Insertion Sort

A classificação por inserção é um dos algoritmos de classificação mais simples, pois classifica um único elemento em uma instância específica. Não é o melhor algoritmo de classificação em termos de desempenho, mas é ligeiramente mais eficiente do que a [classificação por seleção](https://www.javatpoint.com/daa-selection-sort) e [classificação por bolha](https://www.javatpoint.com/daa-bubble-sort) em cenários práticos. É uma técnica de classificação intuitiva.

Vamos considerar o exemplo de cartões para entender melhor a lógica por trás da classificação por inserção.

Suponha que temos um conjunto de cartas em nossa mão, de modo que queremos organizá-las em ordem crescente. Para classificar esses cartões, temos várias maneiras intuitivas.

Uma coisa que podemos fazer é, inicialmente, segurar todas as cartas em nossa mão esquerda, e podemos começar a pegar cartas uma após a outra na mão esquerda, seguido pela construção de um arranjo ordenado na mão direita.

Supondo que a primeira carta já esteja classificada, selecionaremos a próxima carta não classificada. Se o cartão não classificado for maior do que o cartão selecionado, vamos simplesmente colocá-lo no lado direito, caso contrário, no lado esquerdo. Em qualquer estágio durante todo o processo, a mão esquerda não será classificada e a direita será classificada.

Da mesma forma, classificaremos o restante das cartas não classificadas, colocando-as na posição correta. A cada iteração, o algoritmo de inserção coloca um elemento não classificado em seu lugar certo.

## ALGORITMO: TIPO DE INSERÇÃO (A)

1. 1. para  j  =  2  a A. comprimento
2. 2.   chave  =  A [j]
3. 3. // Insira A [j] na sequência classificada A [1 .. j - 1]
4. 4.   i  =  j  - 1
5. 5. enquanto i  **>** 0 e A [i]  **>** tecla
6. 6. A [i + 1] = A [i]
7. 7.       i i  = i -1
8. 8. A [i + 1] = chave

## Como funciona a classificação por inserção

1. Começaremos assumindo que o primeiro elemento da matriz já está classificado. Dentro da **chave** , iremos armazenar o segundo elemento.

A seguir, compararemos nosso primeiro elemento com a **chave** , de modo que se **a chave** for menor que o primeiro elemento, trocaremos seus índices ou colocaremos a chave no primeiro índice. Depois de fazer isso, notaremos que os primeiros dois elementos estão classificados.

2. Agora, vamos passar para o terceiro elemento e compará-lo com os elementos do lado esquerdo. Se for o menor elemento, colocaremos o terceiro elemento no primeiro índice.

Do contrário, se for maior que o primeiro elemento e menor que o segundo elemento, trocaremos sua posição com o terceiro elemento e o colocaremos após o primeiro elemento. Depois de fazer isso, teremos nossos três primeiros elementos ordenados.

3. Da mesma forma, vamos classificar o resto dos elementos e colocá-los em suas posições corretas.

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

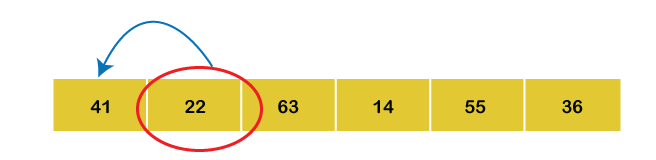
A = (41, 22, 63, 14, 55, 36)

Inicialmente,

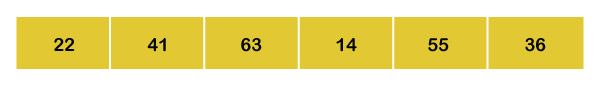
**1r** iteração:

Definir chave = 22

Compare a1 com a0



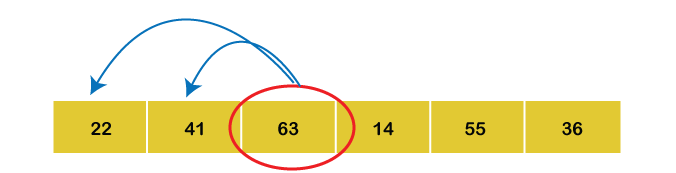
Como a0> a1, troque os dois.



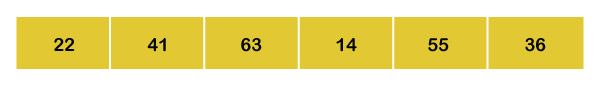
2nd iteração:

Definir chave = 63

Compare a2 com a1 e a0



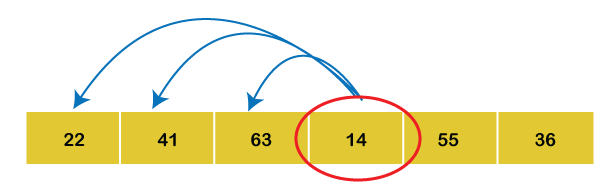
Como a2> a1> a0, mantenha o array como está.



3rd iteração:

Definir chave = 14

Compare a3 com a2, a1 e a0



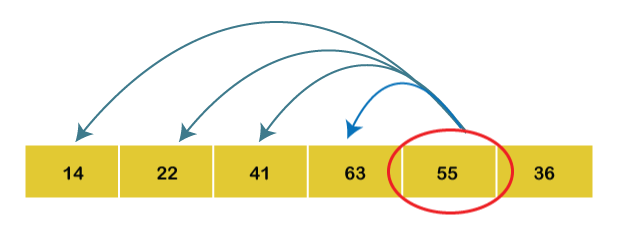
Como a3 é o menor entre todos os elementos do lado esquerdo, coloque a3 no início da matriz.



4th iteração:

Definir chave = 55

Compare a4 com a3, a2, a1 e a0.



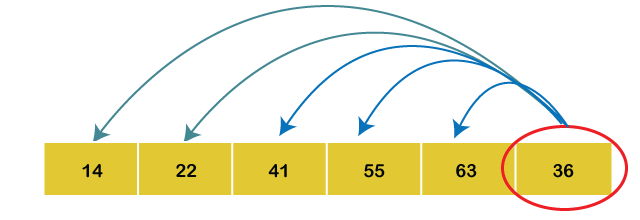
Como a4 <a3, troque os dois.



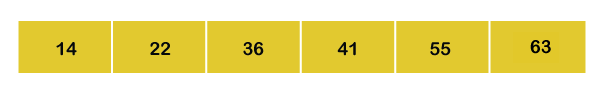
5n iteração:

Definir chave = 36

Compare a5 com a4, a3, a2, a1 e a0.



Como a5 <a2, colocaremos os elementos em suas posições corretas.



Consequentemente, a matriz é organizada em ordem crescente, portanto, nenhuma troca é necessária.

## Análise de complexidade do tipo de inserção

**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;

Resultado;

(n-1) + (n-2) + (n-3) + (n-4) + ...... + 1

Soma = 

ie, O (n 2 )

Portanto, o algoritmo de ordenação por inserção 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 uma variável- **chave** realizar trocas.

### Complexidades de tempo:

* **Complexidade de melhor caso:** o algoritmo de ordenação de inserção tem uma complexidade de tempo de melhor caso de **O (n)** para a matriz já classificada porque aqui, apenas o loop externo está executando n vezes e o loop interno é mantido imóvel.
* **Complexidade de caso médio:** A complexidade de tempo de caso médio para o algoritmo de ordenação por inserção é **O (n 2 )** , que ocorre quando os elementos existentes estão em ordem confusa, 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 crescente de uma matriz na ordem decrescente.  
  Neste algoritmo, cada elemento individual é comparado com o resto dos elementos, devido ao qual n-1 comparações são feitas para cada  elemento.

O algoritmo de ordenação por inserção é altamente recomendado, especialmente quando alguns elementos são deixados para ordenação ou no caso de o array conter poucos elementos.

### Complexidade do Espaço

A classificação de inserção abrange uma complexidade de espaço de **O (1)** devido ao uso de uma **chave de** variável extra.

## Complexidade

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

## Aplicativos de classificação por inserção

O algoritmo de classificação por inserção é usado nos seguintes casos:

* Quando a matriz contém apenas alguns elementos.
* Quando existem poucos elementos para classificar.

## Vantagens da classificação por inserção

1. É simples de implementar.
2. É eficiente em pequenos conjuntos de dados.
3. É estável (não altera a ordem relativa dos elementos com chaves iguais)
4. Ele está instalado (requer apenas uma quantidade constante O (1) de espaço de memória extra).
5. É um algoritmo online, que pode classificar uma lista quando ela é recebida.