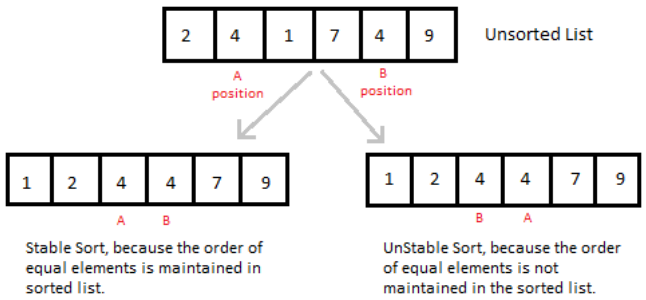
Insertion Sorting

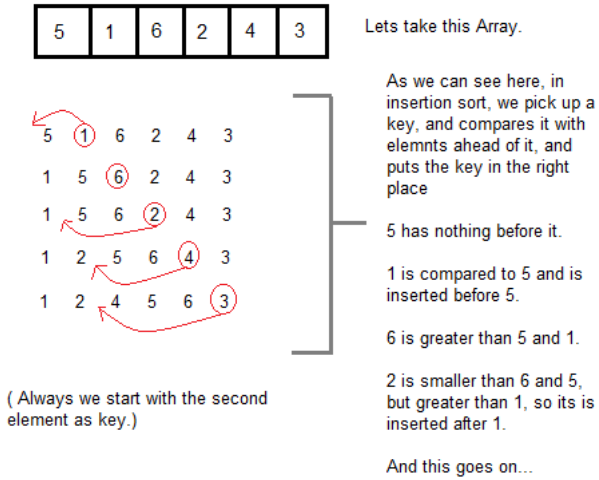
It is a simple Sorting algorithm which sorts the array by shifting elements one by one. Following are some of the important characteristics of Insertion Sort.

1. It has one of the simplest implementation
2. It is efficient for smaller data sets, but very inefficient for larger lists.
3. Insertion Sort is adaptive, that means it reduces its total number of steps if given a partially sorted list, hence it increases its efficiency.
4. It is better than Selection Sort and Bubble Sort algorithms.
5. Its space complexity is less, like Bubble Sorting, inerstion sort also requires a single additional memory space.
6. It is **Stable**, as it does not change the relative order of elements with equal keys



How Insertion Sorting Works

Coming Soon



Sorting using Insertion Sort Algorithm

```
int a[6] = {5, 1, 6, 2, 4, 3};
int i, j, key;
for(i=1; i<6; i++)
{
  key = a[i];
  j = i-1;
  while(j>=0 && key < a[j])
  {
    a[j+1] = a[j];
    j--;
  }
  a[j+1] = key;
}
```

Now lets, understand the above simple insertion sort algorithm. We took an array with 6 integers. We took a variable **key**, in which we put each element of the array, in each pass, starting from the second element, that is **a[1]**.

Then using the while loop, we iterate, until **j** becomes equal to zero or we find an element which is greater than **key**, and then we insert the key at that position.

In the above array, first we pick 1 as key, we compare it with 5(element before 1), 1 is smaller than 5, we shift 1 before 5. Then we pick 6, and compare it with 5 and 1, no shifting this time. Then 2 becomes the key and is compared with, 6 and 5, and then 2 is placed after 1. And this goes on, until complete array gets sorted.

Complexity Analysis of Insertion Sorting

Worst Case Time Complexity : $O(n^2)$

Best Case Time Complexity : $O(n)$

Average Time Complexity : $O(n^2)$

Space Complexity : O(1)

← Prev (bubble-sort)

Next → (selection-sorting)