# Insertion Sort

What is Insertion Sort ? Explain with java code step by step.

## Insertion Sort

Def:1 : " Insertion sort is a simple & intuitive comparison-based sorting algorithm. It works by dividing the list into a sorted & an unsorted region. Elements from the unsorted region are picked & placed at the correct position in the sorted region. "

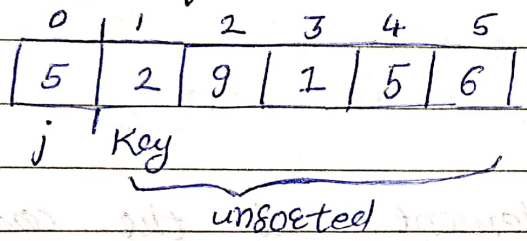
Def:2 : " Insertion sort is one of the best sorting techniques. It is twice as fast as Bubble sort. In Insertion sort the elements comparison are as less as compared to bubble sort. In this comparison the values until all prior elements are less than the compared values is not found. This means that all the previous values are lesser than compared value. Insertion sort is good choice for small values & for nearly sorted values. "

## Step-by-Step Explanation

1) Initialization :

- Consider the 1st element of the array as the sorted part.

- The remaining elements are the unsorted part.

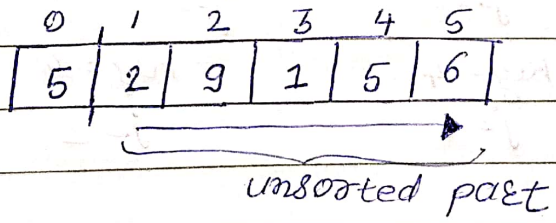



$j = i - 1$

Key = 2

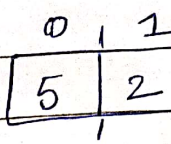
2) Iterate over the Unsorted Part :

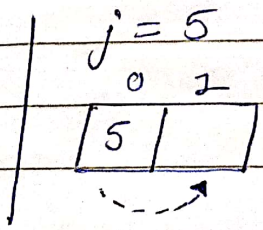
- Start from the 2nd element (index 1) & iterate through the array.

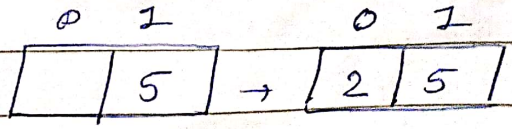



3) Pick & Compare :

- For each element in the unsorted part, compare it with the elements in the sorted part, starting from the end of the sorted part.

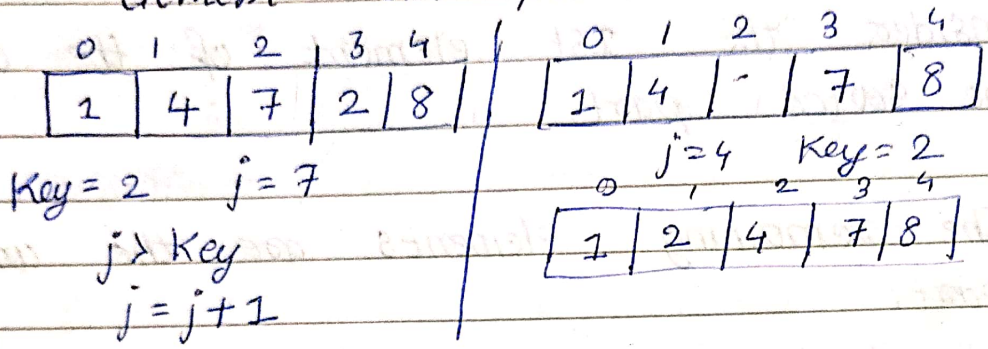


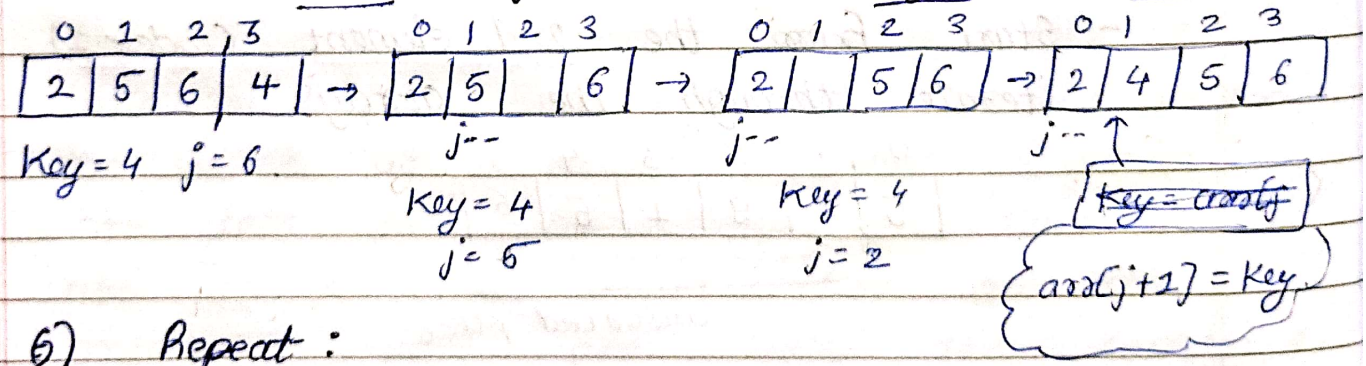
4) Shift Elements :

- Shift all elements in the sorted part that are greater than the picked element one position to the right.




5) Insert the Element :

- Insert the picked element at the correct position where all elements to its left are smaller & to its right are longer.




6) Repeat :

- Repeat the process for all elements in the unsorted part until the entire array is sorted.

Example :-

Consider the array : [ 5, 2, 9, 1, 5, 6]

1.) Start with the 1st element sorted:

[ 5 | 2, 9, 1, 5, 6]

$j = i - 1$ ← Key = arr[i] $j=5$, Key = 2

2) Pick 2, compare with 5, shift 5 to the right, insert 2.

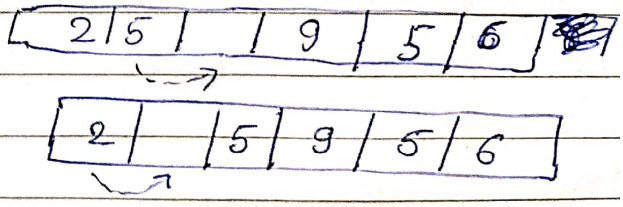
[ 2, 5 | 9, 1, 5, 6]

Key = 9 $j = 5$

3) Pick 9, no shifting needed, insert 9.

[ 2, 5, 9 | 1, 5, 6]

Key = 1 $j = 9$




4) Pick 1, compare with 9, 5, 2, shift all to the right, insert 1.

[ 1, 2, 5, 9 | 5, 6]

Key = 5, $j = 9$

5) Pick 5, compare with 9, shift 9 to the right, insert 5.

~~[ 1, 2, 5, 9 | 6 ]~~

[ 1, 2, 5, 5, 9 | 6 ]

Key = 6 j = 9

6) Pick 6, compare with 9, shift 9 to the right, insert 6.

[ 1, 2, 5, 5, 6, 9 ]

Now the array is sorted.

# Insertion Sort code.

```
public class InsertionSort {

    // Method to perform insertion sort on an array
    public static void insertionSort(int[] arr) {
        int len = arr.length;

        // Loop through each element in the array
        // Starting from the second element.
        for(int i = 1; i < len; i++) {
            // current element to be inserted.
            int key = arr[i];
            // index of previous element.
            int j = i - 1;

            // Shift elements of arr[0..i-1] that are
            // greater than key to 1 position ahead.
            while(j >= 0 && arr[j] > key) {
                arr[j+1] = arr[j];
                j--;
            }
            // Place the key at its correct position.
            arr[j+1] = key;
            printArray(arr); // Print array after each
                             // insertion.
        }
    }
}
```

```
// Method to print the elements of the array.
public static void printArray(int[] arr){
    for(int num : arr){
        System.out.print(num + " ");
    }
    System.out.println();
}

// Main method to test the insertion sort.
public static void main(String[] args){
    int[] arr = {5, 2, 9, 1, 5, 6};

    System.out.println("Array before Sorting:");
    printArray(arr);

    System.out.println("Array after Sorting:");
    insertionSort(arr);
  }
}
```

# Output

```
Array before Sorting:
5 2 9 1 5 6
Array after Sorting:
2 5 9 1 5 6
2 5 9 1 5 6
1 2 5 9 5 6
1 2 5 5 9 6
1 2 5 5 6 9
```

# Explanation

1. **Method Declaration**

- public static void insertionSort(int[] array) :
  This method takes an array of integers & sorts it in place using the insertion sort algorithm.

2. **Outer Loop**

- for(int i = 1; i < len; i++) :
  Starts from the second element (index 1) & iterates through the array.

3. **Key Element**

- int key = arr[i] ;
  The current element to be inserted into the sorted part of the array.

4. **Inner Loop**

- while( j >= 0 && array[j] > key ) :
  Compare the key with elements in the sorted part of the array, moving elements that are greater than the key one position to the right.

5. **Insert the Key**

- `arr[j+1] = key;`
  Inserts the key element into its correct position in the sorted part of the array.

6. **Main Method**

- Initializes an array, call the insertionSort method, & prints the sorted array.

Time complexity:

- Average case: $O(n^2)$
- Worst case: $O(n^2)$