Insertion Sort Algorithm

In this article, we will discuss the Insertion sort Algorithm. The working procedure of insertion sort is also simple. This article will be very helpful and interesting to students as they might face insertion sort as a question in their examinations. So, it is important to discuss the topic.

Insertion sort works similar to the sorting of playing cards in hands. It is assumed that the first card is already sorted in the card game, and then we select an unsorted card. If the selected unsorted card is greater than the first card, it will be placed at the right side; otherwise, it will be placed at the left side. Similarly, all unsorted cards are taken and put in their exact place.

The same approach is applied in insertion sort. The idea behind the insertion sort is that first take one element, iterate it through the sorted array. Although it is simple to use, it is not appropriate for large data sets as the time complexity of insertion sort in the average case and worst case is **$O(n^2)$**, where n is the number of items. Insertion sort is less efficient than the other sorting algorithms like heap sort, quick sort, merge sort, etc.

Insertion sort has various advantages such as -

- Simple implementation
- Efficient for small data sets
- Adaptive, i.e., it is appropriate for data sets that are already substantially sorted.

Now, let's see the algorithm of insertion sort.

Algorithm

The simple steps of achieving the insertion sort are listed as follows -

Step 1 - If the element is the first element, assume that it is already sorted. Return 1.

Step2 - Pick the next element, and store it separately in a **key.**

Step3 - Now, compare the **key** with all elements in the sorted array.

Step 4 - If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.

Step 5 - Insert the value.

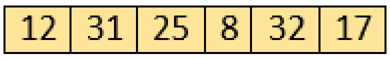
Step 6 - Repeat until the array is sorted.

Working of Insertion sort Algorithm

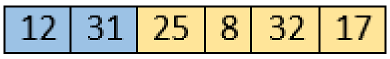
Now, let's see the working of the insertion sort Algorithm.

To understand the working of the insertion sort algorithm, let's take an unsorted array. It will be easier to understand the insertion sort via an example.

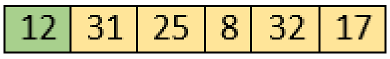
Let the elements of array are -



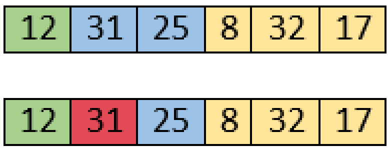
Initially, the first two elements are compared in insertion sort.



Here, 31 is greater than 12. That means both elements are already in ascending order. So, for now, 12 is stored in a sorted sub-array.

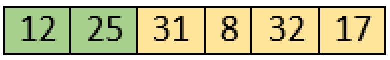


Now, move to the next two elements and compare them.

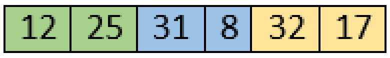


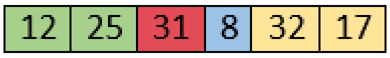
Here, 25 is smaller than 31. So, 31 is not at correct position. Now, swap 31 with 25. Along with swapping, insertion sort will also check it with all elements in the sorted array.

For now, the sorted array has only one element, i.e. 12. So, 25 is greater than 12. Hence, the sorted array remains sorted after swapping.

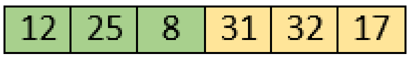


Now, two elements in the sorted array are 12 and 25. Move forward to the next elements that are 31 and 8.

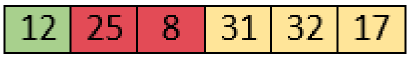




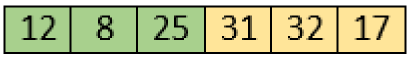
Both 31 and 8 are not sorted. So, swap them.



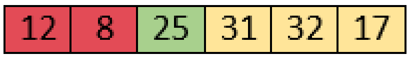
After swapping, elements 25 and 8 are unsorted.



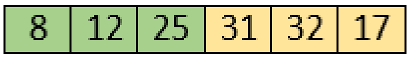
So, swap them.



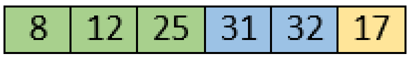
Now, elements 12 and 8 are unsorted.



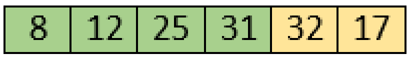
So, swap them too.



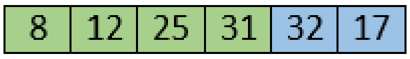
Now, the sorted array has three items that are 8, 12 and 25. Move to the next items that are 31 and 32.



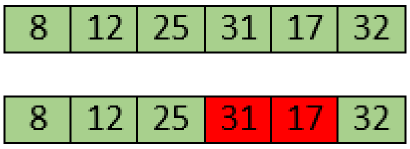
Hence, they are already sorted. Now, the sorted array includes 8, 12, 25 and 31.



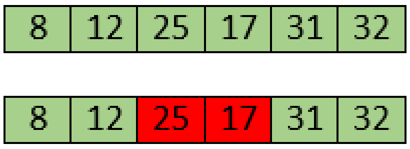
Move to the next elements that are 32 and 17.



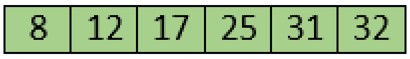
17 is smaller than 32. So, swap them.



Swapping makes 31 and 17 unsorted. So, swap them too.



Now, swapping makes 25 and 17 unsorted. So, perform swapping again.



Now, the array is completely sorted.

Insertion sort complexity

Now, let's see the time complexity of insertion sort in best case, average case, and in worst case. We will also see the space complexity of insertion sort.

1. Time Complexity

Case	Time Complexity
Best Case	O(n)
Average Case	O(n^2)
Worst Case	O(n^2)

- **Best Case Complexity -** It occurs when there is no sorting required, i.e. the array is already sorted. The best-case time complexity of insertion sort is **O(n)**.
- **Average Case Complexity -** It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending. The average case time complexity of insertion sort is **O(n^2)**.

- **Worst Case Complexity -** It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order. The worst-case time complexity of insertion sort is **O(n^2)**.

2. Space Complexity

Space Complexity	O(1)
Stable	YES

- The space complexity of insertion sort is O(1). It is because, in insertion sort, an extra variable is required for swapping.

Implementation of insertion sort

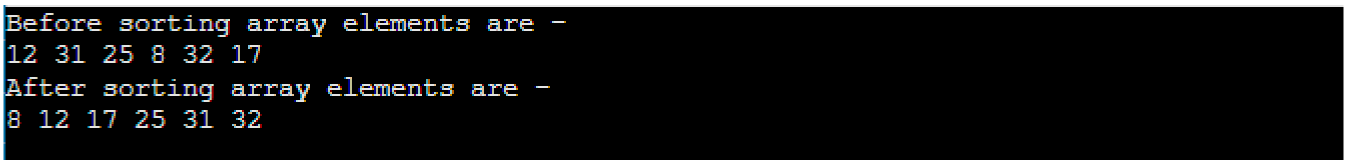
Now, let's see the programs of insertion sort in different programming languages.

Program: Write a program to implement insertion sort in C language.

```
#include <stdio.h>

void insert(int a[], int n) /* function to sort an aay with insertion sort */
{
    int i, j, temp;
    for (i = 1; i < n; i++) {
        temp = a[i];
        j = i - 1;

        while(j>=0 && temp <= a[j])  /* Move the elements greater than temp to one position a
        {
            a[j+1] = a[j];
            j = j-1;
        }
        a[j+1] = temp;
    }
}
```

```
void printArr(int a[], int n) /* function to print the array */
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
}

int main()
{
    int a[] = { 12, 31, 25, 8, 32, 17 };
    int n = sizeof(a) / sizeof(a[0]);
    printf("Before sorting array elements are - \n");
    printArr(a, n);
    insert(a, n);
    printf("\nAfter sorting array elements are - \n");
    printArr(a, n);

    return 0;
}
```

Output:

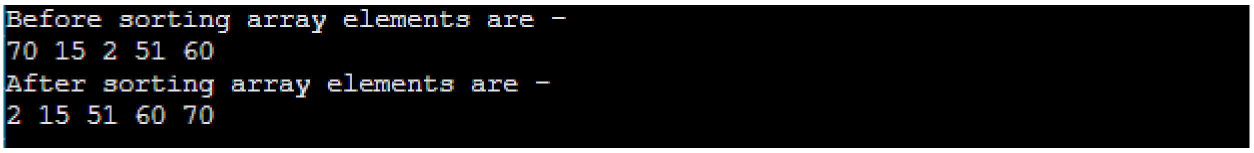


```
Before sorting array elements are -
12 31 25 8 32 17
After sorting array elements are -
8 12 17 25 31 32
```

Program: Write a program to implement insertion sort in python.

```
def insertionSort(a): # Function to implement insertion sort
    for i in range(1, len(a)):
        temp = a[i]

        # Move the elements greater than temp to one position
        #ahead from their current position
        j = i-1
        while j >= 0 and temp < a[j] :
                a[j + 1] = a[j]
                j = j-1
            a[j + 1] = temp

def printArr(a): # function to print the array

    for i in range(len(a)):
        print (a[i], end = " ")

a = [70, 15, 2, 51, 60]
print("Before sorting array elements are - ")
printArr(a)
insertionSort(a)
print("\nAfter sorting array elements are - ")
printArr(a)
```

Output:

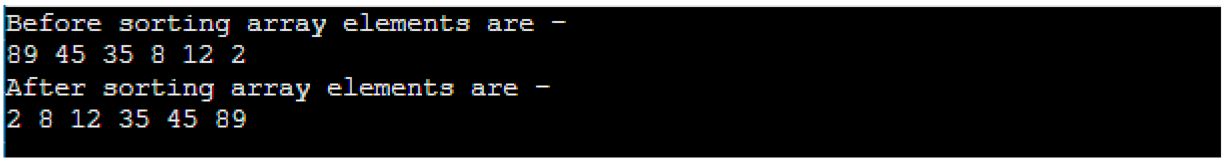


```
Before sorting array elements are -
70 15 2 51 60
After sorting array elements are -
2 15 51 60 70
```

Program: Write a program to implement insertion sort in C++ language.

```
#include <iostream>
using namespace std;

void insert(int a[], int n) /* function to sort an aay with insertion sort */
{
    int i, j, temp;
    for (i = 1; i < n; i++) {
        temp = a[i];
        j = i - 1;

        while(j>=0 && temp <= a[j])  /* Move the elements greater than temp to one position a
        {
            a[j+1] = a[j];
            j = j-1;
        }
        a[j+1] = temp;
    }
}

void printArr(int a[], int n) /* function to print the array */
{
    int i;
    for (i = 0; i < n; i++)
        cout << a[i] <<" ";
}

int main()
{
    int a[] = { 89, 45, 35, 8, 12, 2 };
    int n = sizeof(a) / sizeof(a[0]);
    cout<<"Before sorting array elements are - "<<endl;
    printArr(a, n);
    insert(a, n);
    cout<<"\nAfter sorting array elements are - "<<endl;
    printArr(a, n);

    return 0;
}
```

Output:

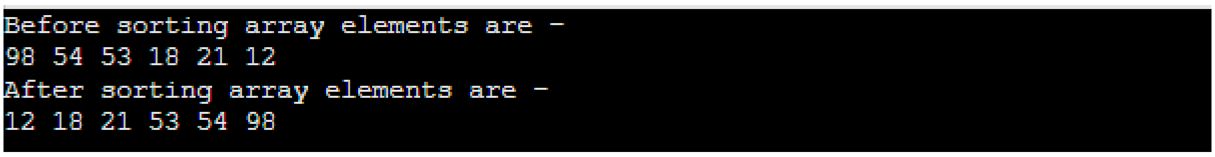


```
Before sorting array elements are -
89 45 35 8 12 2
After sorting array elements are -
2 8 12 35 45 89
```

Program: Write a program to implement insertion sort in C# language.

```
using System;
class Insertion {
static void insert(int[] a) /* function to sort an aay with insertion sort */
{
    int i, j, temp;
    int n = a.Length;
    for (i = 1; i < n; i++) {
        temp = a[i];
        j = i - 1;

        while(j>=0 && temp <= a[j])  /* Move the elements greater than temp to one position a
        {
            a[j+1] = a[j];
            j = j-1;
        }
        a[j+1] = temp;
    }
}

static void printArr(int[] a) /* function to print the array */
{
    int i;
    int n = a.Length;
    for (i = 0; i < n; i++)
    Console.Write(a[i] + " ");
}
  static void Main() {
    int[] a = { 98, 54, 53, 18, 21, 12 };
    Console.Write("Before sorting array elements are - \n");
    printArr(a);
    insert(a);
    Console.Write("\nAfter sorting array elements are - \n");
    printArr(a);
  }
}
```

Output:

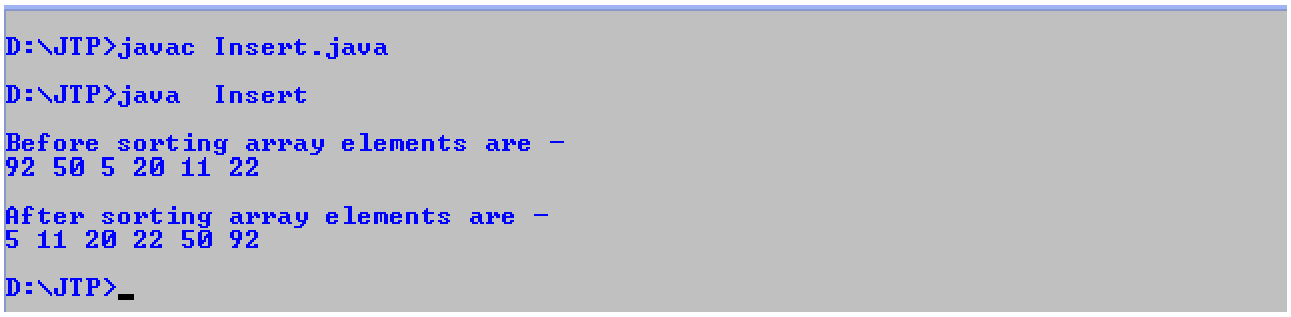


```
Before sorting array elements are -
98 54 53 18 21 12
After sorting array elements are -
12 18 21 53 54 98
```

Program: Write a program to implement insertion sort in Java.

```
public class Insert
{
    void insert(int a[]) /* function to sort an aay with insertion sort */
{
    int i, j, temp;
    int n = a.length;
    for (i = 1; i < n; i++) {
        temp = a[i];
        j = i - 1;

        while(j>=0 && temp <= a[j])  /* Move the elements greater than temp to one position a
        {
            a[j+1] = a[j];
            j = j-1;
        }
        a[j+1] = temp;
    }
}
void printArr(int a[]) /* function to print the array */
{
    int i;
    int n = a.length;
    for (i = 0; i < n; i++)
    System.out.print(a[i] + " ");
}

    public static void main(String[] args) {
    int a[] = { 92, 50, 5, 20, 11, 22 };
    Insert i1 = new Insert();
    System.out.println("\nBefore sorting array elements are - ");
    i1.printArr(a);
    i1.insert(a);
    System.out.println("\n\nAfter sorting array elements are - ");
    i1.printArr(a);
    System.out.println();
    }
}
```

Output:

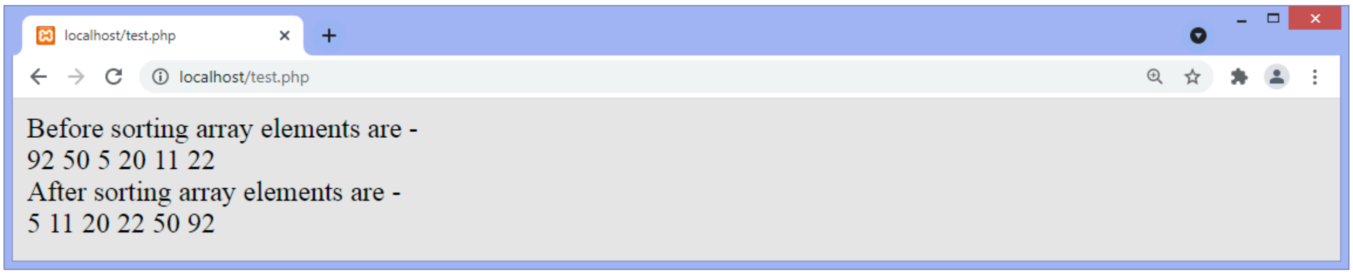


```
D:\JTP>javac Insert.java

D:\JTP>java  Insert

Before sorting array elements are -
92 50 5 20 11 22

After sorting array elements are -
5 11 20 22 50 92

D:\JTP>_
```

Program: Write a program to implement insertion sort in PHP.

```
<?php
```

Output:



```
Before sorting array elements are -
92 50 5 20 11 22
After sorting array elements are -
5 11 20 22 50 92
```

So, that's all about the article. Hope the article will be helpful and informative to you.

This article was not only limited to the algorithm. We have also discussed the algorithm's complexity, working, and implementation in different programming languages.

← Prev Next →

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Learn Latest Tutorials

Splunk tutorial Splunk	SPSS tutorial SPSS	Swagger tutorial Swagger	T-SQL tutorial Transact-SQL
Tumblr tutorial Tumblr	React tutorial ReactJS	Regex tutorial Regex	Reinforcement learning tutorial Reinforcement Learning
R Programming tutorial R Programming	RxJS tutorial RxJS	React Native tutorial React Native	Python Design Patterns Python Design Patterns
Python Pillow tutorial Python Pillow	Python Turtle tutorial Python Turtle	Keras tutorial Keras	

Preparation

Aptitude Aptitude	Logical Reasoning	Verbal Ability Verbal Ability	Interview Questions

Reasoning

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence Tutorial

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing tutorial

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System tutorial

Operating System

Computer Network tutorial

Computer Network

Compiler Design tutorial

Compiler Design

Computer Organization and Architecture

Discrete Mathematics Tutorial

Computer Organization

Discrete Mathematics

Ethical Hacking Tutorial

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering Tutorial

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse