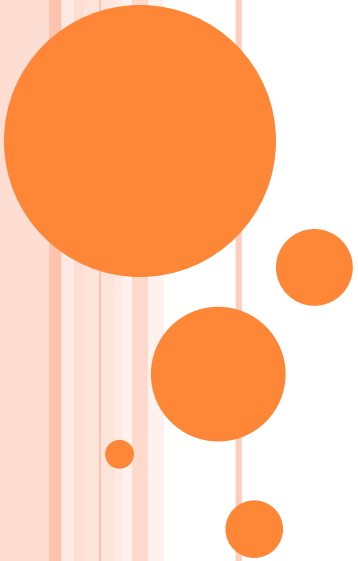
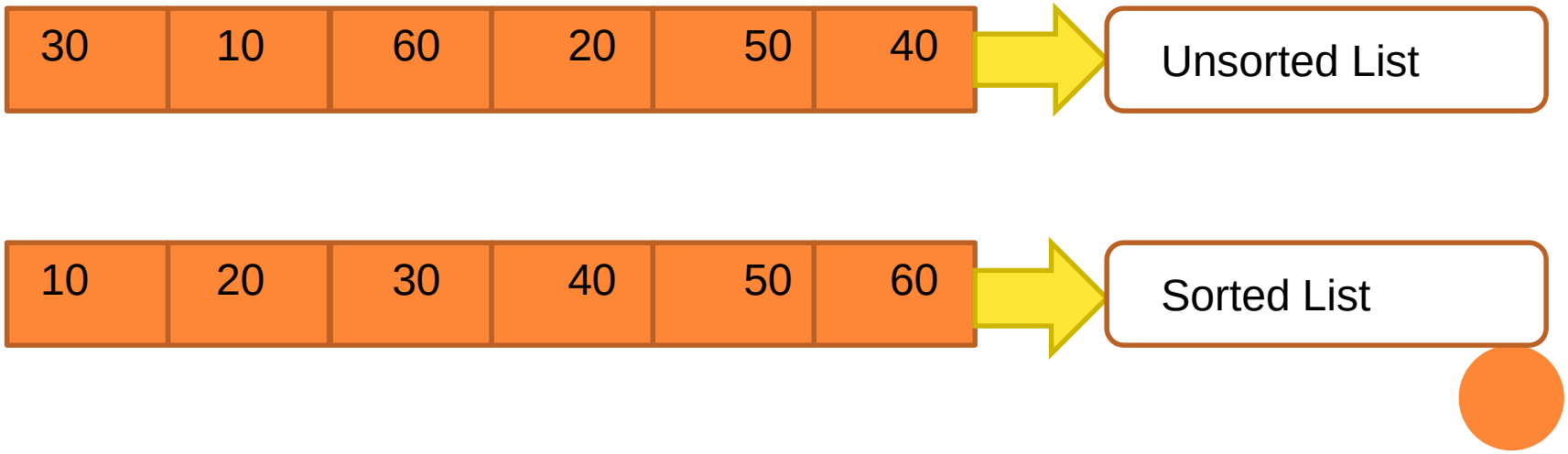
Sorting And Its Types



Sorting

- Sorting refers to operations of arranging a set of data in a given order.



Basic Types :

- Internal Sorting:

If all the data to be sorted can be adjusted in main memory then it is called as Internal Sorting.

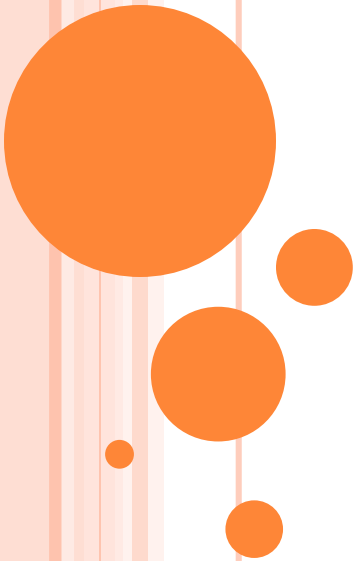
- External Sorting:

If data to be stored is large and aquires external memory then the type is called as External Sorting.

METHODS OF SORTING:

- Bubble Sort
- Selection Sort
- Insertion Sort

Bubble Sort



Algorithm

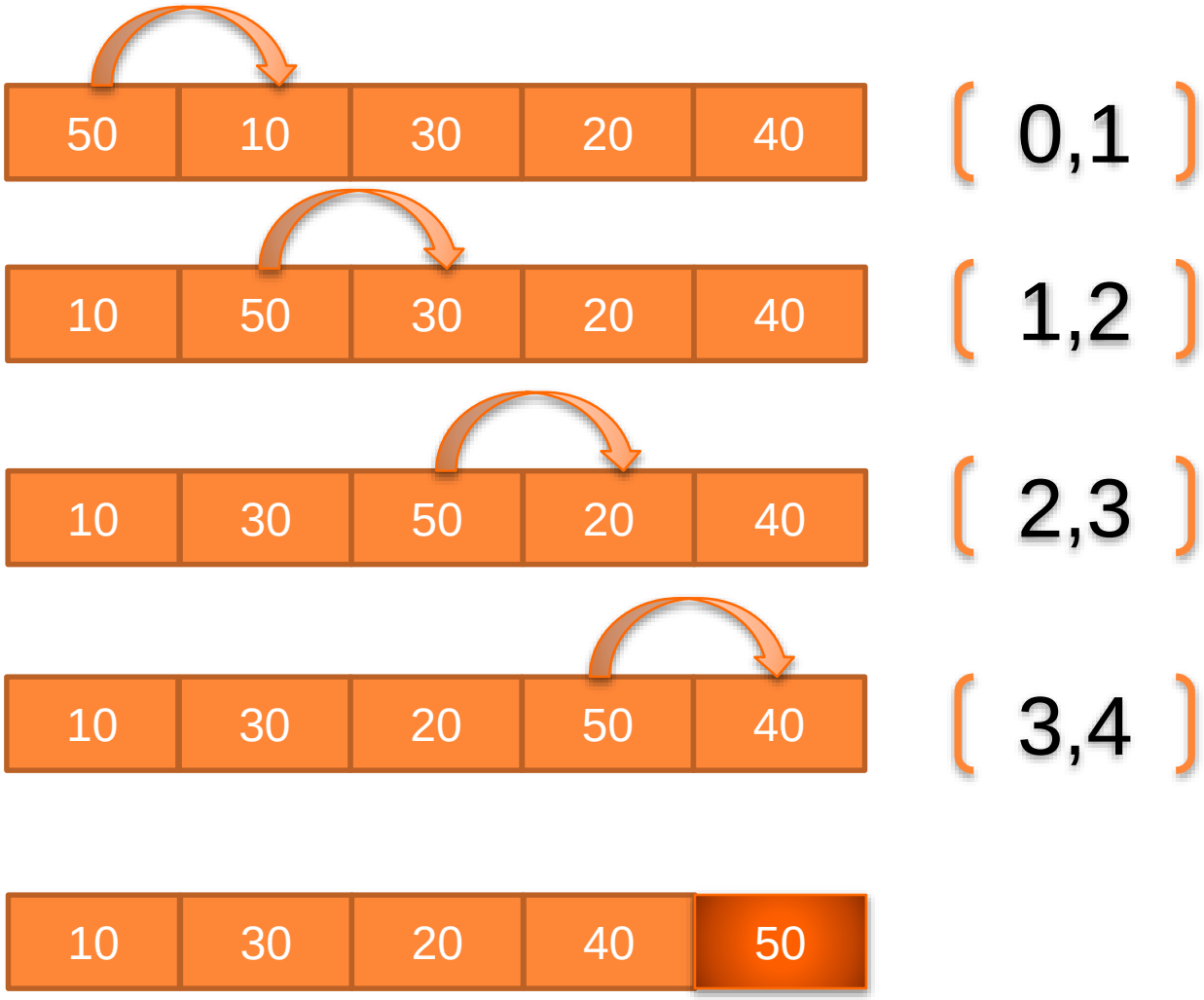
- Bubble Sort:

Algorithm of bubble sort includes two steps repeated until the list is sorted.

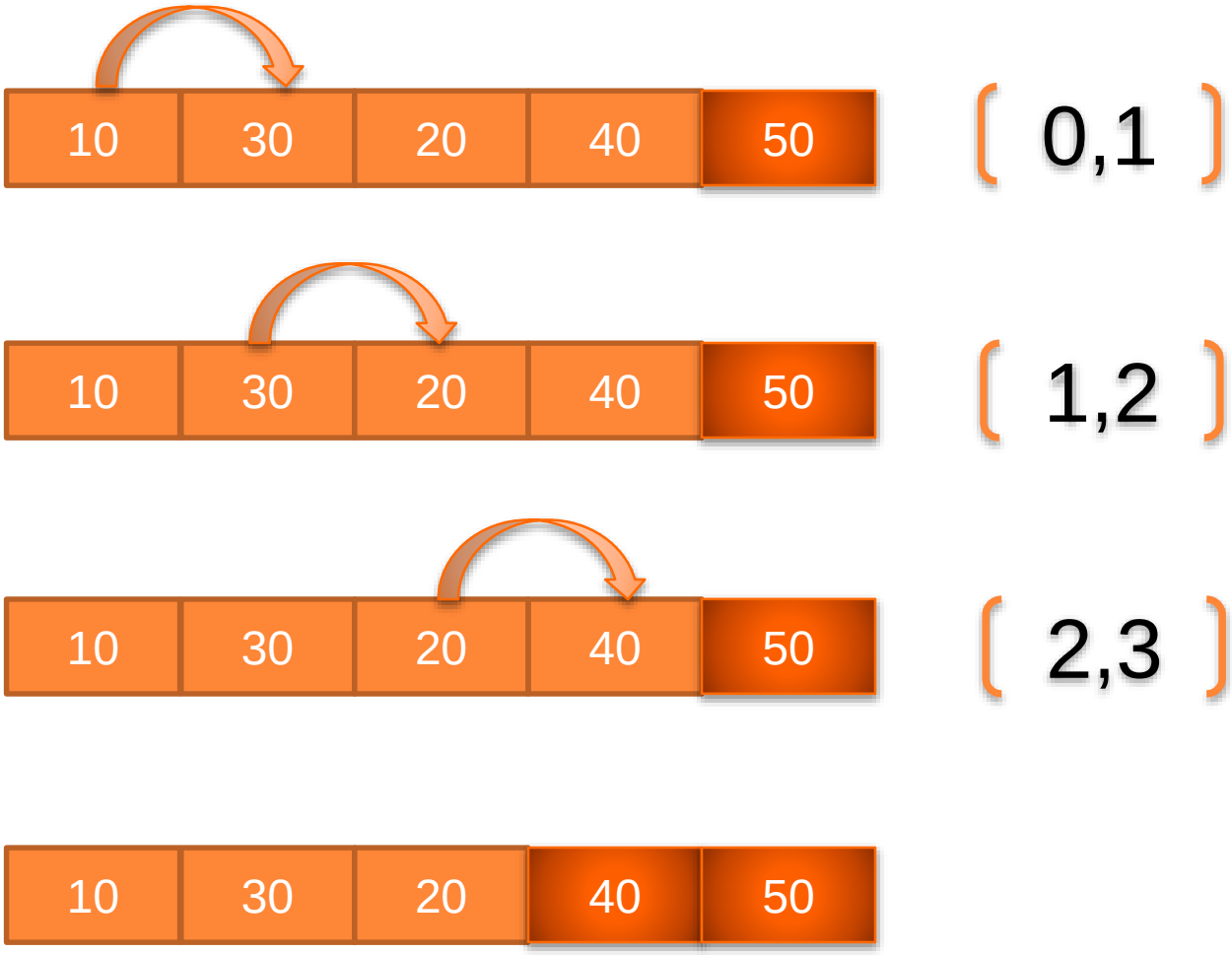
- Compare adjacent elements, if the element on right side is smaller then swap their positions.
- Compare first element, second element and so on on completion of Pass 1 the largest element is at last position.

Pass 1:

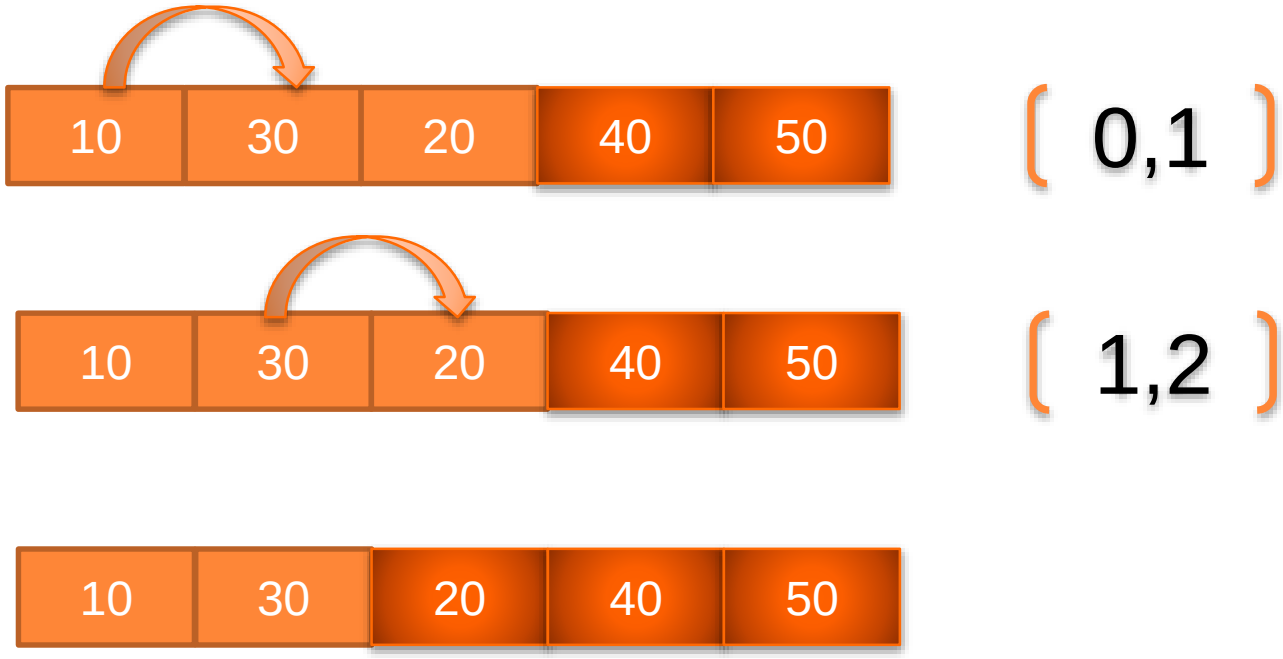
Number Of Passes = Max – 1 = 5 – 1 = 4



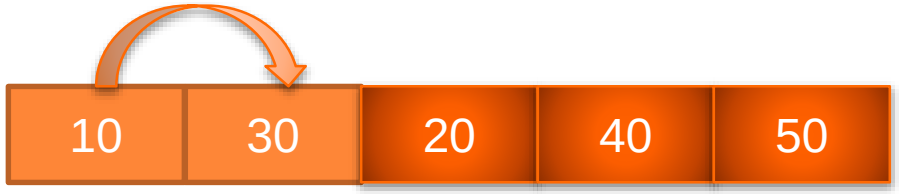
Pass 2:



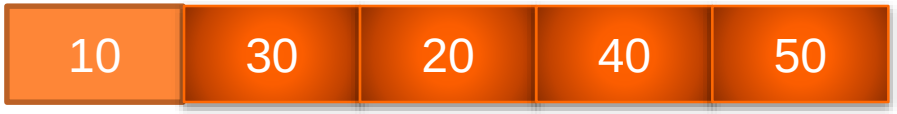
Pass 3:



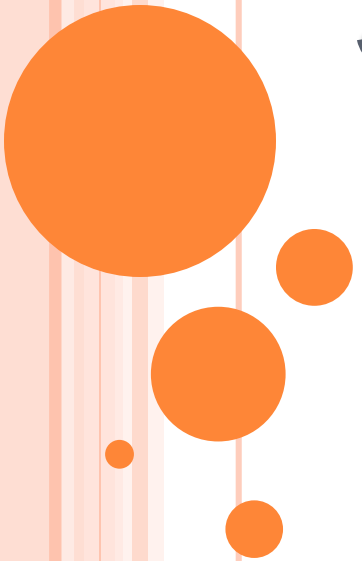
Pass 4:



(0,1)



Selection Sort

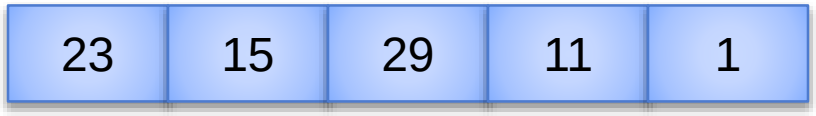


Algorithm

- Selection Sort:

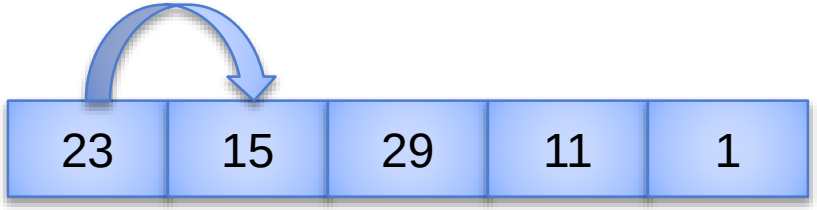
Here in selection sort the algorithm depends on the zeroth element majorly.

- The Zeroth element is compared with the first element and if the element at right is found smaller then their positions are swapped or exchanged.
- The same procedure is carried with all elements of the list resulting into a fully sorted list.

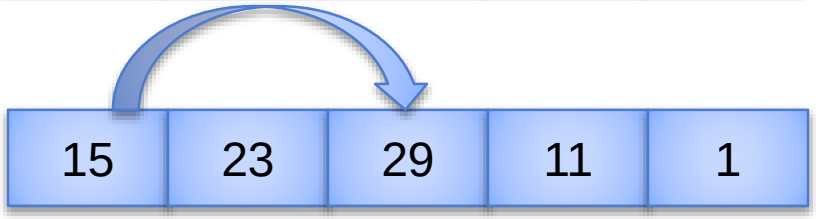


Pass 1:

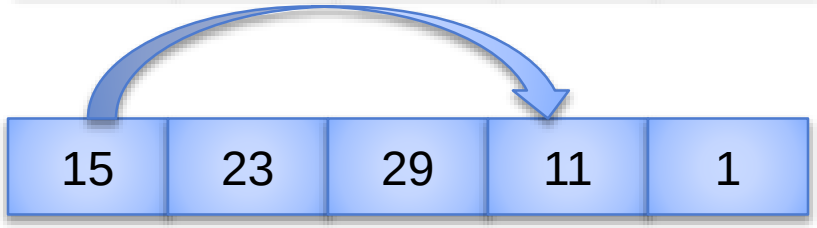
Number Of Passes = Max – 1 = 5 – 1 = 4



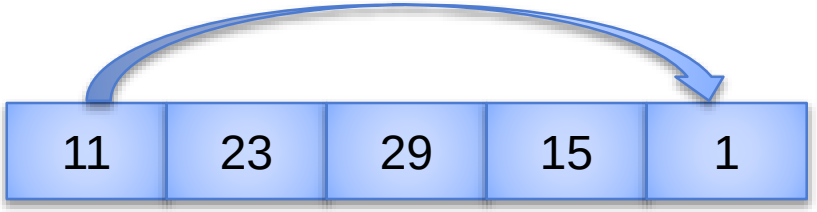
0,1



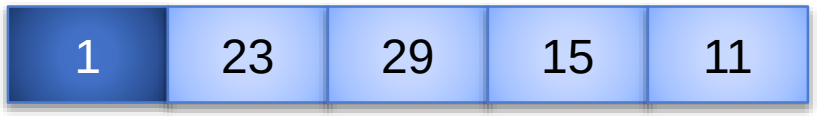
0,2



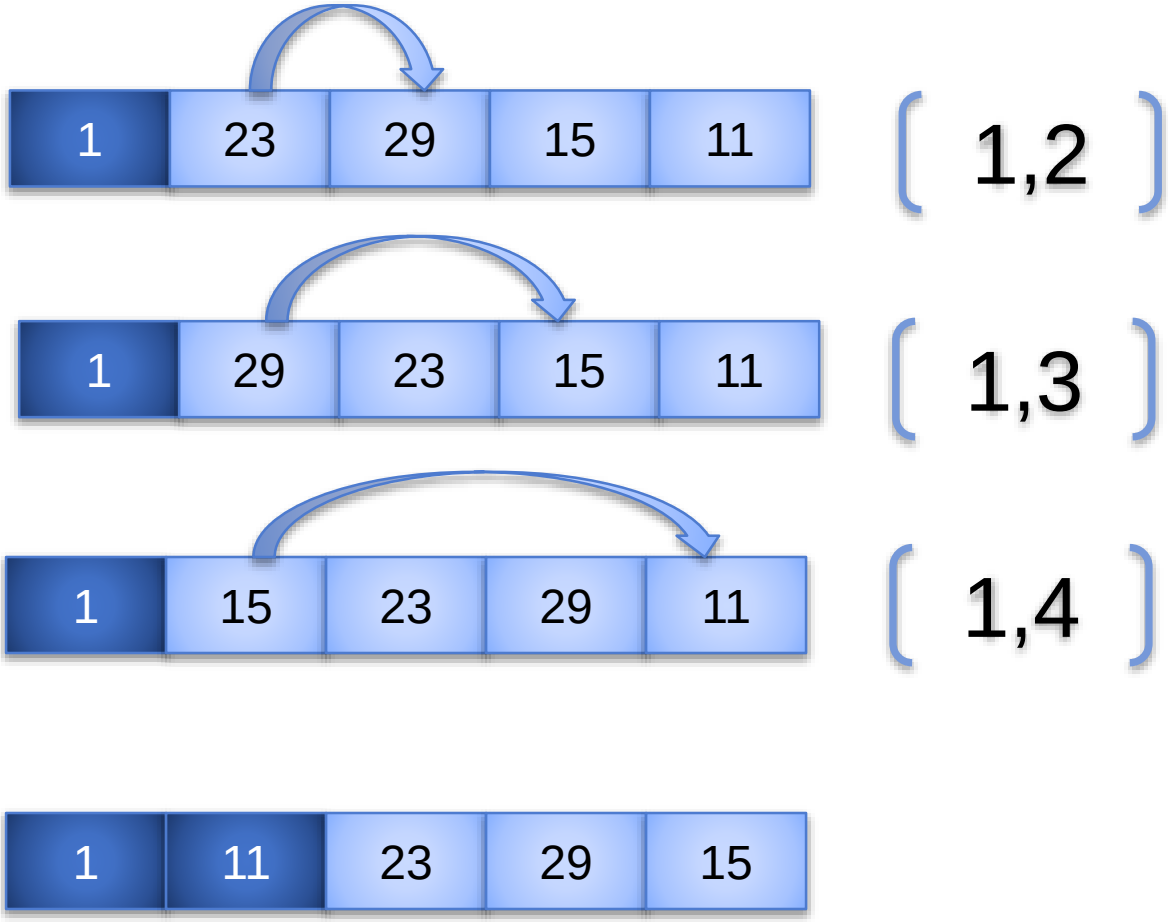
0,3



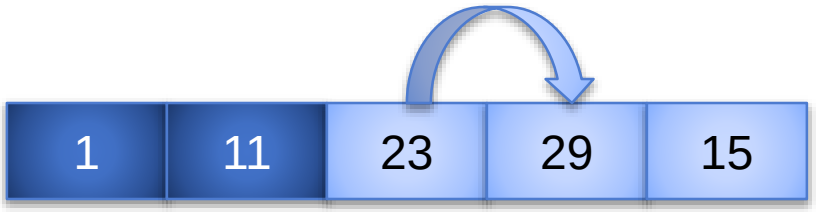
0,4



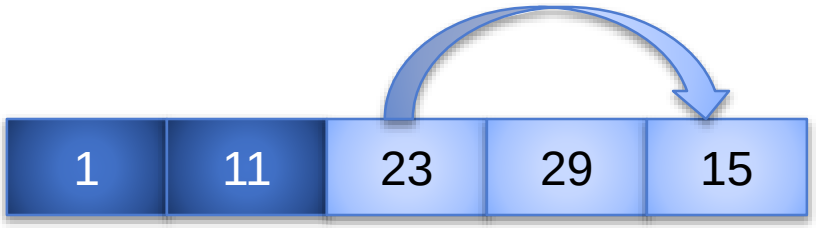
Pass 2:



Pass 3:

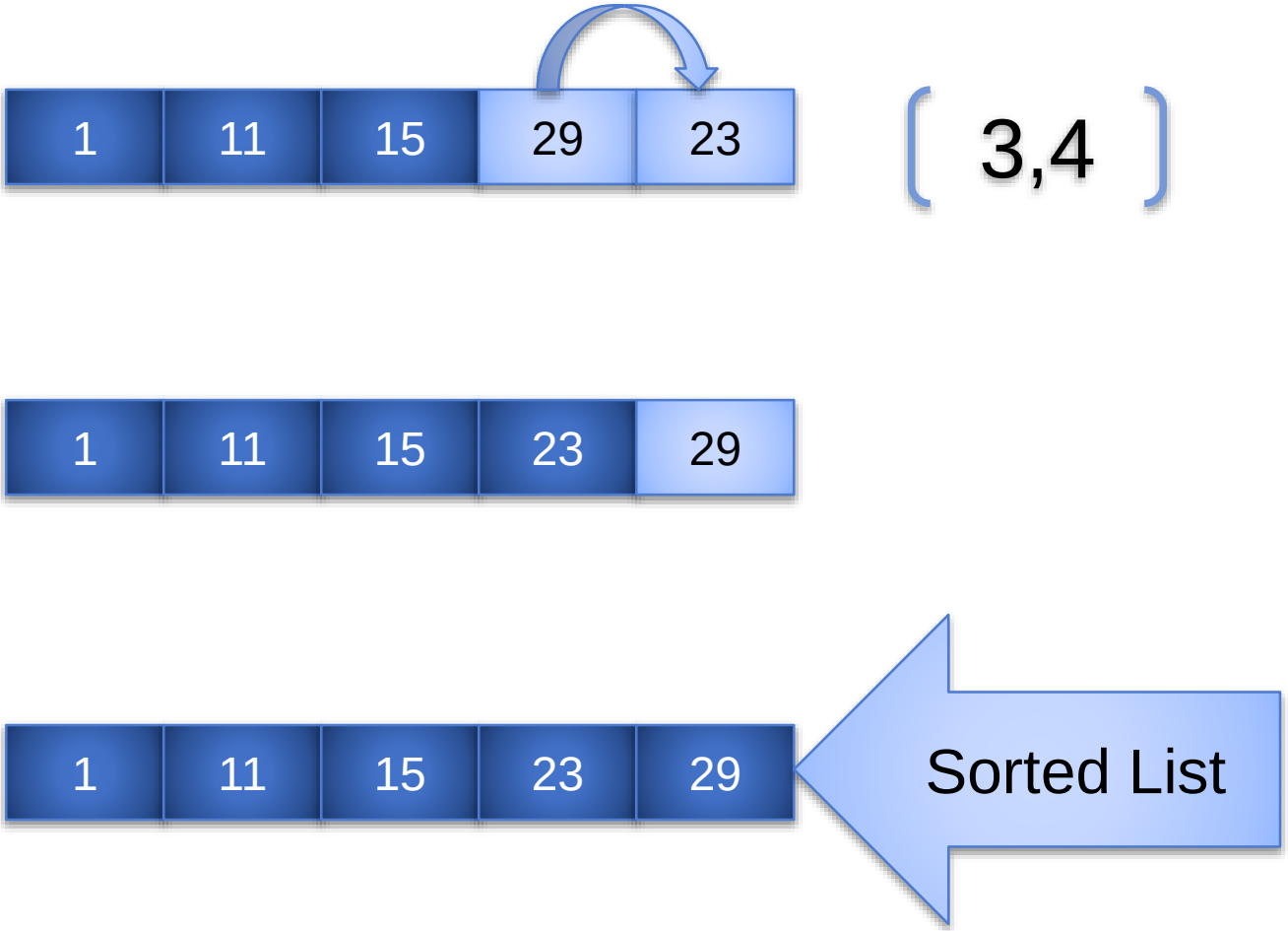


(2,3)

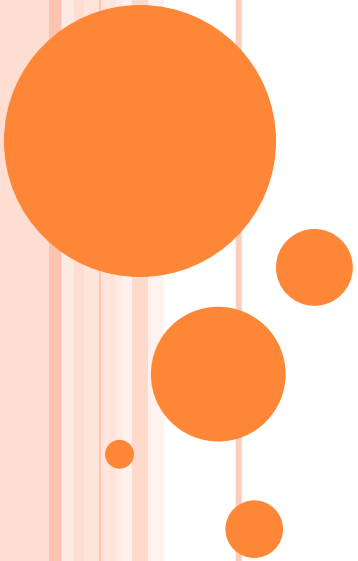


(2,4)

Pass 4:



Insertion Sort



- **when we can use Insertion Sort ?**

This method is effective when dealing with small numbers .

- **Applications using insertion sort**

Mathematical applications : in the search for greater value, or the smallest value. In many other applications.

Algorithm

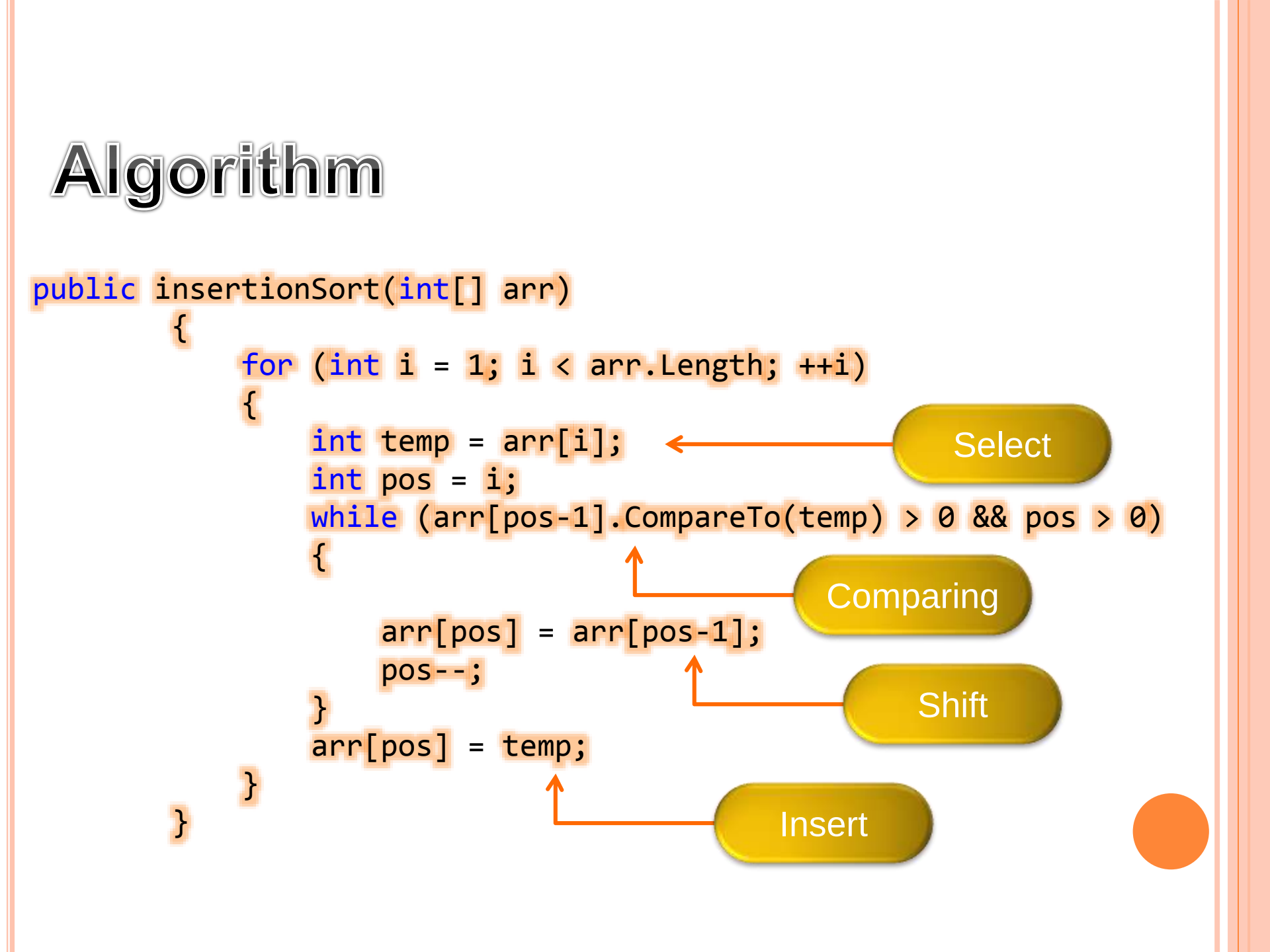
- Insertion Sort

In insertion sort the elements are compared and inserted to respective index place.

- It starts with comparision of 1^{st} and 0^{th} element in pass 1.
- In pass 2 the second element is compared with the 1^{st} and 0^{th} element.
- Doing so with all the elements in the list appropriate element is inserted by shifting elements on right.

Algorithm

```
public insertionSort(int[] arr)
        {
            for (int i = 1; i < arr.Length; ++i)
            {
                int temp = arr[i];
                int pos = i;
                while (arr[pos-1].CompareTo(temp) > 0 && pos > 0)
                {
                    arr[pos] = arr[pos-1];
                    pos--;
                }
                arr[pos] = temp;
            }
        }
```



Best And Worst Case

Best Case

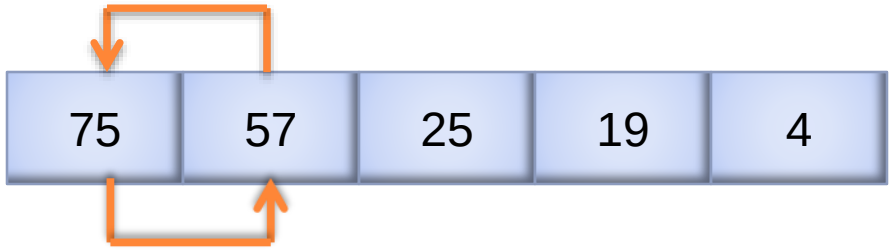
The best case input is an array that is already sorted. In this case insertion sort has a linear running time (i.e., $\Theta(n)$). During each iteration, the first remaining element of the input is only compared with the right-most element of the sorted subsection of the array.

Worst Case

The worst case input is an array sorted in reverse order. In this case every iteration of the inner loop will scan and shift the entire sorted subsection of the array before inserting the next element. For this case insertion sort has a quadratic running time (i.e., $O(n^2)$).

75	57	25	19	4

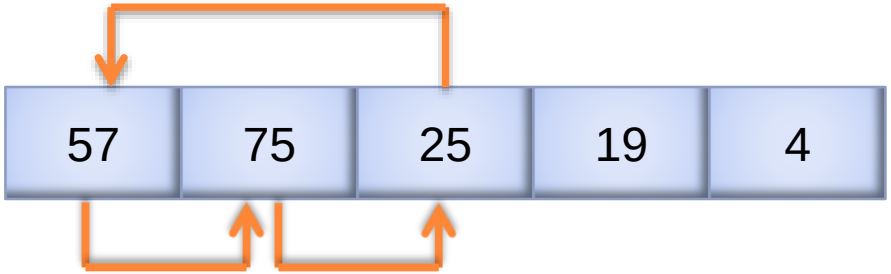
Pass 1: ⇨ Number Of Passes = Max – 1 = 5 – 1 = 4



(0,1)

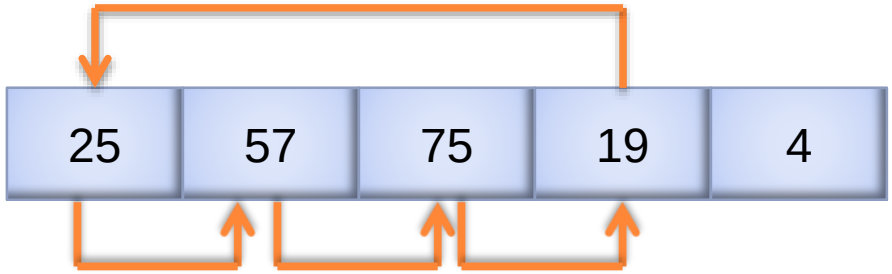
57	75	25	19	4

Pass 2:



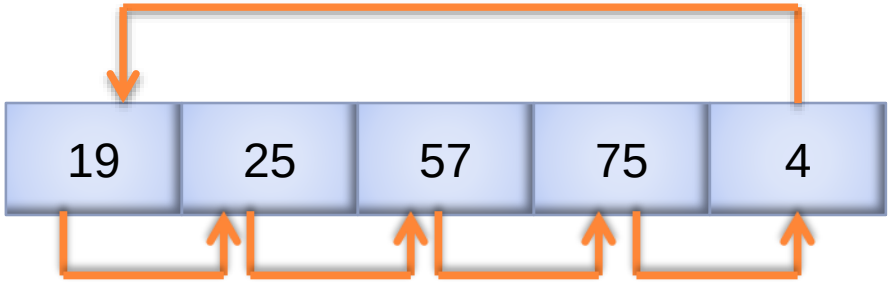
0,2

Pass 3:



0,3

Pass 4:



0,4

Thank You…!