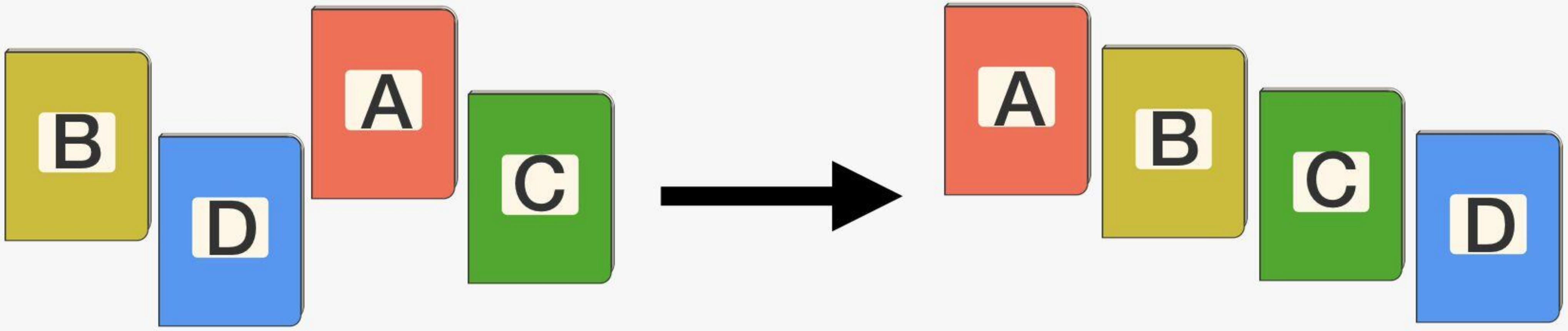
BML MUNJAL UNIVERSITY™

Data Structures & Algorithms

Sorting Algorithms



Sorting

Reordering a set of data into a particular order(either in ascending or descending order).

There are several algorithms that solve the sorting problem:

- Selection Sort, Bubble Sort, Insertion Sort,
- Merge Sort, Quick Sort, Heap sort

In this course we will discuss few internal sorting techniques.

Internal sorting takes place in the main memory of a computer.

Assume that the array contains distinct elements.

Assume that we want to sort an array in ascending order sorted sequence.

Bubble Sort

Idea behind Bubble Sort

The bubble sort makes multiple passes through a list.

In each pass, it compares the adjacent elements in the list and exchanges those that are not in order.

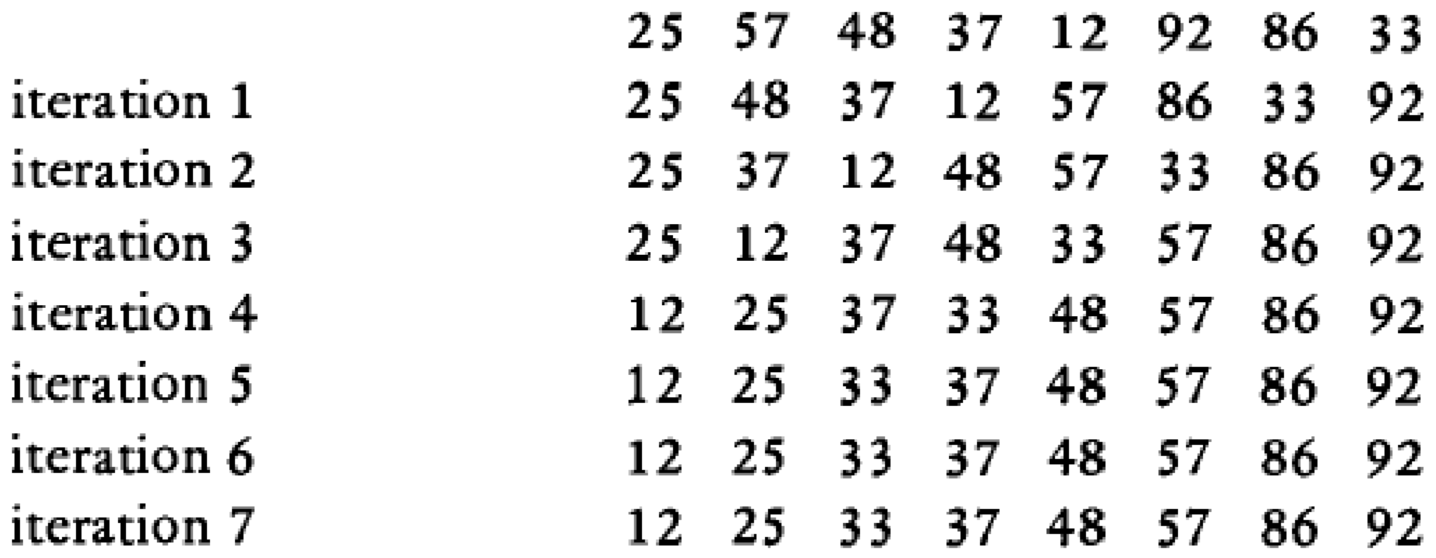
After the first pass, the largest element is in its proper position within the array. After the second pass, the second largest element is in its proper position within the array, …, and so on.

In general, A[n-i] will be in its proper position after iteration i. Given a list of n elements, bubble sort requires up to n-1 passes to sort the data.

Bubble Sort

```
Bubble_Sort( A[], n)
{ // Input : Unsorted array of size n
   for (i=0; i<n-1; i++)
      for (j=0; j<n-i-1; j++)
         if ( A[j]> A[j+1])
            swap ( A[j], A[j + 1]);
// Output :  Array of size n, sorted in ascending order
}
```

Bubble Sort



	25	57	48	37	12	92	86	33
iteration 1	25	48	37	12	57	86	33	92
iteration 2	25	37	12	48	57	33	86	92
iteration 3	25	12	37	48	33	57	86	92
iteration 4	12	25	37	33	48	57	86	92
iteration 5	12	25	33	37	48	57	86	92
iteration 6	12	25	33	37	48	57	86	92
iteration 7	12	25	33	37	48	57	86	92

Analysis of Bubble Sort

Worst case analysis

$T(n) = (n-1)+(n-2)+\ldots\ldots\ldots+1$

$= [n(n-1)]/2$

$= O(n^2)$

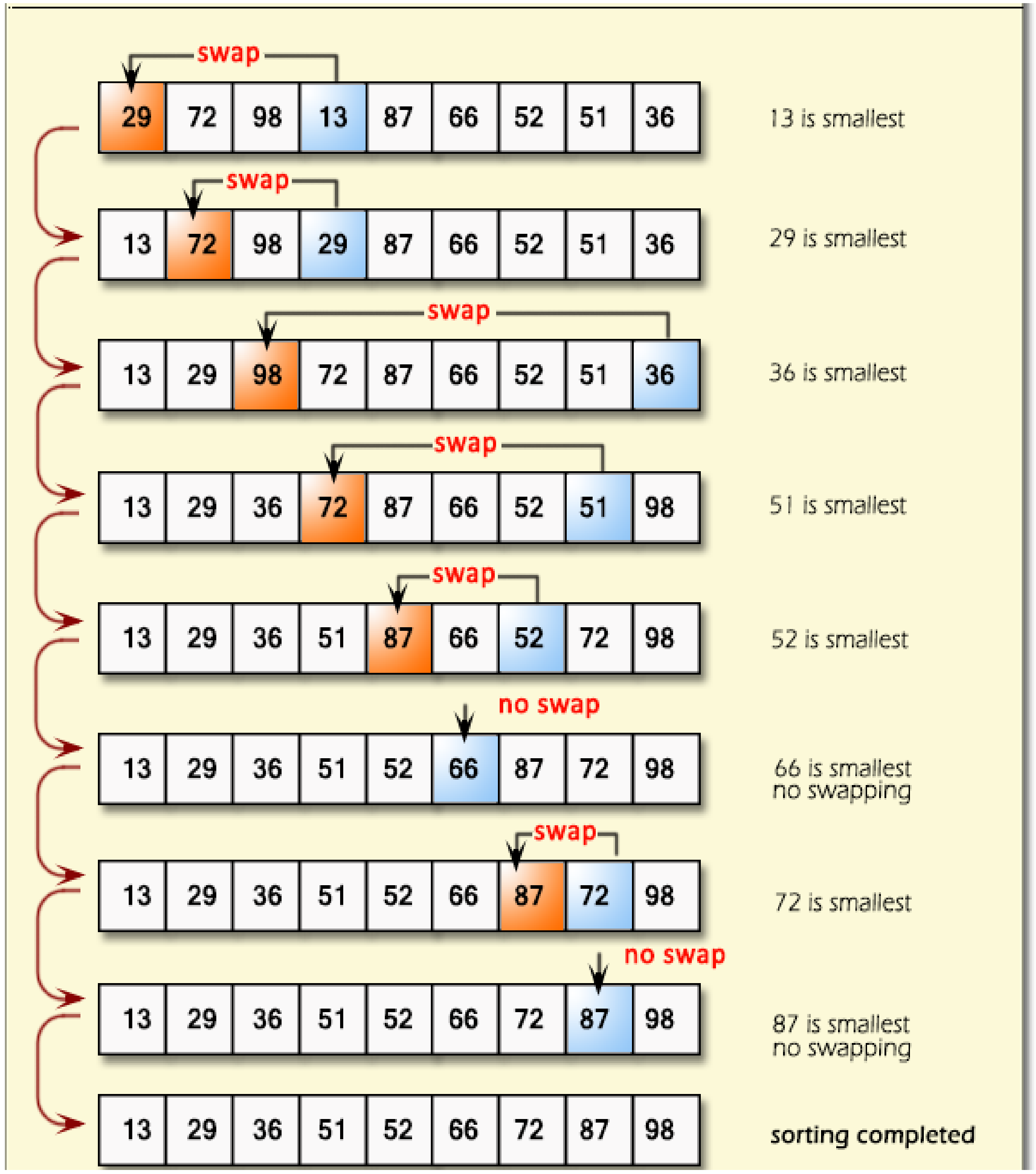
Also $O(n^2)$ in best and average cases.

Selection Sort

Idea behind Selection Sort :

- The algorithm first finds the smallest element in the array and exchanges it with the element in the first position.
- Then finds the second smallest element and exchanges it with the element in the second position.
- This process is continued until the entire array is sorted.

Selection Sort



Selection Sort

```
Selection_Sort (A[],  n)
{
  for(i=0; i < n-1; i++ )
    {   min = i;
         for(j=i+1; j < n; j++)
          {      if ( A[j] < A[min] )
                  min = j;
           }
           if (min != i)
               swap( A[i], A[min] );
     }
 }
```

Complexity analysis of Selection Sort

Worst case analysis

T(n) = (n-1)+(n-2)+………+1

= [n(n-1))]/2

= O(n^2)

Also O(n^2) in best and average cases.

Insertion Sort

Idea behind Insertion sort

Insertion sort maintains a sorted sub-array, and repetitively adds new elements one by one (i.e. one item at a time) to the sorted sub-array.

Elements are removed one by one from the unsorted sub-array and inserted into the correct position in a new, sorted sub-array.

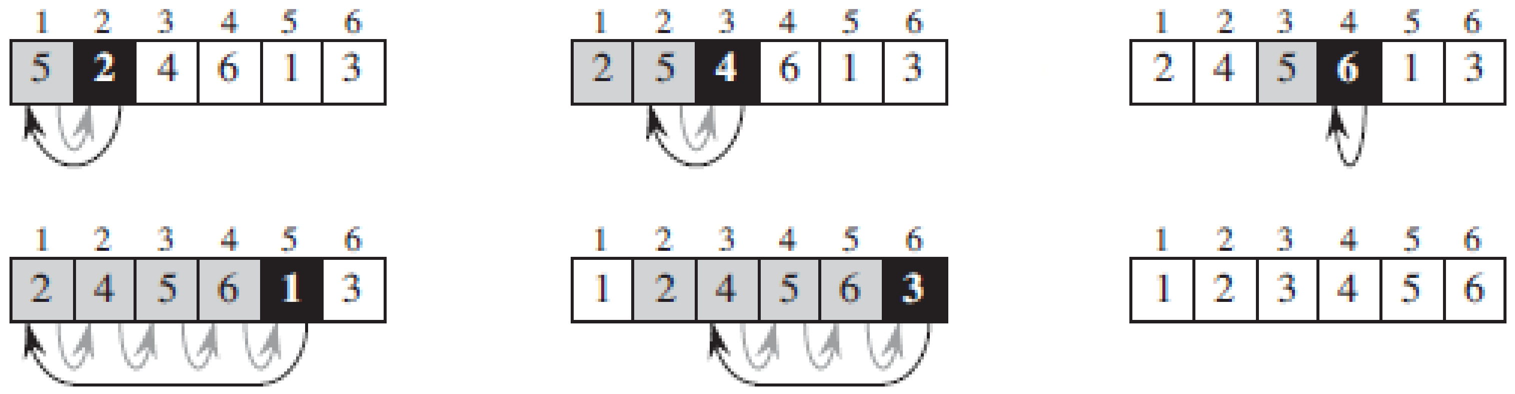
At the k-th step, put the k-th input element in the correct/proper place among the first k elements, as a result, after the k-th step, the first k elements are sorted.

Insertion Sort

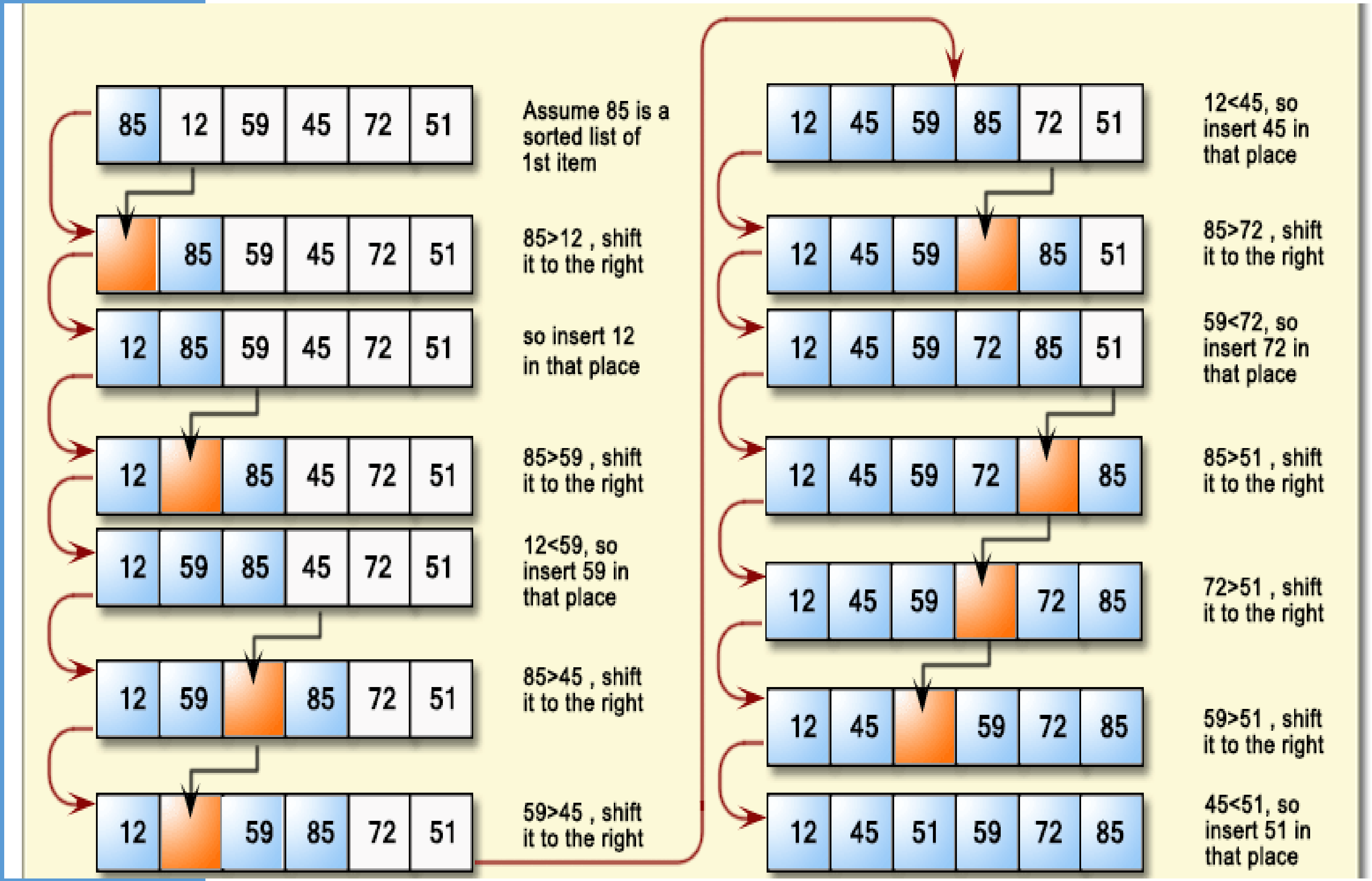
```
Insertion_Sort (A[],  n)
{
   for (i = 1; i < n; i++)
     {
         value = A[i];
          j = i;
          while ( j > 0  &&  A[j-1]> value)
             {
                A[j] = A[j-1];
                 j = j -1;
              }
            A[j] = value;
       }
}
```

Insertion Sort

Consider the array A = [5, 2, 4, 6, 1, 3]



Insertion Sort


85 12 59 45 72 51
Assume 85 is a sorted list of 1st item
85 59 45 72 51
85>12 , shift it to the right
12 85 59 45 72 51
so insert 12 in that place
12 85 45 72 51
85>59 , shift it to the right
12 59 85 45 72 51
12<59, so insert 59 in that place
12 59 85 72 51
85>45 , shift it to the right
12 59 85 72 51
59>45 , shift it to the right
12 45 59 85 72 51
12<45, so insert 45 in that place
12 45 59 85 51
85>72 , shift it to the right
12 45 59 72 85 51
59<72, so insert 72 in that place
12 45 59 72 85
85>51 , shift it to the right
12 45 59 72 85
72>51 , shift it to the right
12 45 59 72 85
59>51 , shift it to the right
12 45 51 59 72 85
45<51, so insert 51 in that place

Complexity Analysis of Insertion Sort

Worst case analysis

$T(n) = (n-1)+(n-2)+\dots\dots\dots+1$

$= [n(n-1)]/2$

$= O(n^2)$

The average case is also $O(n^2)$

$O(n)$ in best case