



Tutorial Link <https://course.testpad.chitkara.edu.in/tutorials/InsertionSort/5a12e8a046765b2b63e3474e>

TUTORIAL

Insertion Sort

Topics

- 1.2 Insertion Sort Code
- 1.5 Properties of Insertion sort
- 1.6 Video Solution

One of the simplest methods to sort an array is an insertion sort. An example of an insertion sort occurs in everyday life while playing cards. To sort the cards in your hand you extract a card, shift the remaining cards, and then insert the extracted card in the correct place. This process is repeated until all the cards are in the correct sequence. Following algorithm will describe the insertion sort procedure: -

```
for i = 1 to n-1
    Pick element at position i
    insert it into sorted sequence from index 0 to i-1.
end
```

Let's take the below example: -

15 11 14 12 18

To sort this array in ascending order, Insertion sort will perform following steps: -

When i=1

15 is the element, first element is always sorted so nothing to do.

When i=2

11 is the element. Now to insert 11 in array from 0 to 1 ($i-1=1$) we have to shift 15 and then insert 11. So array is

```
11  15  14  12  18
When i=3
```

14 is the element. Now to insert 14 in array from 0 to 2 ($i-1=2$) we have to shift 15 and then insert 14. So array is

```
11  14  15  12  18
When i=4
```

12 is the element. Now to insert 12 in array from 0 to 3 ($i-1=3$) we have to shift 15, 14 and then insert 12. So array is

```
11  12  14  15  18
When i=5
```

18 is the element. Now to insert 18 in array from 0 to 4 ($i-1=4$) we do not have to shift any number as 18 is the largest number. So array is

```
11  12  14  15  18
```

Following is the implementation of insertion sort: -

Insertion Sort Code

```
1 function insertionSort(array,n){
2     let i, key, j;
3     for (i = 0; i < n; i++){
4         key = array[i];
5         j = i-1;
6         while (j >= 0 && array[j] > key)
7             {                               // find the correct
position of the element
8                 array[j+1] = array[j];       // shift all
lesser elements
9                 j = j-1;
10            }
11            array[j+1] = key;                 // place the
element at position
```

Javascript

```
12     }
13 }
14
15 function main(){
16     let A = [15,11,14,12,18]
17     console.log('Unsorted Array:')
18     console.log(A.join(' '));
19     console.log()
20
21     insertionSort(A,A.length);
22
23     console.log('Sorted Array');
24     console.log(A.join(' '));
25 }
26
27 main()
```

```
1 #include<stdio.h>
2
3 void printArray(int array[], int size){
4     int i;
5     for (i=0; i < size; i++)
6         printf("%d ", array[i]);
7     printf("\n");
8 }
9
10 void insertionSort(int array[], int n){
11     int i, key, j;
12     for (i = 0; i < n; i++){
13         key = array[i];
14         j = i-1;
15         while (j >= 0 && array[j] > key){
16             // find the correct position of the element
17             array[j+1] = array[j];           // shift all
lesser elements
18             j = j-1;
19         }
20         array[j+1] = key;                     // place the
element at position
21
22     }
```

```

23 }
24
25 int main()
26 {
27     int array[] = {15, 11, 14, 12, 18};
28     int n = 5;
29     /* we can calculate the number of elements in an
        array by using sizeof(array)/sizeof(array[0]).
        */
30     printf("Un-Sorted array: \n");
31     printArray(array, n);    // Unsorted array
32     insertionSort(array, n);    // Call the sorting
        routine
33     printf("\nSorted array: \n");
34     printArray(array, n);    // Sorted array
35     return 0;
36 }
37

```

```

1  import java.util.Scanner;
2  // Other imports go here
3  // Do NOT change the class name
4  class Main{
5      static void printArray(int array[], int size){
6          int i;
7          for (i=0; i < size; i++)
8              System.out.printf("%d ", array[i]);
9              System.out.printf("\n");
10     }
11
12     static void insertionSort(int array[], int n)
13     {
14         int i, key, j;
15         for (i = 0; i < n; i++){
16             key = array[i];
17             j = i-1;
18
19             while (j >= 0 && array[j] > key)
20                 {
21                     // find the
                correct position of the element
22                     array[j+1] = array[j];    //
                shift all lesser elements
23                     j = j-1;

```

Java

```

23         }
24         array[j+1] = key;           // place
the element at position
25     }
26 }
27
28 public static void main(String[] args){
29     int array[] = {15, 11, 14, 12, 18};
30     int n = 5;
31     /* we can calculate the number of elements
in an array by using
sizeof(array)/sizeof(array[0]).*/
32     System.out.printf("Un-Sorted array: \n");
33     printArray(array, n);           // Unsorted array
34     insertionSort(array, n);       // Call the
sorting routine
35     System.out.printf("\nSorted array: \n");
36     printArray(array, n);           // Sorted array
37 }
38 }
39

```

```

1 def printArray(A,size):
2     for i in range(size):
3         print(A[i],end=' ');
4     print()
5
6 def insertionSort(array,size):
7     for i in range(1, size):
8         key = array[i]
9         j = i-1
10        while j >=0 and key < array[j] :
11            array[j+1] = array[j]
12            j -= 1
13        array[j+1] = key
14
15 if __name__=="__main__":
16     A = [15,11,14,12,18]
17     print('Unsorted Array:')
18     printArray(A, len(A));
19     print()
20

```

Python 3

```
21     insertionSort(A, len(A));
22
23     print('Sorted Array');
24     printArray(A, len(A));
```

```
1  #include<iostream>
2  using namespace std;
3
4  void printArray(int array[], int size){
5      int i;
6      for (i=0; i < size; i++){
7          cout<<array[i]<<" ";
8      }
9      cout<<endl;
10 }
11
12 void insertionSort(int array[], int n){
13     int i, key, j;
14     for (i = 0; i < n; i++){
15         key = array[i];
16         j = i-1;
17         while (j >= 0 && array[j] > key)
18             {
19                 // find the correct
20                 position of the element
21                 array[j+1] = array[j];          // shift
22                 all lesser elements
23                 j = j-1;
24             }
25         array[j+1] = key;          // place the
26         element at position
27     }
28 }
29
30 int main(){
31     int array[] = {15, 11, 14, 12, 18};
32     int n = 5;
33     /* we can calculate the number of elements in
34     an array by using sizeof(array)/sizeof(array[0]).*/
35     cout<<"Un-Sorted array:"<<endl;
36     printArray(array, n);          // Unsorted array
37     insertionSort(array, n);        // Call the
38     sorting routine
39     cout<<endl<<"Sorted array:"<<endl;
```

C++

```
34     printArray(array, n);    // Sorted array
35     return 0;
36 }
37
```

Output of above program: -

Un-Sorted array:

15 11 14 12 18

Sorted array:

11 12 14 15 18

The output above shows the elements shifted in each pass and final array after each pass. The algorithm goes to inner loop only if required. So, the time complexity of this algorithm is $O(n^2)$ if both loops in `insertionSort()` function will run n times. Otherwise the complexity will be $O(n)$ if array is already sorted.

Properties of Insertion sort

Worst and Average Case Time Complexity: $O(n^2)$. Worst case occurs when array is sorted in opposite direction.

Best Case Time Complexity: $O(n)$. Best case occurs when array is already sorted.

Auxiliary Space: $O(1)$

Sorting In Place: Yes

Stable: Yes

Insertion sort is used when number of elements is small. It can also be useful when input array is almost sorted, only few elements are misplaced in complete big array.

Video Solution

```
<iframe width="560" height="315"  
src="https://www.youtube.com/embed/_fXogocanDQ"  
title="YouTube video player" frameborder="0" allow="accelerometer;  
autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-  
picture" allowfullscreen></iframe>
```



CodeQuotient

codequotient.com

Tutorial by codequotient.com | All rights

reserved, CodeQuotient 2025