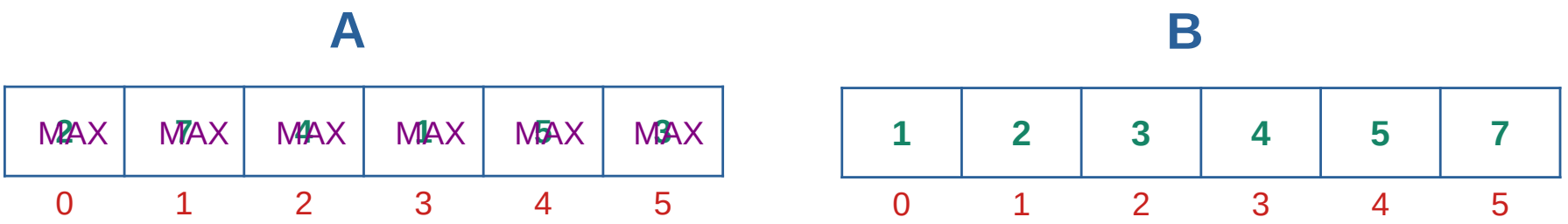
$MAX = 2^{31} - 1$

Selection Sort - 1ª abordagem



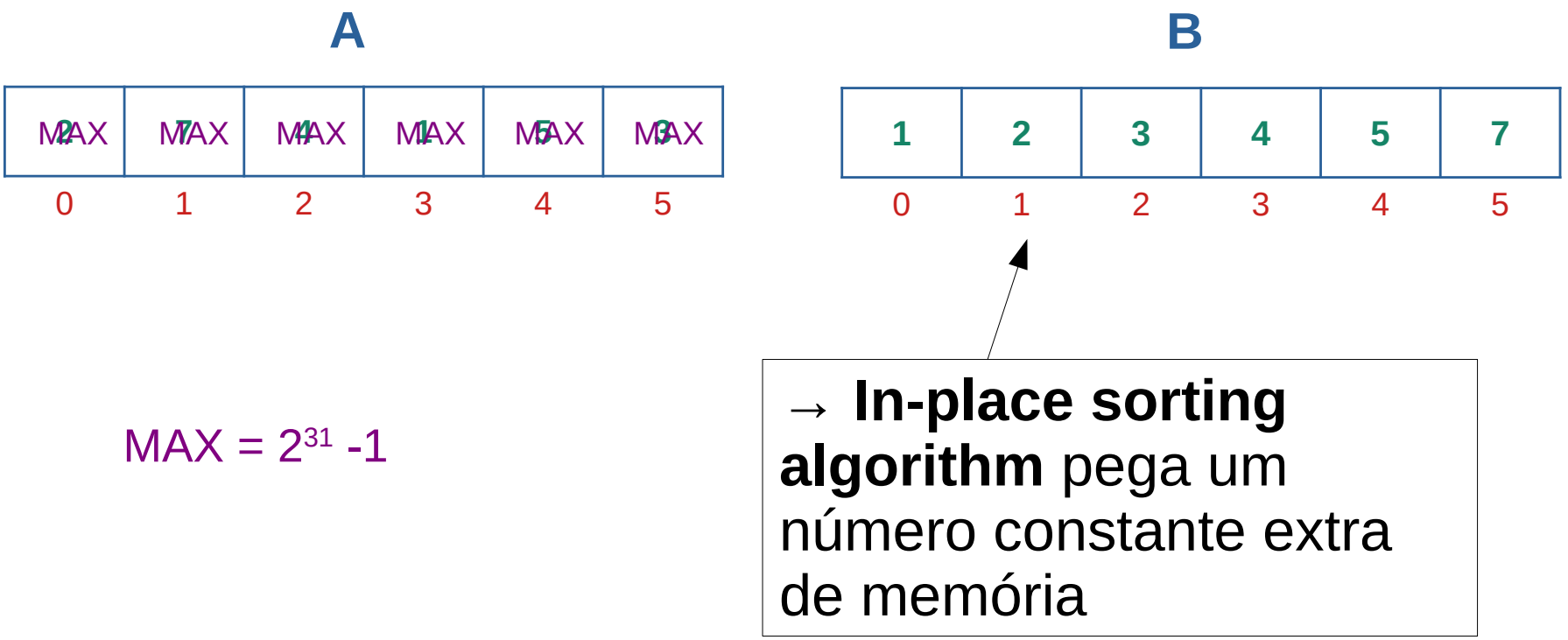
$MAX = 2^{31} - 1$

Selection Sort - 1ª abordagem



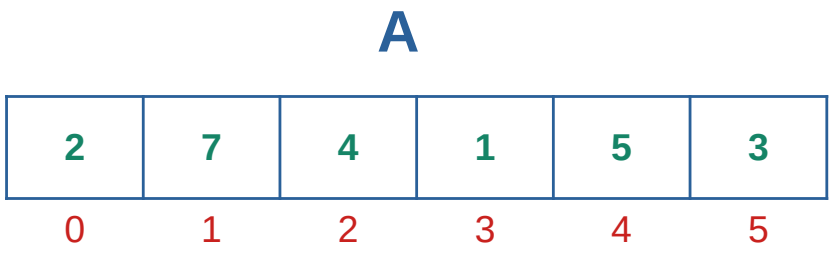
$MAX = 2^{31} - 1$

Selection Sort - 1ª abordagem

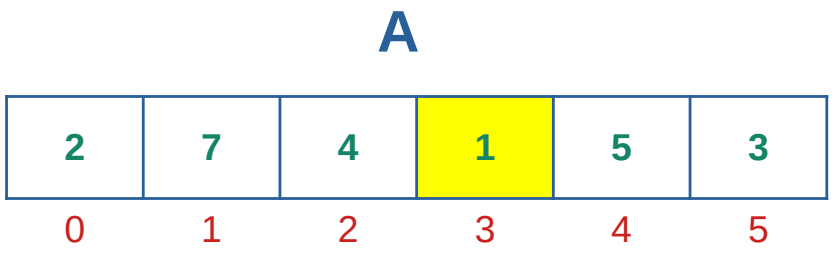


$MAX = 2^{31} - 1$

Selection Sort - 2ª abordagem



Selection Sort - 2ª abordagem

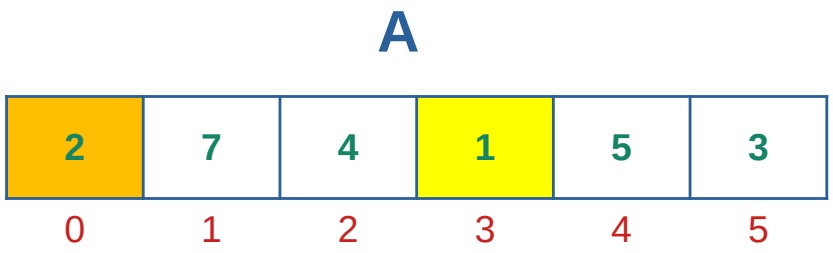


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 0

Selection Sort - 2ª abordagem

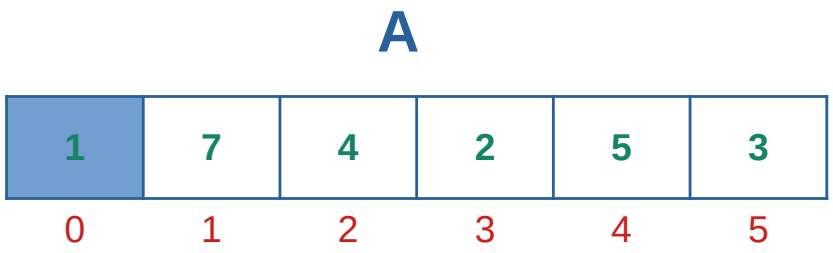


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 0

Selection Sort - 2ª abordagem

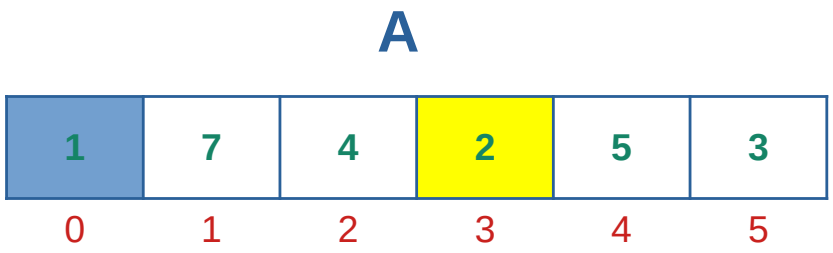


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 0

Selection Sort - 2ª abordagem

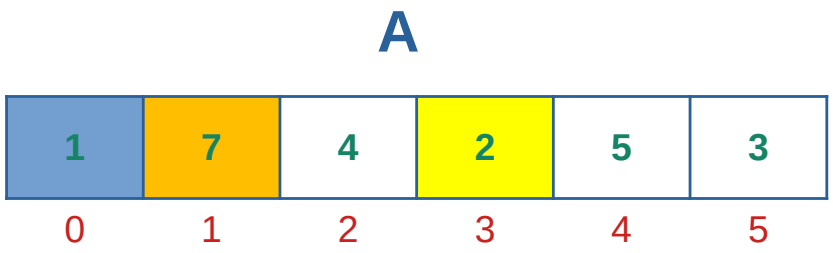


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 1

Selection Sort - 2ª abordagem

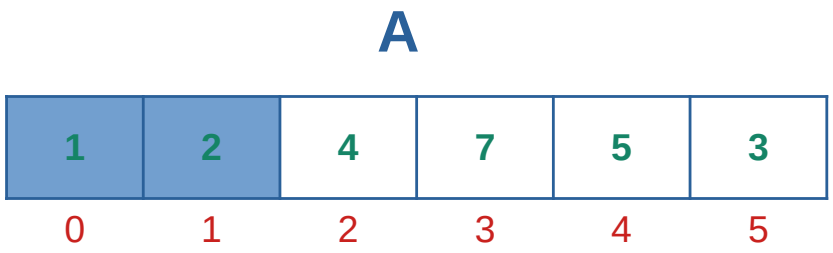


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 1

Selection Sort - 2ª abordagem

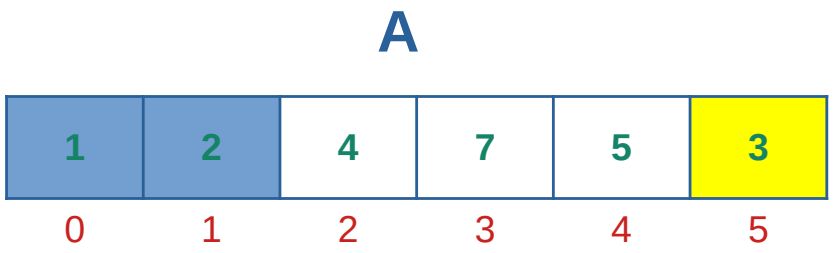


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 1

Selection Sort - 2ª abordagem

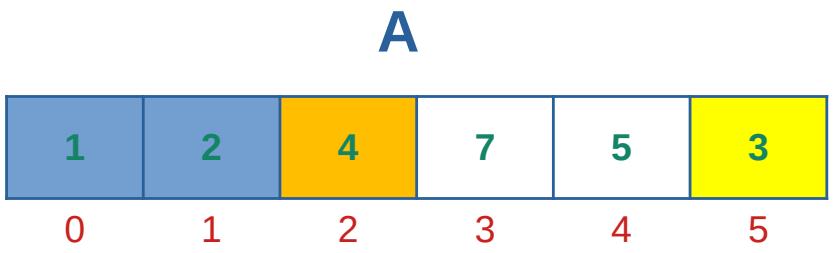


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 2

Selection Sort - 2ª abordagem

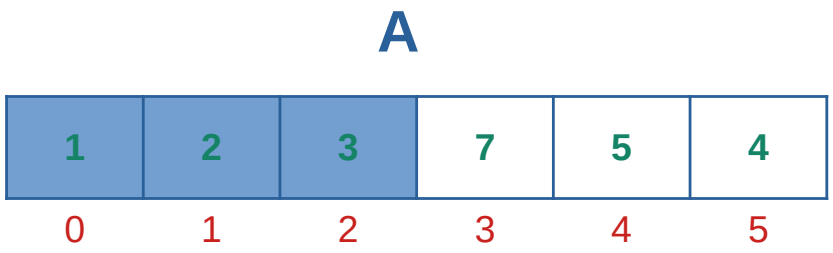


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 2

Selection Sort - 2ª abordagem

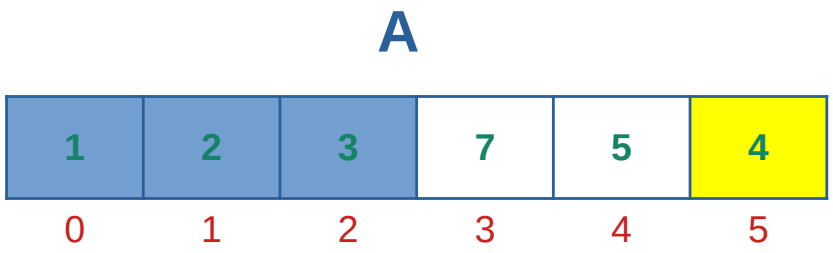


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 2

Selection Sort - 2ª abordagem

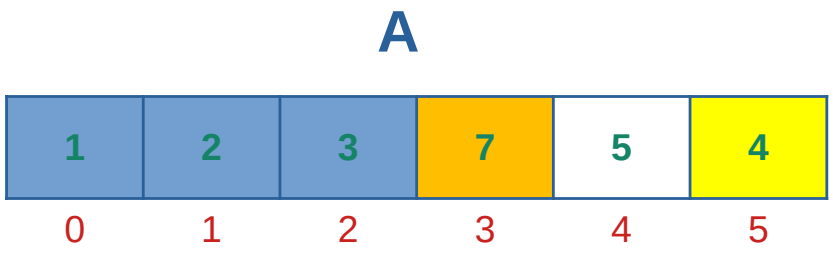


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 3

Selection Sort - 2ª abordagem

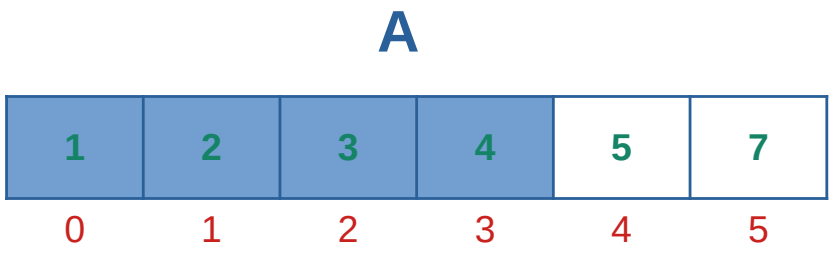


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 3

Selection Sort - 2ª abordagem

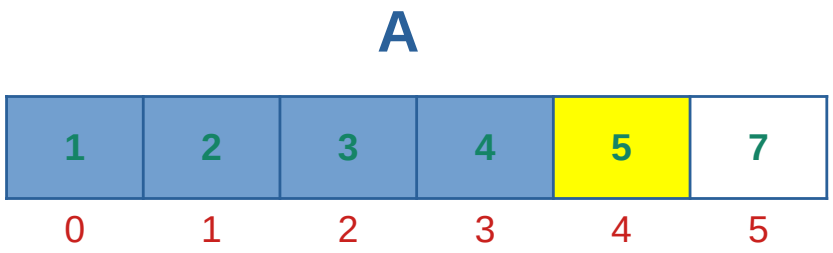


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 3

Selection Sort - 2ª abordagem

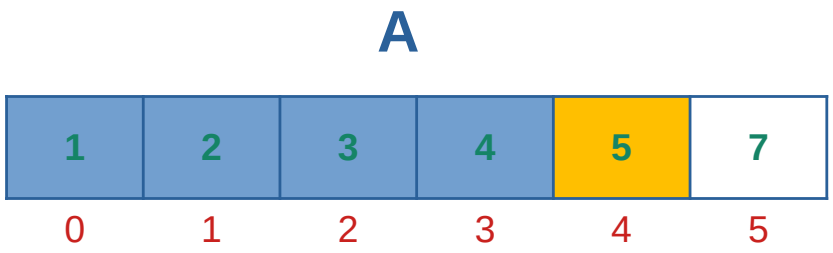


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 4

Selection Sort - 2ª abordagem

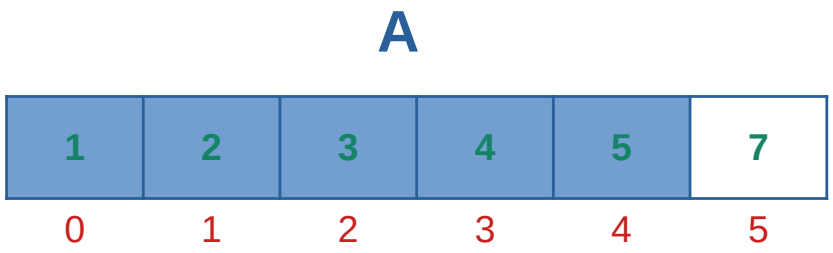


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 4

Selection Sort - 2ª abordagem

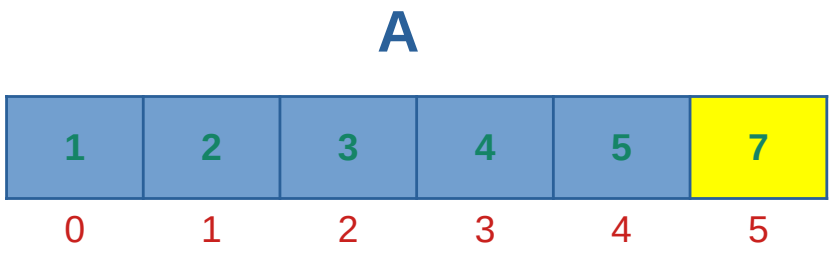


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 4

Selection Sort - 2ª abordagem

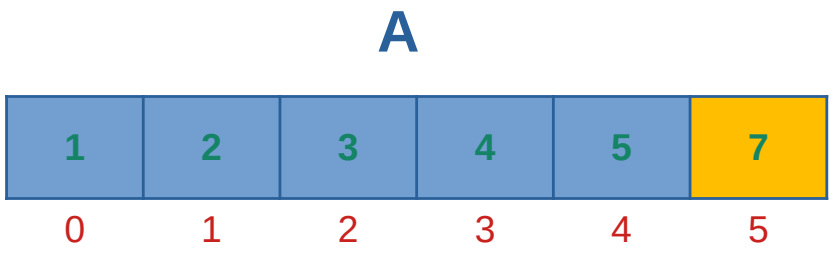


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 5

Selection Sort - 2ª abordagem

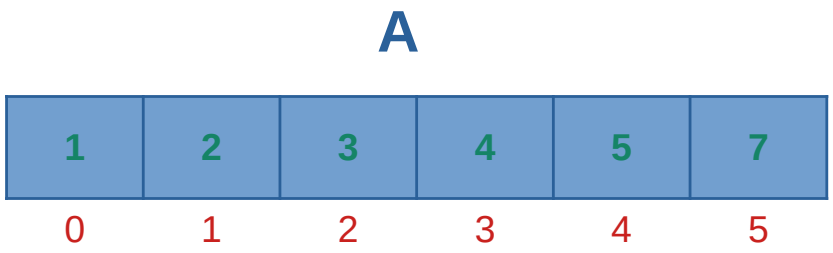


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 5

Selection Sort - 2ª abordagem

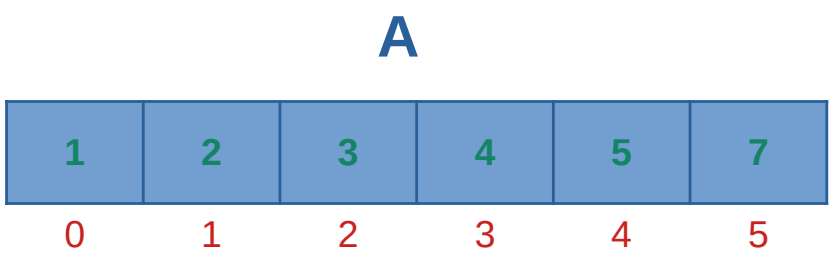


Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 5

Selection Sort - 2ª abordagem



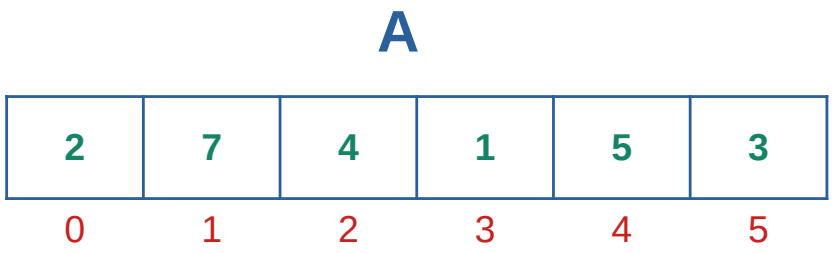
Ordenado!

Algoritmo:

1) Encontra o elemento mínimo

2) Troca ele com o elemento do índice 5

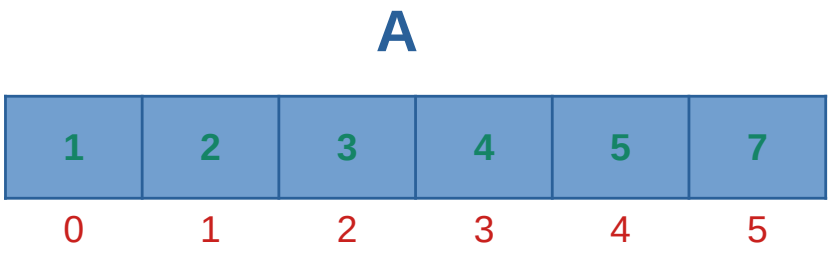
Selection Sort - 2ª abordagem



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

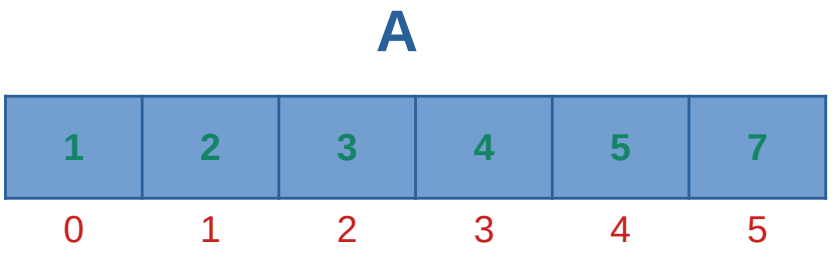


Custo computacional: $O(n^2)$

Pseudocódigo:

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

Selection Sort - Complexidade

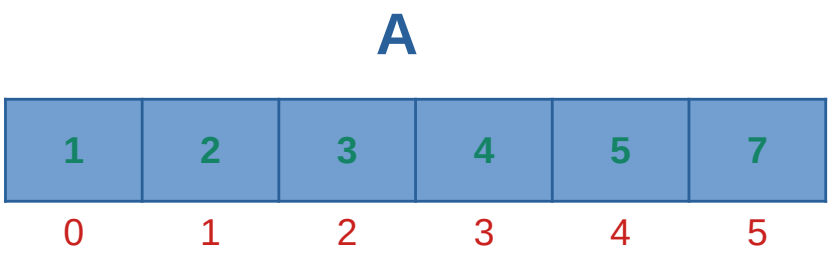


Custo computacional: $O(n^2)$

Pseudocódigo:

```
selectionSort(Array, n) {
    for i ← 0 to n-2 {
        iMin ← i                    c1 * n-1 vezes
        for j ← i+1 to n-1 {
            if(Array[j] < Array[iMin])      c2
                iMin ←j
        }
        temp ← Array[i]
        Array[i] ← Array[iMin]      C3 * n-1 vezes
        Array[iMin] ← temp
    }
}
```

Selection Sort - Complexidade



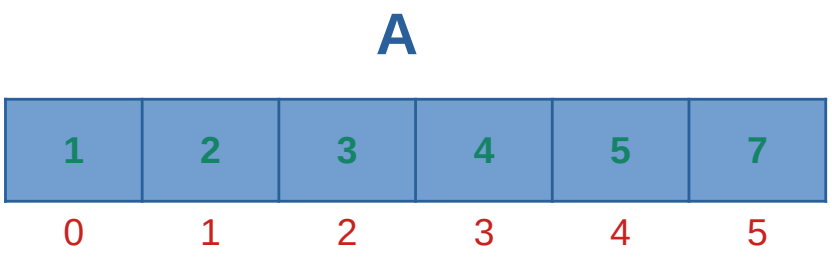
Custo computacional: $O(n^2)$

(n-1) + (n-2) + (n-3) + ... + 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 {
            if(Array[j] < Array[iMin])      c2
                iMin ←j
        }
        temp ← Array[i]
        Array[i] ← Array[iMin]      C3 * n-1 vezes
        Array[iMin] ← temp
    }
}
```

Selection Sort - Complexidade



Pseudocódigo:

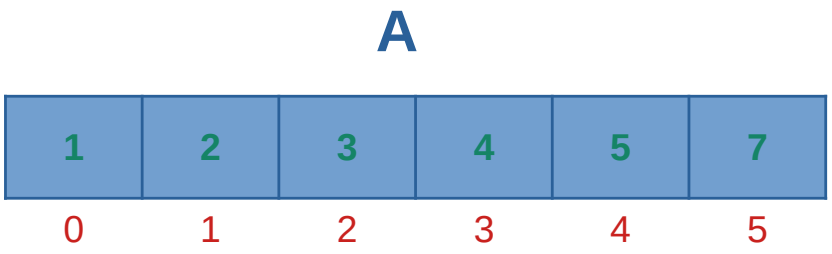
```
selectionSort(Array, n) {
    for i ← 0 to n-2 {
        iMin ← i                    c1 * n-1 vezes
        for j ← i+1 to n-1 {
            if(Array[j] < Array[iMin])      c2
                iMin ←j
        }
        temp ← Array[i]
        Array[i] ← Array[iMin]      C3 * n-1 vezes
        Array[iMin] ← temp
    }
}
```

Custo computacional: $O(n^2)$

(n-1) + (n-2) + (n-3) + ... + 1
=
n(n-1)/2

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

Selection Sort - Complexidade



Pseudocódigo:

```
selectionSort(Array, n) {
    for i ← 0 to n-2 {
        iMin ← i                    c1 * n-1 vezes
        for j ← i+1 to n-1 {
            if(Array[j] < Array[iMin])      c2
                iMin ←j
        }
        temp ← Array[i]
        Array[i] ← Array[iMin]      C3 * n-1 vezes
        Array[iMin] ← temp
    }
}
```

Custo computacional: $O(n^2)$

(n-1) + (n-2) + (n-3) + ... + 1
=
n(n-1)/2

T(n) = (n-1)*c1 + n(n-1)/2 * c2 + (n-1) * c3 → **T(n) = $an^2 + bn + c$** → $O(n^2)$

Selection Sort - Complexidade