Selection Sort

A classificação por seleção aprimora a classificação por bolha, fazendo apenas uma única troca para cada passagem do resumo. Para fazer isso, uma ordenação de seleção busca o maior valor à medida que faz um passe e, após terminar o passe, o posiciona na melhor área possível. Da mesma forma, como acontece com uma classificação por bolha, após a primeira passagem, o maior item está no lugar certo. Após a segunda passagem, a seguinte maior é configurada. Este procedimento continua e requer que n-1 vá para classificar n itens, uma vez que o último item deve ser configurado após a (n-1) ª passagem.

## ALGORITMO: SELEÇÃO DE SELEÇÃO (A)

1. 1 . k ← comprimento [A]
2. 2 . **para** j ← 1  a n- 1
3. 3 . menor ← j
4. 4 . **para** I ← j +  1  para k
5. 5 . **se** A [i] <A [menor]
6. 6 . então o menor ← i
7. 7 . troca (A [j], A [menor])

## Como funciona a classificação por seleção

1. Na ordenação de seleção, em primeiro lugar, definimos o elemento inicial como um **mínimo** .
2. Agora vamos comparar o mínimo com o segundo elemento. Se o segundo elemento acabar sendo menor do que o mínimo, vamos trocá-los e, em seguida, atribuir um mínimo ao terceiro elemento.
3. Caso contrário, se o segundo elemento for maior do que o mínimo, que é o nosso primeiro elemento, não faremos nada e passaremos para o terceiro elemento e então o compararemos com o mínimo.  
   Vamos repetir esse processo até chegarmos ao último elemento.
4. Após a conclusão de cada iteração, notaremos que nosso mínimo atingiu o início da lista não classificada.
5. Para cada iteração, iniciaremos a indexação do primeiro elemento da lista não classificada. Vamos repetir os passos de 1 a 4 até que a lista seja ordenada ou todos os elementos sejam posicionados corretamente.  
   Considere o seguinte exemplo de uma matriz não classificada que classificaremos com a ajuda do algoritmo de Classificação por Seleção.

A [] = (7, 4, 3, 6, 5).  
A [] =



**1r iteração:**

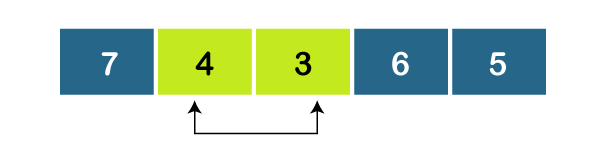
Definir mínimo = 7

* Compare um 0 e um 1



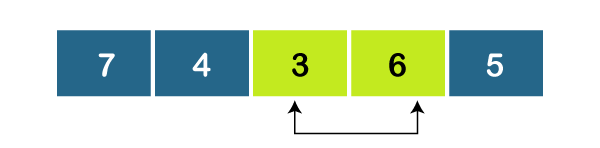
Como, a 0 > a 1 , defina mínimo = 4.

* Compare 1 e 2



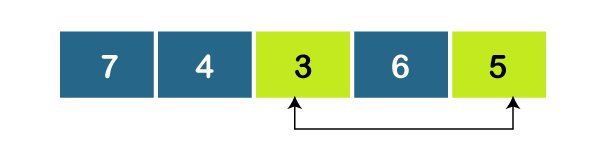
Como, a 1 > a 2 , defina mínimo = 3.

* Compare um 2 e um 3



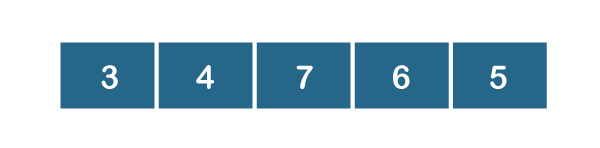
Como, a 2 <a 3 , defina o mínimo = 3.

* Compare um 2 e um 4



Como, a 2 <a 4 , defina o mínimo = 3.

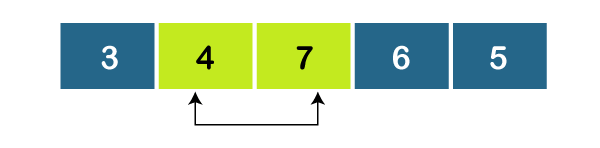
Como 3 é o menor elemento, trocaremos 0 por 2 .



**2nd iteração:**

Definir mínimo = 4

* Compare 1 e 2



Como, a 1 <a 2 , defina mínimo = 4.

* Compare 1 e 3



Como, A [1] <A [3], defina mínimo = 4.

* Compare 1 e 4



Novamente, a 1 <a 4 , defina o mínimo = 4.

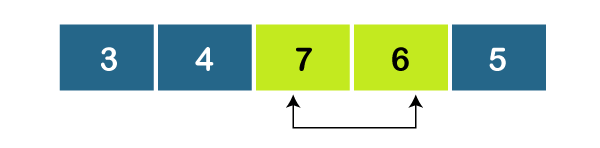
Como o mínimo já está colocado na posição correta, não haverá troca.



**3rd iteração:**

Definir mínimo = 7

* Compare um 2 e um 3



Como, a 2 > a 3 , defina mínimo = 6.

* Compare um 3 e um 4



Como, a 3 > a 4 , defina mínimo = 5.

Como 5 é o menor elemento entre os elementos não classificados restantes, trocaremos 7 e 5.



**4th iteração:**

Definir mínimo = 6

* Compare um 3 e um 4



Como 3 <a 4 , defina mínimo = 6.

Como o mínimo já está colocado na posição correta, não haverá troca.

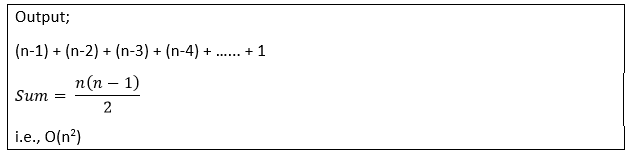


## Análise de complexidade de classificação por seleção

**Entrada:** Dados **n** elementos de entrada.

**Saída:** número de etapas incorridas para classificar uma lista.

**Lógica:** se tivermos n elementos, então, na primeira passagem, ele fará **n-1** comparações; na segunda passagem, ele fará **n-2** ; na terceira passagem, fará **n-3** e assim por diante. Assim, o número total de comparações pode ser encontrado por;



Portanto, o algoritmo de classificação de seleção abrange uma complexidade de tempo de **O (n 2 )** e uma complexidade de espaço de **O (1)** porque necessita de algum espaço de memória extra para a variável temporária para troca.

### Complexidades de tempo:

* **Complexidade de melhor caso:** O algoritmo de ordenação de seleção tem uma complexidade de tempo de melhor caso de **O (n 2 )** para a matriz já classificada.
* **Complexidade de caso médio:** A complexidade de tempo de caso médio para o algoritmo de ordenação de seleção é **O (n 2 )** , em que os elementos existentes estão em ordem confusa, ou seja, nem em ordem crescente nem decrescente.
* **Complexidade de** pior caso: a complexidade de tempo de pior caso também é **O (n 2 )** , que ocorre quando classificamos a ordem decrescente de uma matriz na ordem crescente.

No algoritmo de ordenação por seleção, a complexidade do tempo é **O (n 2 )** em todos os três casos. Isso porque, em cada etapa, somos obrigados a encontrar elementos **mínimos** para que ele possa ser colocado na posição correta. Assim que rastrearmos a matriz completa, obteremos nosso elemento mínimo.

## Complexidade

|  |  |  |  |
| --- | --- | --- | --- |
| **Complexidade** | **Melhor caso** | **Caso Médio** | **Pior caso** |
| Tempo | Ω (n) | θ (n 2 ) | o (n 2 ) |
| Espaço |  |  | o (1) |