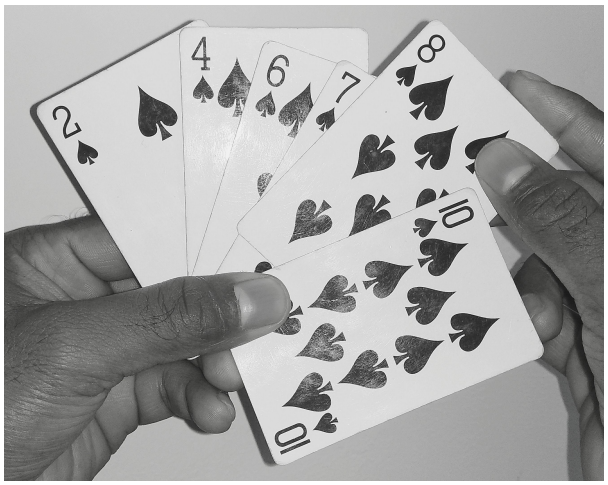
geeksforgeeks.org

# Insertion Sort - GeeksforGeeks

5-6 minutes

---

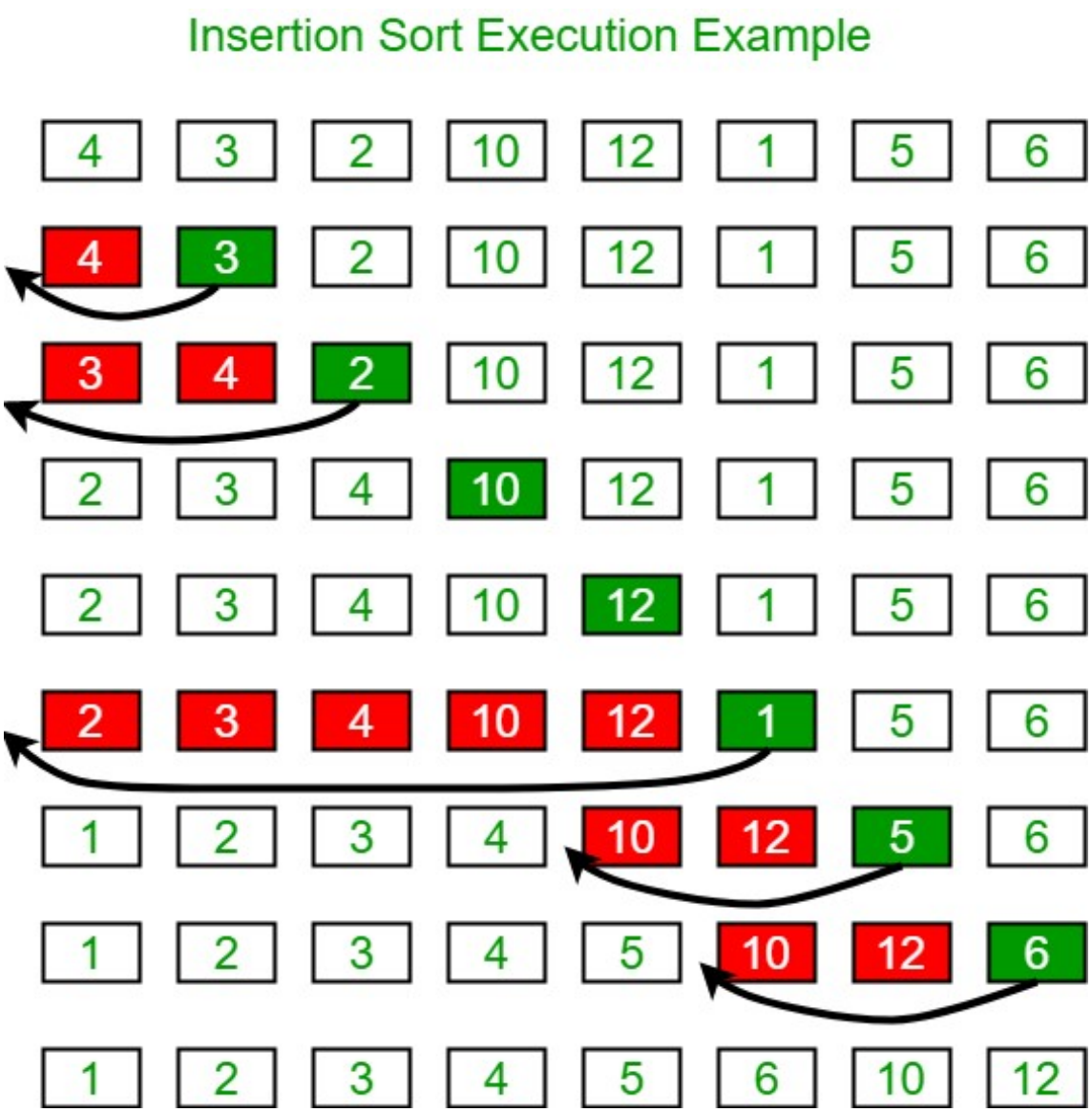
Insertion sort is a simple sorting algorithm that works the way we sort playing cards in our hands.



**Algorithm**

// Sort an arr[] of size n
insertionSort(arr, n)
Loop from i = 1 to n-1.
……a) Pick element arr[i] and insert it into sorted sequence arr[0…i-1]

**Example:**




**Another Example:**

**12**, 11, 13, 5, 6

Let us loop for i = 1 (second element of the array) to 5 (Size of input array)

i = 1. Since 11 is smaller than 12, move 12 and insert 11 before 12

**11, 12**, 13, 5, 6

i = 2. 13 will remain at its position as all elements in A[0..I-1] are smaller than 13

**11, 12, 13**, 5, 6

i = 3. 5 will move to the beginning and all other elements from 11 to 13 will move one position ahead of their current position.

**5, 11, 12, 13**, 6

i = 4. 6 will move to position after 5, and elements from 11 to 13 will move one position ahead of their current position.
**5, 6, 11, 12, 13**

- C/C++
- Python
- Java

## C/C++

```
#include <stdio.h>
#include <math.h>
void insertionSort(int arr[], int n)
{
    int i, key, j;
    for (i = 1; i < n; i++)
    {
        key = arr[i];
        j = i-1;
        while (j >= 0 && arr[j] > key)
        {
            arr[j+1] = arr[j];
            j = j-1;
        }
        arr[j+1] = key;
    }
}
```

```
void printArray(int arr[], int n)
{
   int i;
   for (i=0; i < n; i++)
       printf("%d ", arr[i]);
   printf("\n");
}
int main()
{
    int arr[] = {12, 11, 13, 5, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

## Python

```
def insertionSort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >=0 and key < arr[j] :
                arr[j+1] = arr[j]
                j -= 1
```

```
        arr[j+1] = key
arr = [12, 11, 13, 5, 6]
insertionSort(arr)
print ("Sorted array is:")
for i in range(len(arr)):
    print ("%d" %arr[i])
```

## Java

```
class InsertionSort
{
    void sort(int arr[])
    {
        int n = arr.length;
        for (int i=1; i<n; ++i)
        {
            int key = arr[i];
            int j = i-1;
            while (j>=0 && arr[j] > key)
            {
                arr[j+1] = arr[j];
                j = j-1;
            }
            arr[j+1] = key;
        }
```

```
    }
    static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }
    public static void main(String args[])
    {
        int arr[] = {12, 11, 13, 5, 6};
        InsertionSort ob = new
InsertionSort();
        ob.sort(arr);
        printArray(arr);
    }
}
```

Output:

```
5 6 11 12 13
```

**Time Complexity:** O(n*n)

**Auxiliary Space:** O(1)

**Boundary Cases**: Insertion sort takes maximum time to sort if elements are sorted in reverse order. And it takes minimum time (Order of n) when elements are already sorted.

**Algorithmic Paradigm:** Incremental Approach

**Sorting In Place:** Yes

**Stable:** Yes

**Online:** Yes

**Uses:** 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.

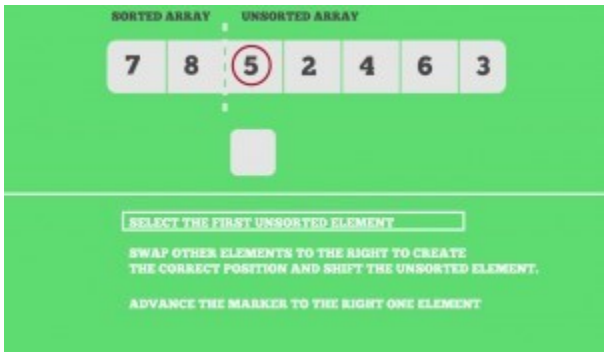
**What is Binary Insertion Sort?**
We can use binary search to reduce the number of comparisons in normal insertion sort. Binary Insertion Sort find use binary search to find the proper location to insert the selected item at each iteration. In normal insertion, sort it takes O(i) (at ith iteration) in worst case. we can reduce it to O(logi) by using binary search. The algorithm as a whole still has a running worst case running time of O(n2) because of the series of swaps required for each insertion. Refer this for implementation.

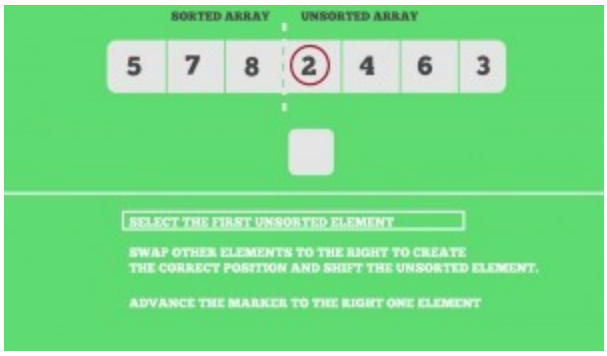
**How to implement Insertion Sort for Linked List?**
Below is simple insertion sort algorithm for linked list.

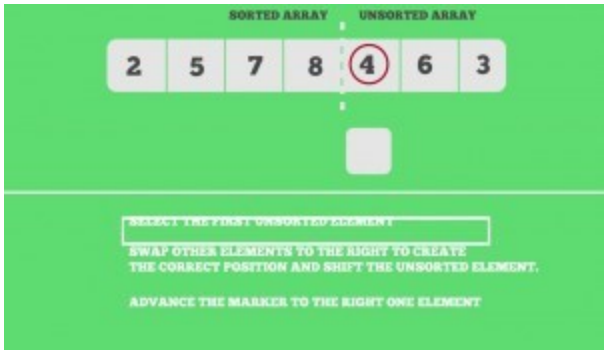
```
1) Create an empty sorted (or result) list
2) Traverse the given list, do following for every
node.
......a) Insert current node in sorted way in
sorted or result list.
3) Change head of given linked list to head of
sorted (or result) list.
```

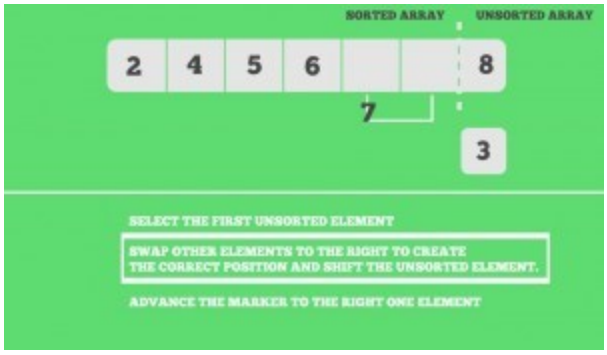
Refer this for implementation.

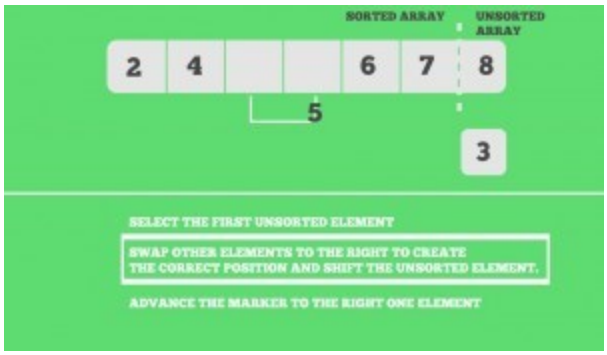
**Snapshots:**

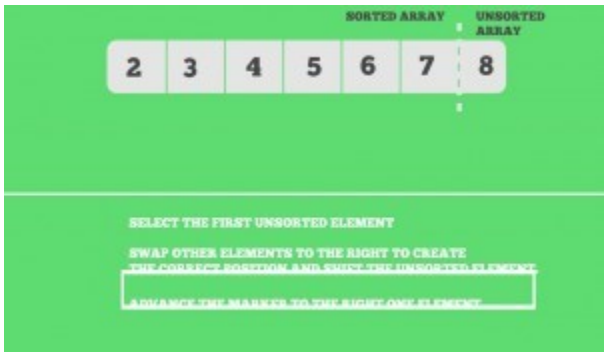

SORTED ARRAY
UNSORTED ARRAY
7 8 5 2 4 6 3
SELECT THE FIRST UNSORTED ELEMENT
SWAP OTHER ELEMENTS TO THE RIGHT TO CREATE
THE CORRECT POSITION AND SHIFT THE UNSORTED ELEMENT.
ADVANCE THE MARKER TO THE RIGHT ONE ELEMENT


SORTED ARRAY
UNSORTED ARRAY
5 7 8 2 4 6 3
SELECT THE FIRST UNSORTED ELEMENT
SWAP OTHER ELEMENTS TO THE RIGHT TO CREATE
THE CORRECT POSITION AND SHIFT THE UNSORTED ELEMENT.
ADVANCE THE MARKER TO THE RIGHT ONE ELEMENT


SORTED ARRAY
UNSORTED ARRAY
2 5 7 8 4 6 3
SWAP OTHER ELEMENTS TO THE RIGHT TO CREATE
THE CORRECT POSITION AND SHIFT THE UNSORTED ELEMENT.
ADVANCE THE MARKER TO THE RIGHT ONE ELEMENT


SORTED ARRAY
UNSORTED ARRAY
2 4 5 6 8
7
3
SELECT THE FIRST UNSORTED ELEMENT
SWAP OTHER ELEMENTS TO THE RIGHT TO CREATE
THE CORRECT POSITION AND SHIFT THE UNSORTED ELEMENT.
ADVANCE THE MARKER TO THE RIGHT ONE ELEMENT


SORTED ARRAY
UNSORTED ARRAY
2 4 6 7 8
5
3
SELECT THE FIRST UNSORTED ELEMENT
SWAP OTHER ELEMENTS TO THE RIGHT TO CREATE
THE CORRECT POSITION AND SHIFT THE UNSORTED ELEMENT.
ADVANCE THE MARKER TO THE RIGHT ONE ELEMENT




## Quiz on Insertion Sort

**Other Sorting Algorithms on GeeksforGeeks/GeeksQuiz**
Selection Sort, Bubble Sort, Insertion Sort, Merge Sort, Heap Sort, QuickSort, Radix Sort, Counting Sort, Bucket Sort, ShellSort, Comb Sort,

## Coding practice for sorting.

Image Source: http://www.just.edu.jo/~basel/algorithms/Algo%20Slides/algo_ch2_getting_started.pdf
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.