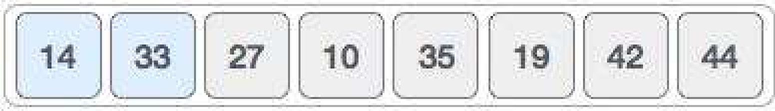
<u>**Lecture-25**</u>

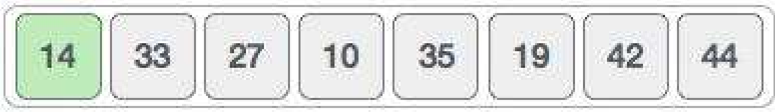
Insertion Sort

We take an unsorted array for our example.

Insertion sort compares the first two elements.



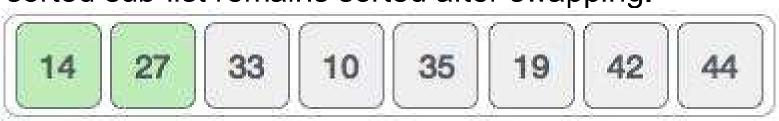
It finds that both 14 and 33 are already in ascending order. For now, 14 is in sorted sub-list.



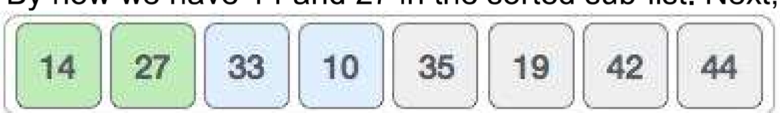
Insertion sort moves ahead and compares 33 with 27.

And finds that 33 is not in the correct position.

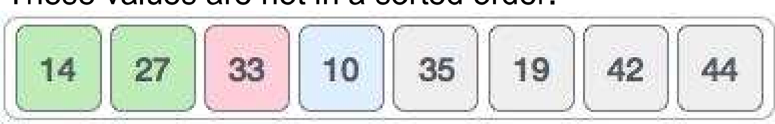
It swaps 33 with 27. It also checks with all the elements of sorted sub-list. Here we see that the sorted sub-list has only one element 14, and 27 is greater than 14. Hence, the sorted sub-list remains sorted after swapping.



By now we have 14 and 27 in the sorted sub-list. Next, it compares 33 with 10.

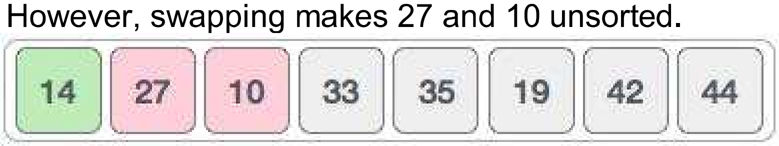


These values are not in a sorted order.

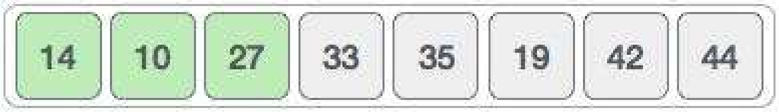


So we swap them.

However, swapping makes 27 and 10 unsorted.

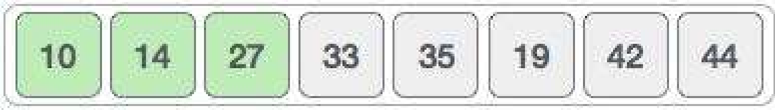


Hence, we swap them too.



Again we find 14 and 10 in an unsorted order.

We swap them again. By the end of third iteration, we have a sorted sub-list of 4 items.



This process goes on until all the unsorted values are covered in a sorted sub-list. Now we shall see some programming aspects of insertion sort.

Algorithm

Now we have a bigger picture of how this sorting technique works, so we can derive simple steps by which we can achieve insertion sort.

Step 1 − If it is the first element, it is already sorted. return 1;
Step 2 − Pick next element
Step 3 − Compare with all elements in the sorted sub-list
Step 4 − Shift all the elements in the sorted sub-list that is greater than the
value to be sorted
Step 5 − Insert the value
Step 6 − Repeat until list is sorted

Pseudocode

```
procedure insertionSort( A : array of items )
  int holePosition
  int valueToInsert
  for i = 1 to length(A) inclusive do:
    valueToInsert = A[i]
    holePosition = i

    while holePosition > 0 and A[holePosition-1] > valueToInsert do:
      A[holePosition] = A[holePosition-1]
      holePosition = holePosition -1
    end while
    A[holePosition] = valueToInsert

  end for
         end procedure
```