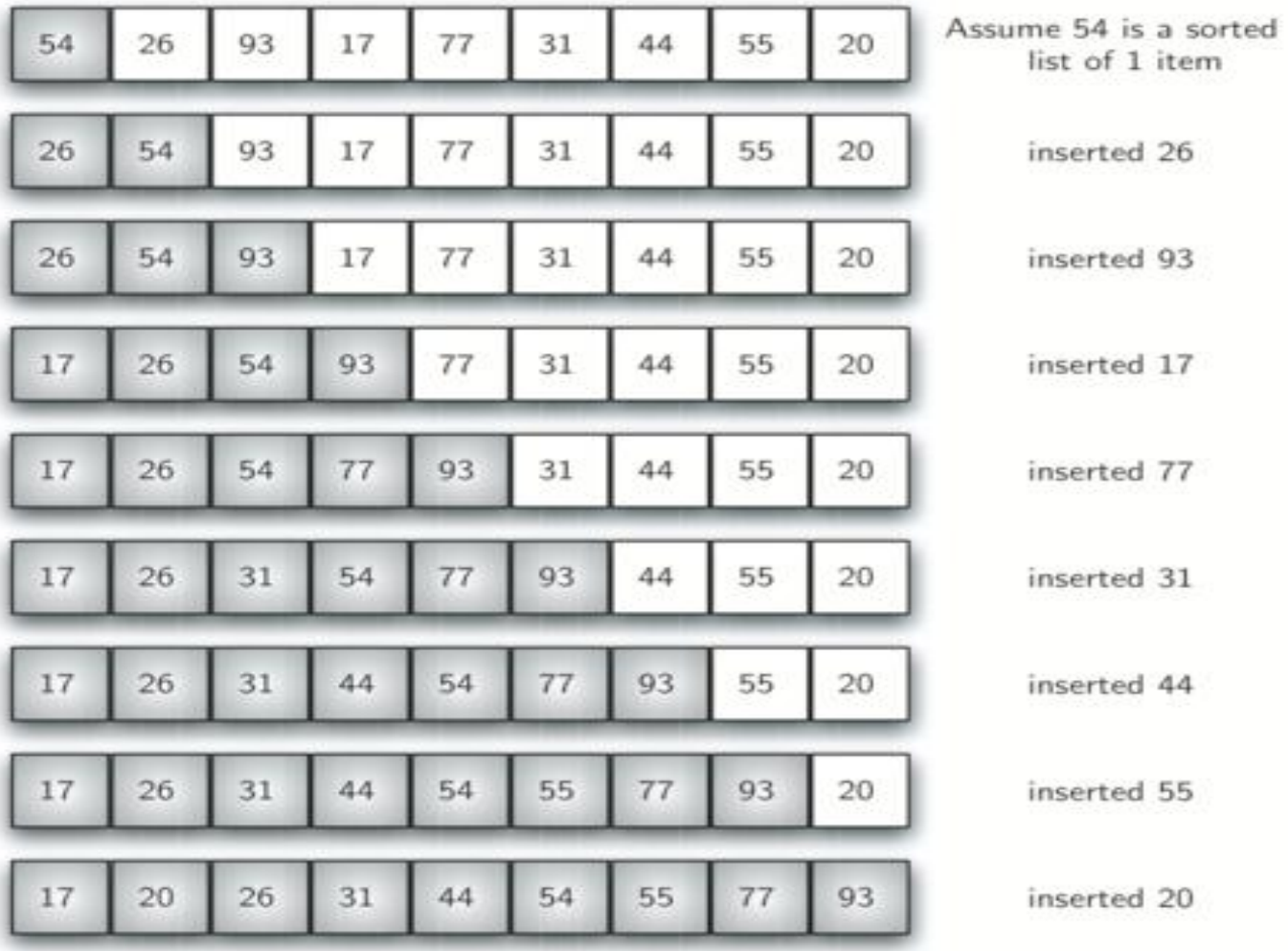
Sorting

Unit -III

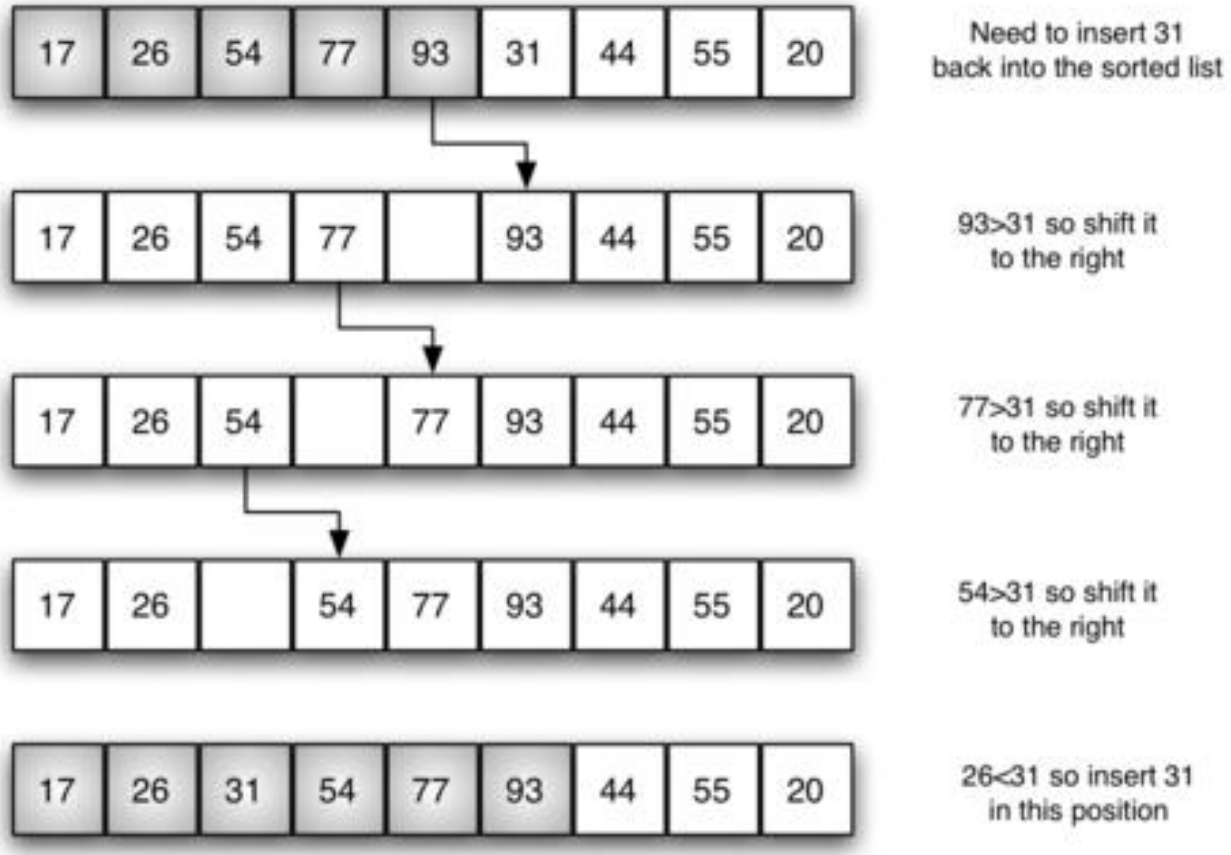
Insertion Sort

- It is a comparison-based sorting algorithm
- A sub-list is maintained and always sorted
- The lower part of an array is maintained as sub-list and sorted
- An element to be inserted in the sorted sub-list should find an appropriate place and it is inserted there – hence called “Insertion Sort”

Example – Insertion Sort



Process for inserting an element



Time Complexity of Insertion Sort

- Best case O(n)
- Ex. After inserting 11 in the sorted array [2,3,5,7,11], we found no slide or inversion in sorted sub-list which takes constant time O(1), So totally O(n)

- Worst case $O(n^2)$
- Ex. The array is in *reverse* sorted order, such as [11, 7, 5, 3, 2]. Here reversing need O(n) and traversing the entire list O(n), so totally $O(n^2)$

Insertion Sort Program

```
def insertionSort(mylist):
   for i in range(1,len(mylist)):

      currentvalue = mylist[i]
      position = i

      while position>0 and alist[position-1]>currentvalue:
         mylist[position]=mylist[position-1]
         position = position-1

      mylist[position]=currentvalue

mylist = [54,26,93,17,77,31,44,55,20]
insertionSort(mylist)
print(mylist)
```

Selection Sort

- It is also a comparison-based algorithm
- List is divided into two parts, left end sorted part and the right end unsorted part
- Initially, the sorted part is empty and the unsorted part is the entire list

Contd...

- The smallest element is selected from the unsorted array and swapped with the leftmost element
- That element becomes a part of the sorted array
- This process continues increasing the unsorted array size by one element to the right

Example

- [45, 12, 56, 11, 35] min is 11, so swapped with first
- [11, 12, 56, 45, 35] 11 is in sorted sub-list
- [11, 12, 56, 45, 35] 12 is next min, so added to sorted sub-list
- [11, 12, 56, 45, 35] 35 is next min, so swapped to element in 3rd position
- [11, 12, 35, 45, 56] 45 is the next min, so no swapping
- [11, 12, 35, 45, 56] Final sorted list

Selection Sort Program

```
a = [16, 19, 11, 15, 10, 12, 14]
i = 0
while i<len(a):
  smallest = min(a[i:])    #smallest element in the sublist
  smallestindex = a.index(smallest)   #index of smallest element
  a[i],a[smallestindex] = a[smallestindex], a[i]
  i=i+1
print (a)
```

Time Complexity of Selection Sort

- $O(n^2)$
- As both inner and outer loop is executed ‘n’ times