Bubble Sort

Bubble Sort, também conhecido como Exchange Sort, é um algoritmo de classificação simples. Funciona percorrendo repetidamente a lista a ser classificada, comparando dois itens por vez e trocando-os se estiverem na ordem errada. A passagem pela lista é duplicada até que nenhuma troca seja desejada, o que significa que a lista é classificada.

Este é o método mais fácil entre todos os algoritmos de classificação.

## Algoritmo

**Etapa 1 ➤ Inicialização**

1. conjunto  1  ← n, p ←  1

**Etapa 2 ➤ loop,**

1. Repita até a etapa  4 **enquanto** (p ≤ n- 1 )
2. definir E ←  0  ➤ Variável de troca de inicialização.

**Etapa 3 ➤ comparação, loop.**

1. Repita  **para** i ←  1 ,  1 ,… ... l- 1 .
2. **if** (A [i]> A [i +  1 ]) então
3. definir A [i] ↔ A [i +  1 ] ➤ Trocando valores.
4. Conjunto E ← E +  1

**Etapa 4 ➤ Concluir ou reduzir o tamanho.**

1. **se** (E =  0 ) então
2. saída
3. **outro**
4. definir l ← l -  1 .

## Como funciona a classificação por bolha

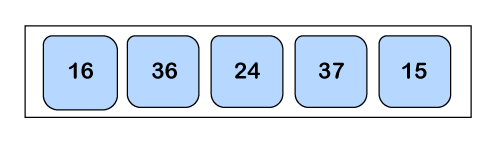
1. A classificação da bolha começa com o primeiro índice e o torna um elemento da bolha. Em seguida, ele compara o elemento de bolha, que atualmente é nosso primeiro elemento de índice, com o próximo elemento. Se o elemento de bolha for maior e o segundo elemento for menor, ambos serão trocados.  
   Após a troca, o segundo elemento se tornará o elemento bolha. Agora vamos comparar o segundo elemento com o terceiro, como fizemos na etapa anterior, e trocá-los, se necessário. O mesmo processo é seguido até o último elemento.
2. Seguiremos o mesmo processo para o restante das iterações. Após cada iteração, notaremos que o maior elemento presente no array não classificado atingiu o último índice.

Para cada iteração, a classificação por bolha comparará até o último elemento não classificado.

Depois que todos os elementos forem classificados em ordem crescente, o algoritmo será encerrado.

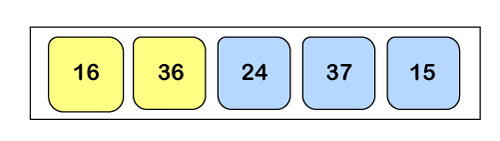
Considere o seguinte exemplo de uma matriz não classificada que classificaremos com a ajuda do algoritmo Bubble Sort.

**Inicialmente,**



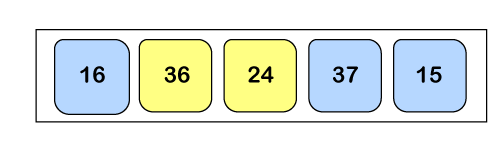
**Passe 1:**

* **Compare um 0 e um 1**

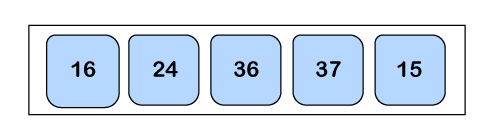


Como 0 <a 1 , a matriz permanecerá como está.

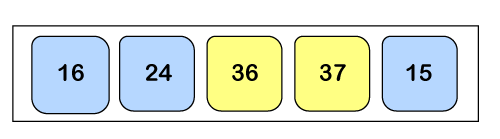
* **Compare 1 e 2**



Agora, a 1 > a 2 , então vamos trocar os dois.

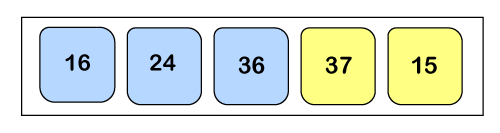


* **Compare um 2 e um 3**

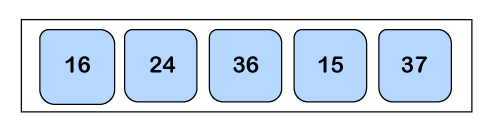


Como 2 <a 3 , a matriz permanecerá como está.

* **Compare um 3 e um 4**

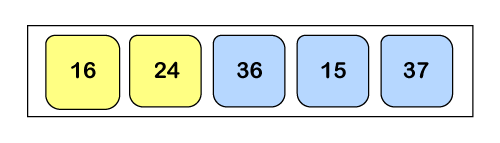


Aqui, um 3 > um 4 , então vamos trocar novamente os dois.



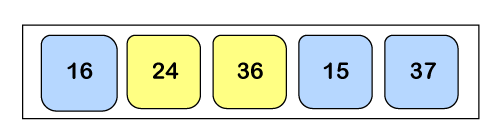
**Passe 2:**

* **Compare um 0 e um 1**



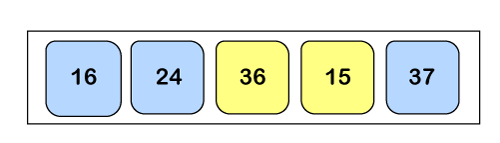
Como 0 <a 1 , a matriz permanecerá como está.

* **Compare 1 e 2**

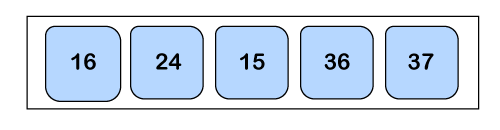


Aqui, a 1 <a 2 , então a matriz permanecerá como está.

* **Compare um 2 e um 3**

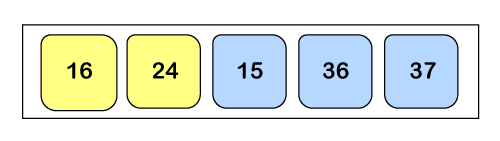


Nesse caso, a 2 > a 3 , então os dois serão trocados.



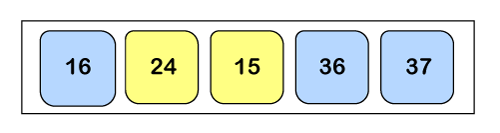
**Passe 3:**

* **Compare um 0 e um 1**

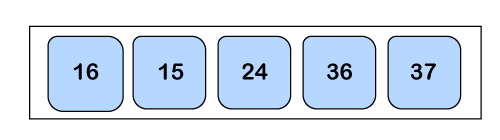


Como 0 <a 1 , a matriz permanecerá como está.

* **Compare 1 e 2**

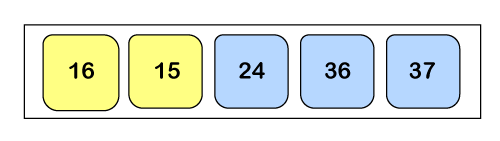


Agora a 1 > a 2, então os dois serão trocados.

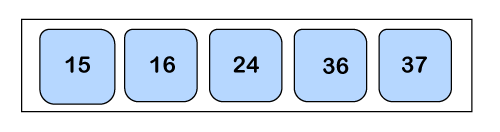


**Passe 4:**

* **Compare um 0 e um 1**



Aqui, a 0 > a 1 , então vamos trocar os dois.



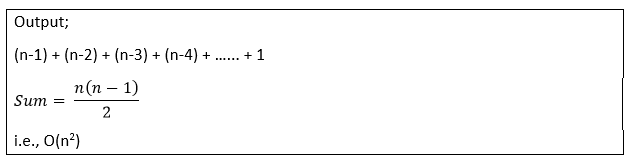
Portanto, a matriz é classificada, já que não há mais necessidade de troca.

## Análise de complexidade de classificação de bolhas

**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 bolhas 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:

* **Melhor caso de complexidade**: O algoritmo de classificação de bolha tem um melhor caso de complexidade de tempo de **O (n)** para a matriz já classificada.
* **Complexidade de caso médio**: A complexidade de tempo de caso médio para o algoritmo de classificação de bolhas é **O (n 2 )** , que acontece quando 2 ou mais elementos estão misturados, ou seja, nem na ordem crescente nem na ordem 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.

## Complexidade

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

## Vantagens do Bubble Sort

1. Facilmente compreensível.
2. Não necessita de memória extra.
3. O código pode ser escrito facilmente para este algoritmo.
4. Requisito de espaço mínimo do que outros algoritmos de classificação.

## Desvantagens do Bubble Sort

1. Não funciona bem quando temos listas grandes e não classificadas e exige mais recursos que acabam demorando muito.
2. Destina-se apenas a fins acadêmicos, não para implementações práticas.
3. Envolve a ordem de n 2 etapas para classificar um algoritmo.