Data Structure

Trainer : Nisha Dingare

Email : nisha.dingare@sunbeaminfo.com

Introduction :

- Data structure is a form of Organising, Processing and Storing the data in the memory.
- There is a need of Data structures in programming to achieve efficiency in operations like addition, deletion, traversal, searching, sorting, etc.

- Linear/Basic Data Structures :
  1. Array
  2. Linked List
  3. Stack
  4. Queue
  5. Hash Tables

- Non-linear/Advanced Data Structures :
  1. Tree
  2. Graph

Algorithm and Program :

- **What is an Algorithm?**
    - An algorithm is a finite set of instructions written in human understandable language (like english), if followed, accomplishes a given task. It is also called as pseudocode.

- **What is a Program?**
    - A program is a set of instructions written in any programming language (like C, C++, Java, Python, Assembly etc...) given to the machine to do specific task.

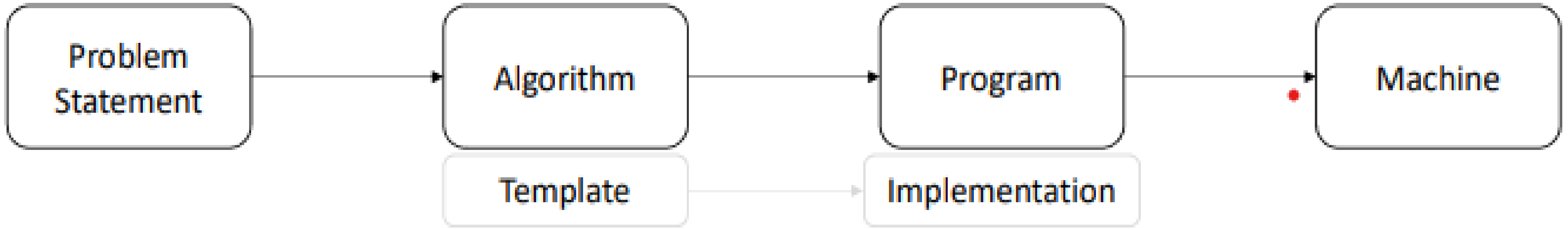
An algorithm is a template whereas a program is an implementation of an algorithm.

Algorithm :

- **Algorithm**
  - is a clearly specified set of simple instructions.
  - is a solution to solve a problem.
  - is written in human understandable language.

  - Algorithm is also referred as “pseudo code”.

One problem statement has multiple solutions, out of which we need to select efficient one. Hence we need to do analysis of an algorithm.

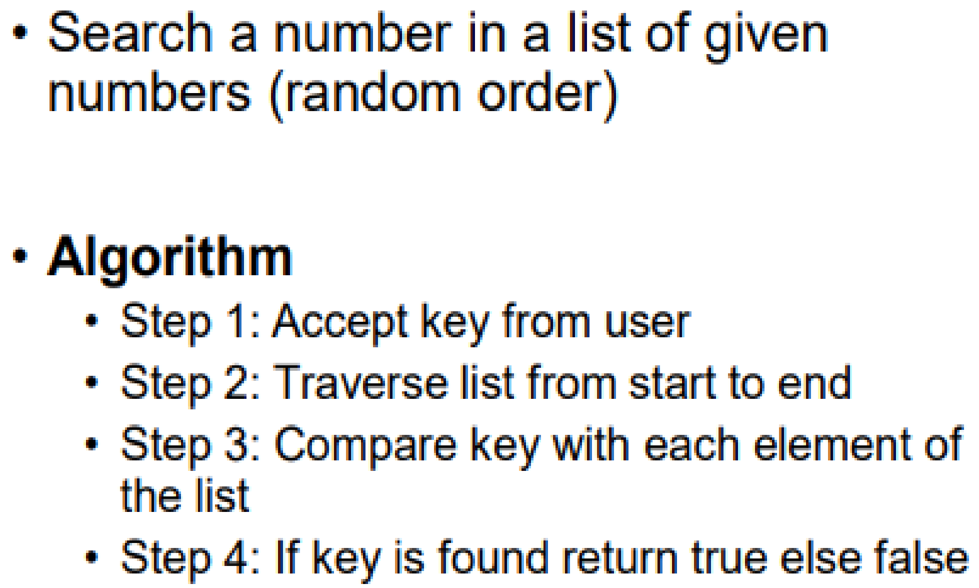


e.g. Write an algorithm to find sum of all array elements.
Algorithm:
Step 1: Initialize sum =0
Step 2: Traverse array form index 0 to N-1
Step 3: Add each element in the sum variable
Step 4: Return the final sum

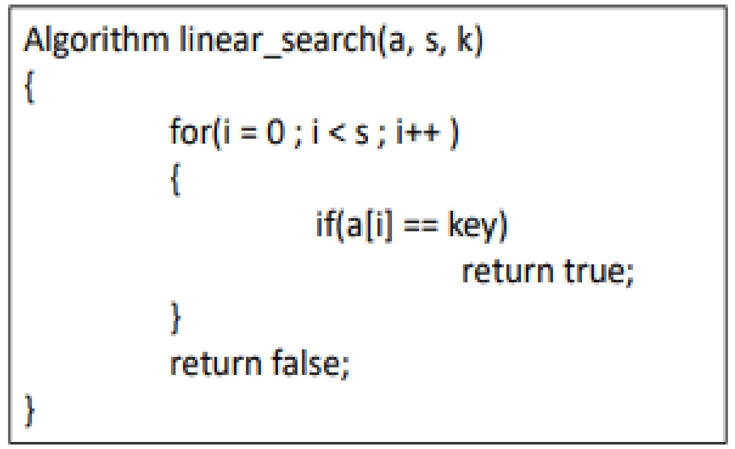
```
Algorithm SumArray(array, size) {
        sum = 0;
        For(index = 0 ; index < size ; index++)
                sum += array[index];
        return sum;
}
```

Searching Algorithms : Linear Search

- **Linear Search** : This is an algorithm to find the given number from the list of elements of an array, in a linear order, starting from the 0th index.
- In this algorithm, key element gets compared sequentially with each array element by traversing it from first element till either match is found or maximum till the last element.



- Search a number in a list of given numbers (random order)
- **Algorithm**
  - Step 1: Accept key from user
  - Step 2: Traverse list from start to end
  - Step 3: Compare key with each element of the list
  - Step 4: If key is found return true else false



```
Algorithm linear_search(a, s, k)
{
        for(i = 0 ; i < s ; i++ )
        {
                if(a[i] == key)
                        return true;
        }
        return false;
}
```

Searching Algorithms : Binary Search :

- This algorithm follows divide-and-conquer approach.
- To apply binary search on an array prerequisite is that array elements must be in a sorted manner.
- In this algorithm, in first iteration, by means of calculating mid position big size array gets divided into two subarray’s, **left subarray** and **right subarray**. Then key element gets compared with element which is at mid position.
- If key is not found in the first iteration then either it will get searched into left subarray or into the right subarray by applying same logic repeatedly till either key is not found or till max the size of any subarray is valid.

Binary Search Algorithm :

- Given an integer x and integers A0, A1, …An-1, which are pre-sorted and already in memory, find i such that Ai = x or return i = -1 if x is not in the input

- **Algorithm**
  - Step 1: Accept key from user
  - Step 2: Check if x is the middle element. If so x is found at mid
  - Step 3: If x is smaller than the middle element, apply same strategy to the sorted subarray to the left of middle element.
  - Step 4: If x is larger than the middle element, apply same strategy to the sorted subarray to the right of middle element

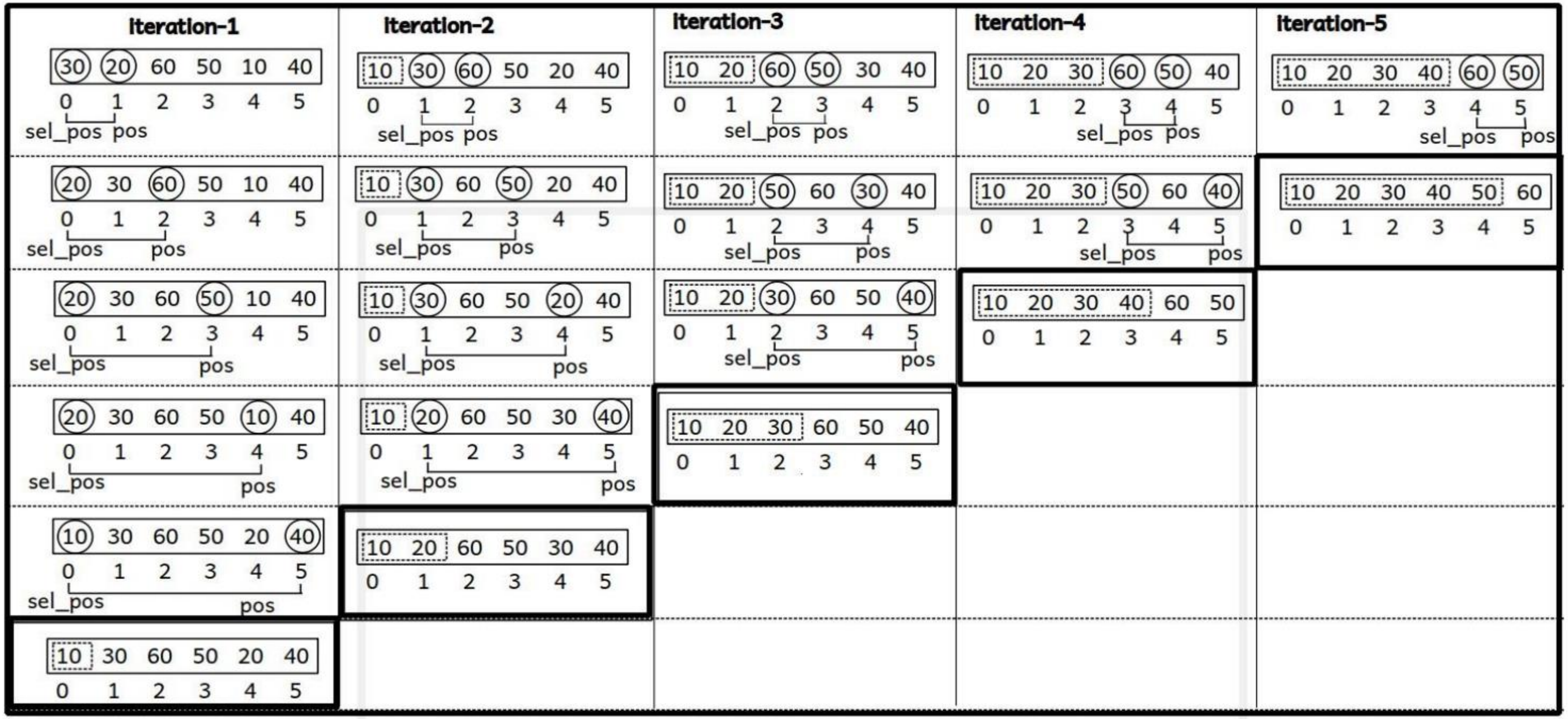
Sorting Algorithms : Selection Sort :

- Sorting : Arranging data elements either in ascending or descending order. By default sorting takes place in ascending order.

1. **Selection Sort** :
   - In this algorithm, in first iteration, first position gets selected and element which is at selected position gets compared with all the elements after that. If selected position element found greater than any other position element then swapping takes place. At the end of first iteration smallest element gets settled at first position.

   - In the second iteration, second position gets selected and element which is at selected position gets compared with all elements after that. If selected position element found greater than any other position element then swapping takes place. At the end of second iteration second smallest element gets settled at second position, and so on in maximum (n-1) no. of iterations all array elements gets arranged in a sorted manner.

Selection Sort :



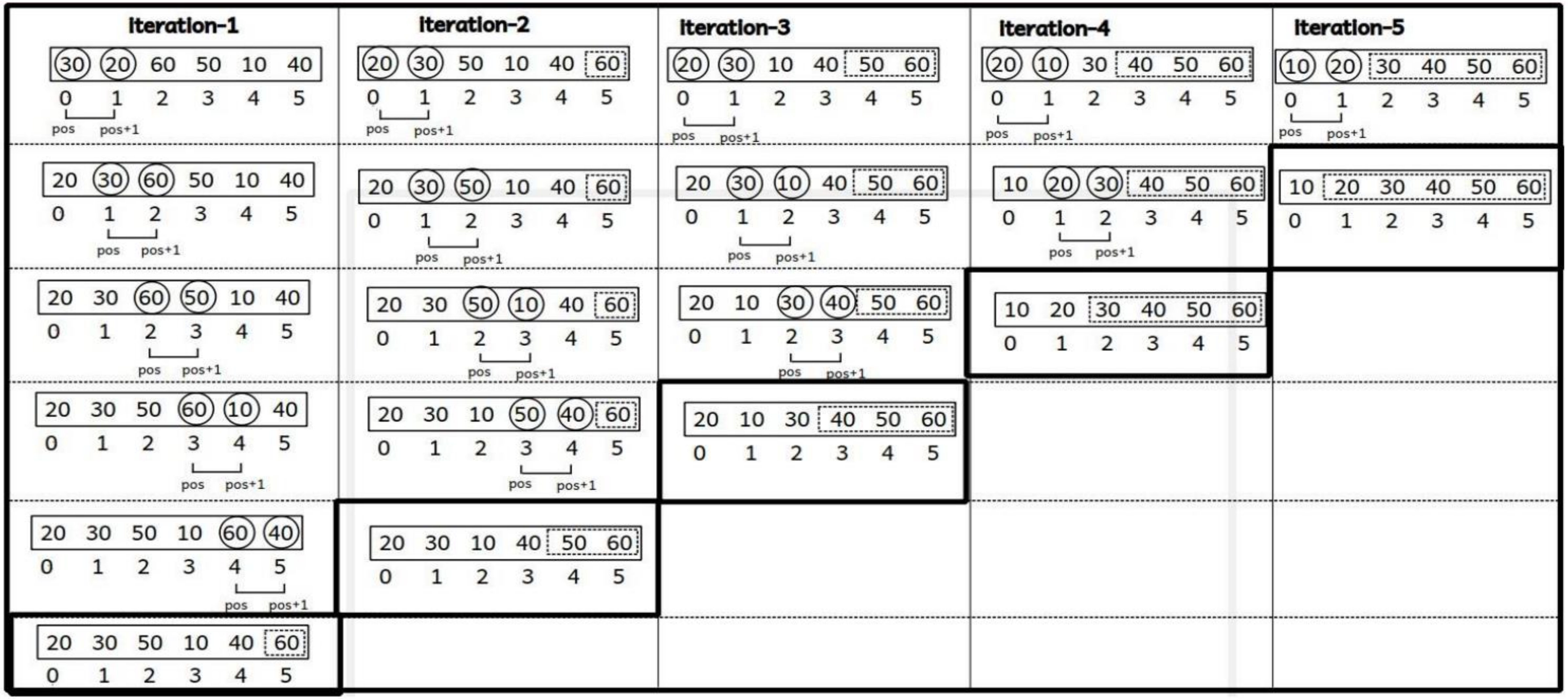
Sorting Algorithms : Bubble Sort :

2. Bubble Sort :

- In this algorithm, in every iteration, elements which are at two consecutive positions gets compared. If they are already in order then no need of swapping them, but if they are not in order i.e. if previous position element is greater than its next position element then swapping takes place.

- By this logic in first iteration largest element gets settled at last position, in second iteration second largest element gets settled at second last position and so on, in max (n-1) no. of iterations all elements gets arranged in a sorted manner.

Bubble Sort :



Iteration-1
Iteration-2
Iteration-3
Iteration-4
Iteration-5
pos
pos+1

Sorting Algorithms : Insertion Sort :

- In this algorithm, in every iteration one element gets selected as a key element and key element gets inserted into an array at its appropriate position in a such a way that elements which are at left side are arranged in a sorted manner, and so on. In max (n-1) no. of iterations all array elements gets arranged in a sorted manner.

- This algorithm works efficiently for already sorted input sequence by design .

- Insertion sort algorithm is an efficient algorithm for smaller input size array

Thank You!