SNS COLLEGE OF TECHNOLOGY
Coimbatore-37.
An Autonomous Institution

COURSE NAME : 23CAT602 - DATA STRUCTURES & ALGORITHMS

I YEAR/ I SEMESTER

UNIT – III SORTING & SEARCHING

Topic: Insertion Sort

Ms.B.Sumathi
Assistant Professor
Department of Computer Science and Engineering

INSERTION SORT

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

INSERTION SORT

Insertion Sort Algorithm

To sort an array of size N in ascending order iterate over the array and compare the current element (key) to its predecessor, if the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

INSERTION SORT

Working of Insertion Sort algorithm

Consider an example: arr[]: **{12, 11, 13, 5, 6}**

12 11 13 5 6

First Pass:

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

12 **11** 13 5 6 Here, 12 is greater than 11 hence they are not in the ascending order and 12 is not at its correct position.

Thus, swap 11 and 12.

So, for now 11 is stored in a sorted sub-array.

11 **12** 13 5 6

INSERTION SORT

Second Pass:

Now, move to the next two elements and compare them

11 **12** **13** 5 6 Here, 13 is greater than 12, thus both elements seems to be in ascending order, hence, no swapping will occur. 12 also stored in a sorted sub-array along with 11

INSERTION SORT

Third Pass:

Now, two elements are present in the sorted sub-array which are **11** and **12**

Moving forward to the next two elements which are 13 and 5

11 12 **13** **5** 6 Both 5 and 13 are not present at their correct place so swap them

11 12 **5** **13** 6 After swapping, elements 12 and 5 are not sorted, thus swap again

11 **5** **12** 13 6 Here, again 11 and 5 are not sorted, hence swap again

5 **11** 12 13 6

Here, 5 is at its correct position

INSERTION SORT

Fourth Pass:

Now, the elements which are present in the sorted sub-array are **5, 11** and **12**

Moving to the next two elements 13 and 6

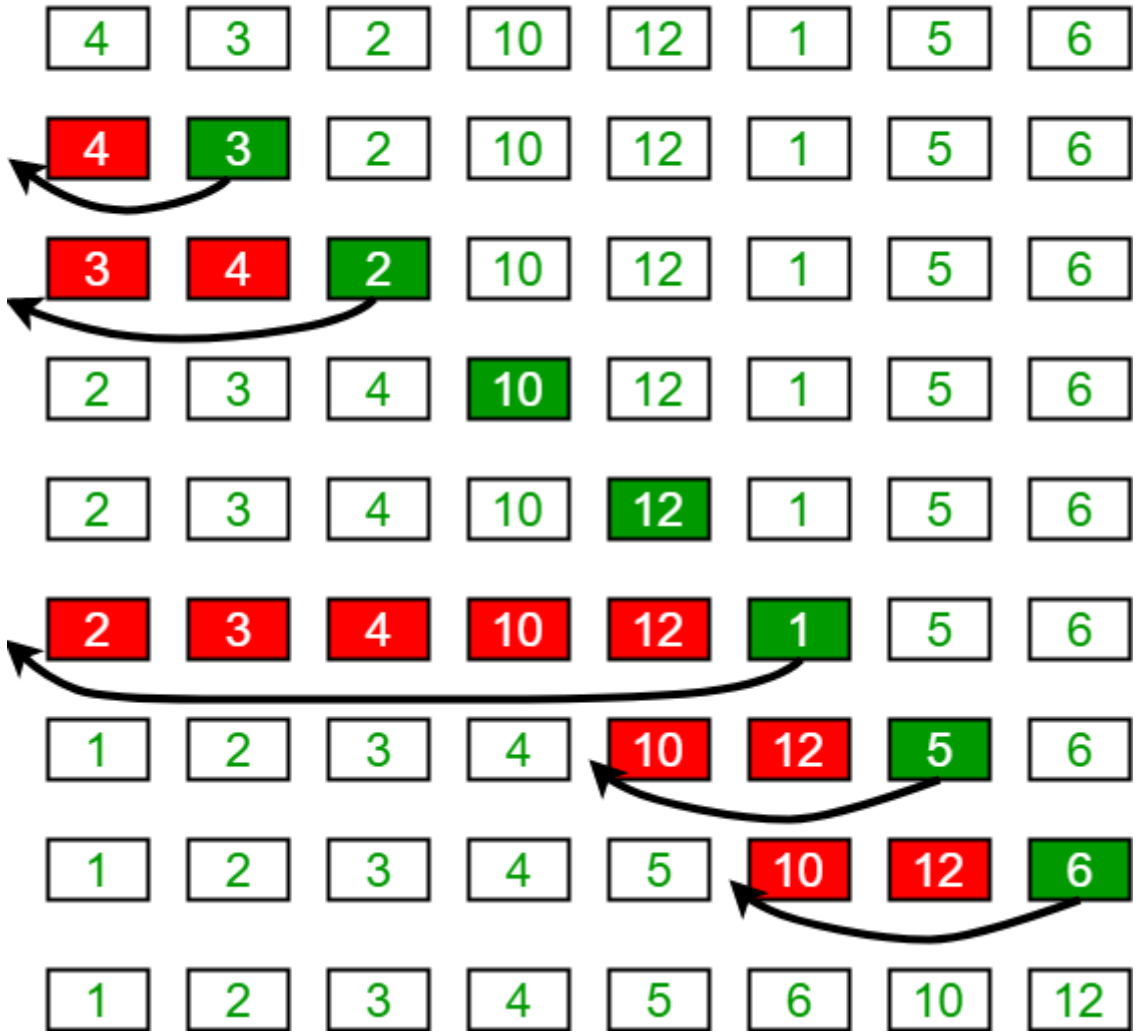
5 11 12 **13** **6** Clearly, they are not sorted, thus perform swap between both

5 11 12 **6** **13** Now, 6 is smaller than 12, hence, swap again

5 11 **6** **12** 13 Here, also swapping makes 11 and 6 unsorted hence, swap again

5 **6** **11** 12 13 Finally, the array is completely sorted.

Insertion Sort Execution Example



INSERTION SORT

Time Complexity: O(N^2)
Auxiliary Space: O(1)
Complexity Analysis of Insertion Sort:
Time Complexity of Insertion Sort
The **worst-case** time complexity of the Insertion sort is **O(N^2)**
The **average case** time complexity of the Insertion sort is **O(N^2)**
The time complexity of the **best case** is **O(N)**.
Space Complexity of Insertion Sort
The auxiliary space complexity of Insertion Sort is **O(1)**

INSERTION SORT

Characteristics of Insertion Sort

✓This algorithm is one of the simplest algorithms with a simple implementation
✓Basically, Insertion sort is efficient for small data values
✓Insertion sort is adaptive in nature, i.e. it is appropriate for data sets that are already partially sorted.

THANK YOU