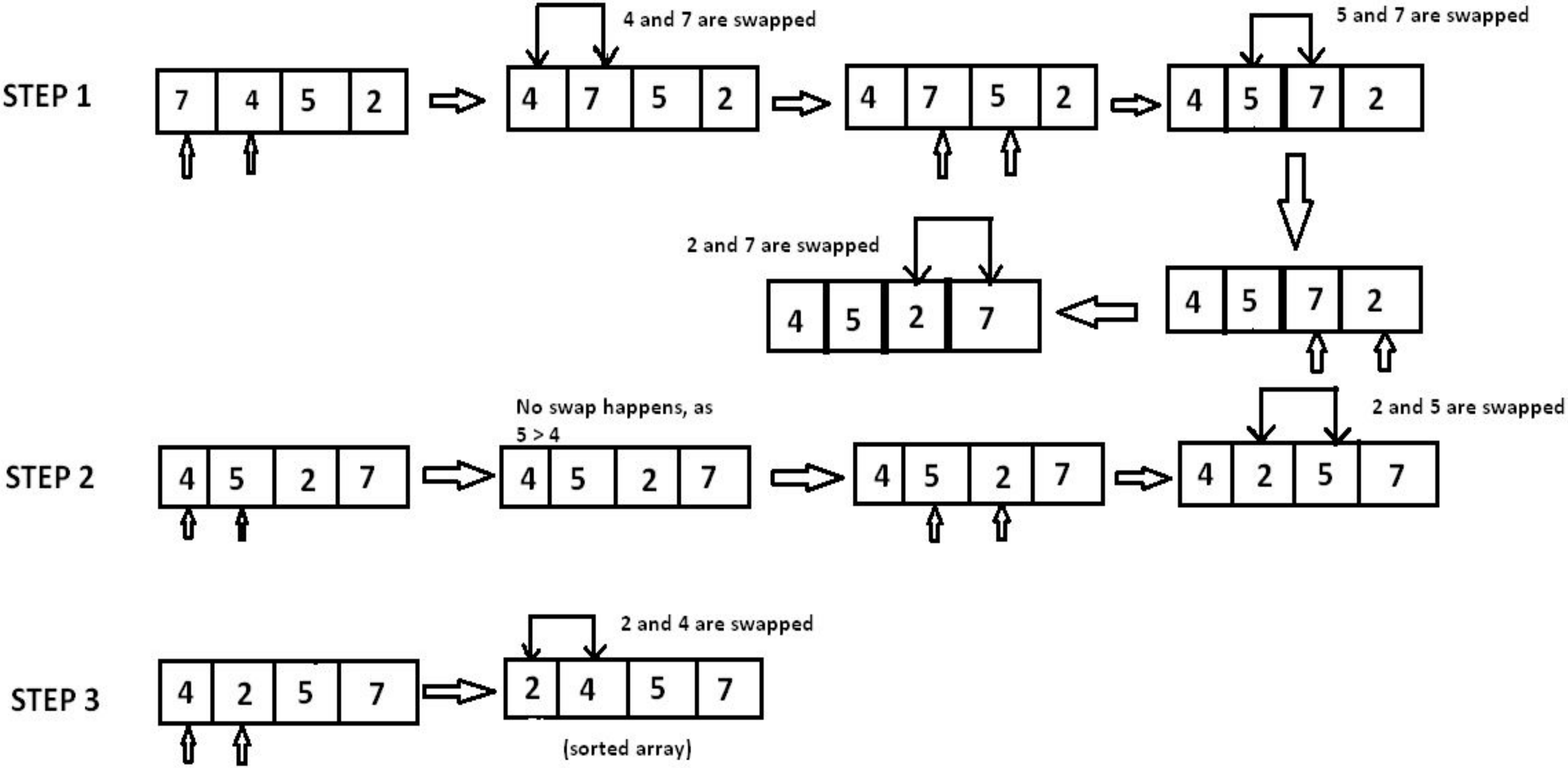
Sorting Techniques

Prof. Harish D.G.
Dept. of Computer and IT
College of Engineering,Pune
www.harishgadade.com

Sorting Techniques

- Bubble Sort
- Insertion Sort
- Selection Sort
- Merge Sort
- Quick Sort

Bubble Sort



STEP 1

7 4 5 2

4 and 7 are swapped

4 7 5 2

4 7 5 2

5 and 7 are swapped

4 5 7 2

4 5 7 2

2 and 7 are swapped

4 5 2 7

STEP 2

4 5 2 7

No swap happens, as
5 > 4

4 5 2 7

4 5 2 7

2 and 5 are swapped

4 2 5 7

STEP 3

4 2 5 7

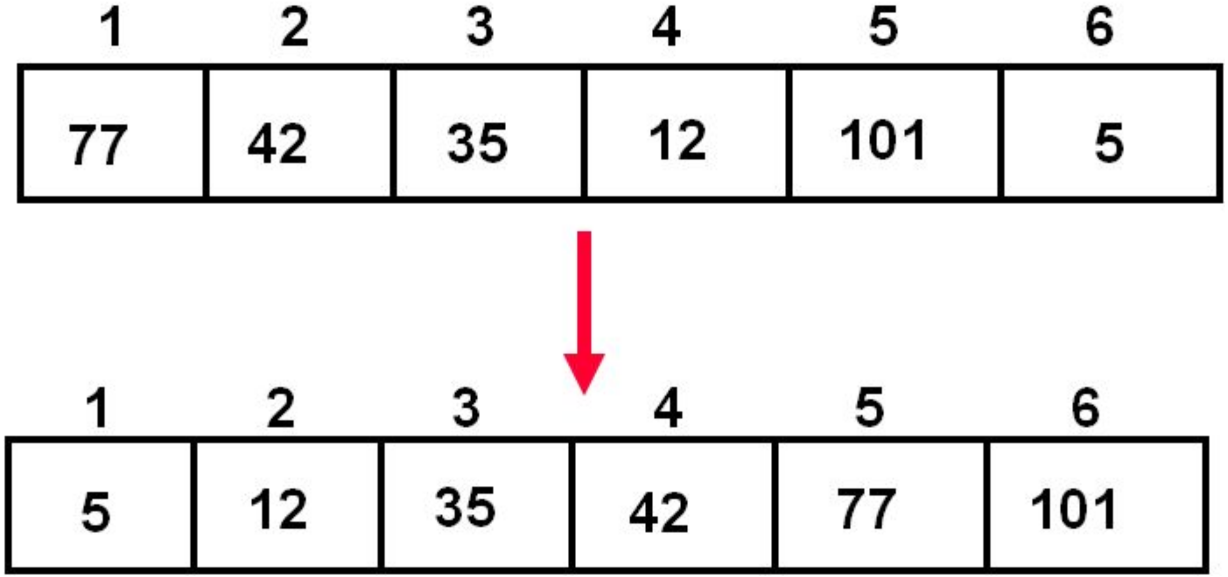
2 and 4 are swapped

2 4 5 7

(sorted array)

Bubble Sort

This is simplest and most popular sorting method. We do this bubble sort in several Iterations called Passes.



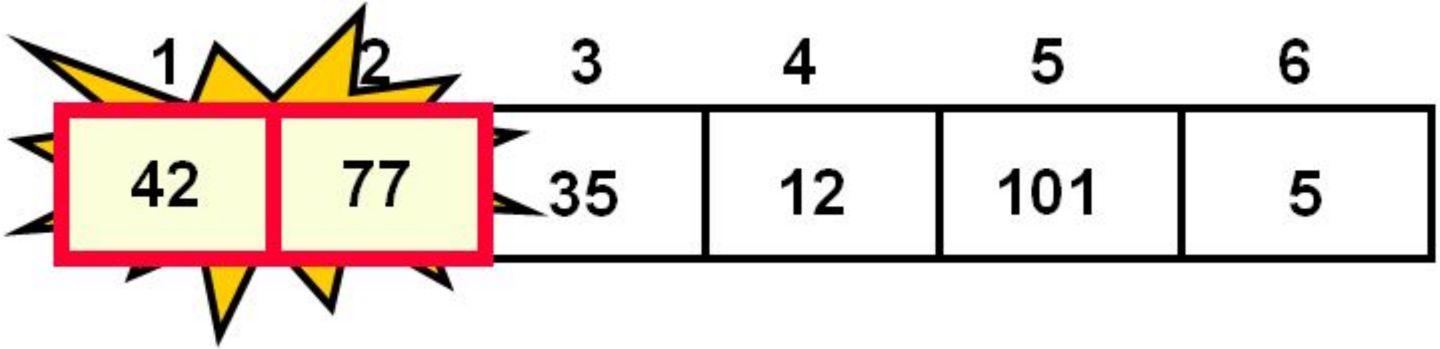
1	2	3	4	5	6
77	42	35	12	101	5

1	2	3	4	5	6
5	12	35	42	77	101

Bubble Sort

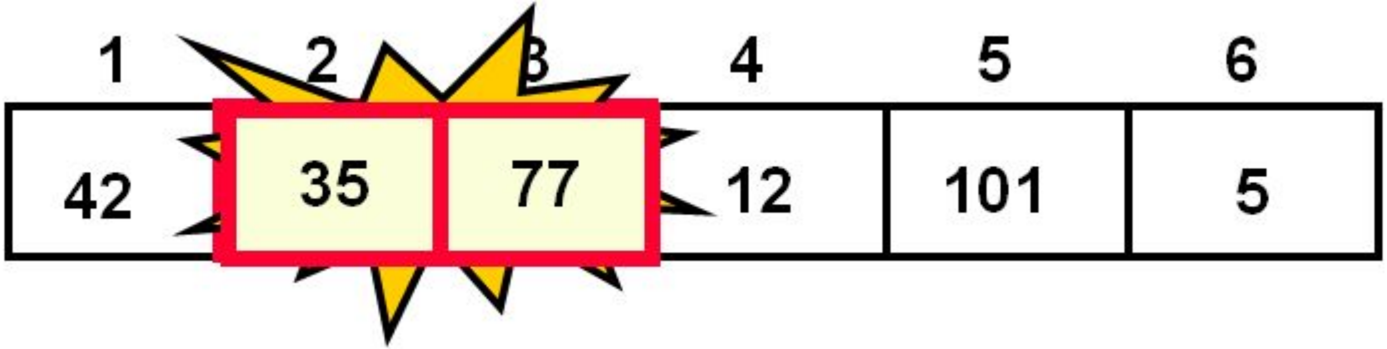
1	2	3	4	5	6
77	42	35	12	101	5

Bubble Sort



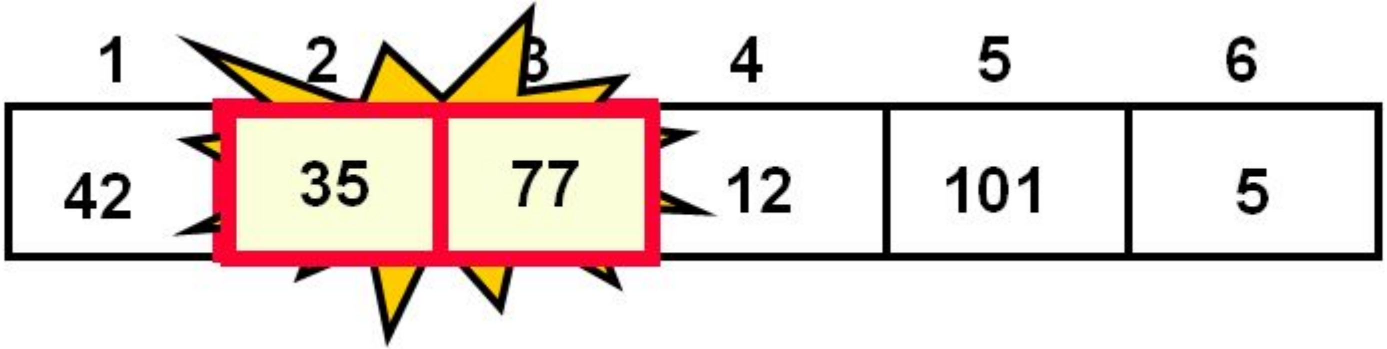
1	2	3	4	5	6
42	77	35	12	101	5

Bubble Sort



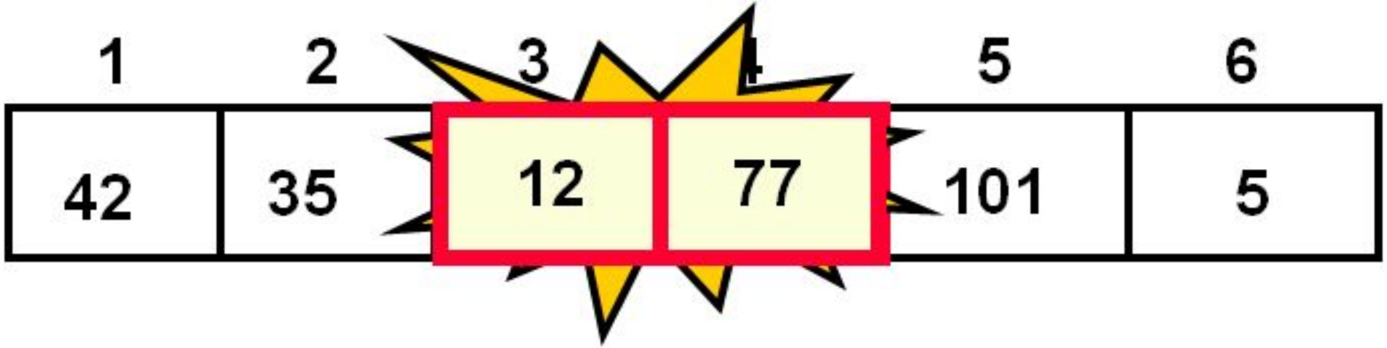
1	2	3	4	5	6
42	35	77	12	101	5

Bubble Sort



1	2	3	4	5	6
42	35	77	12	101	5

Bubble Sort



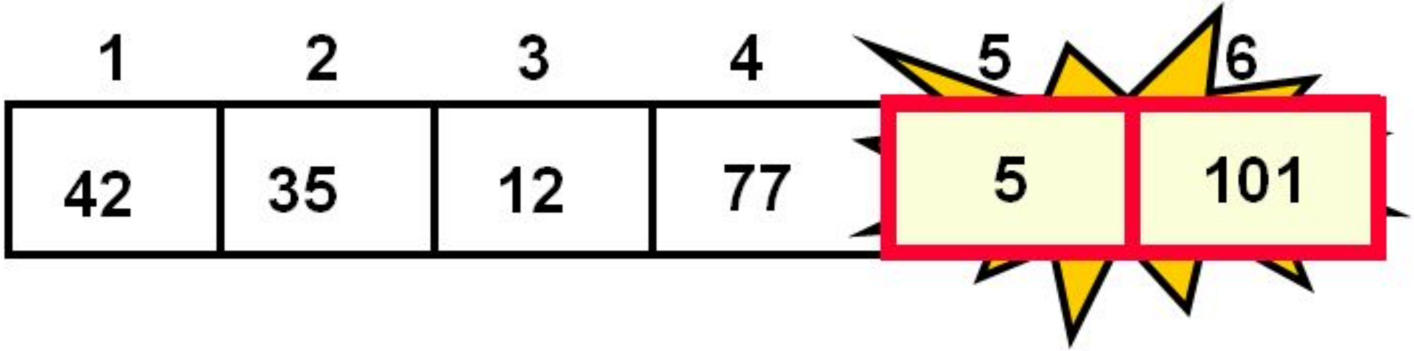
1	2	3	4	5	6
42	35	12	77	101	5

Bubble Sort

1	2	3	4	5	6
42	35	12	77	101	5

No need to swap

Bubble Sort



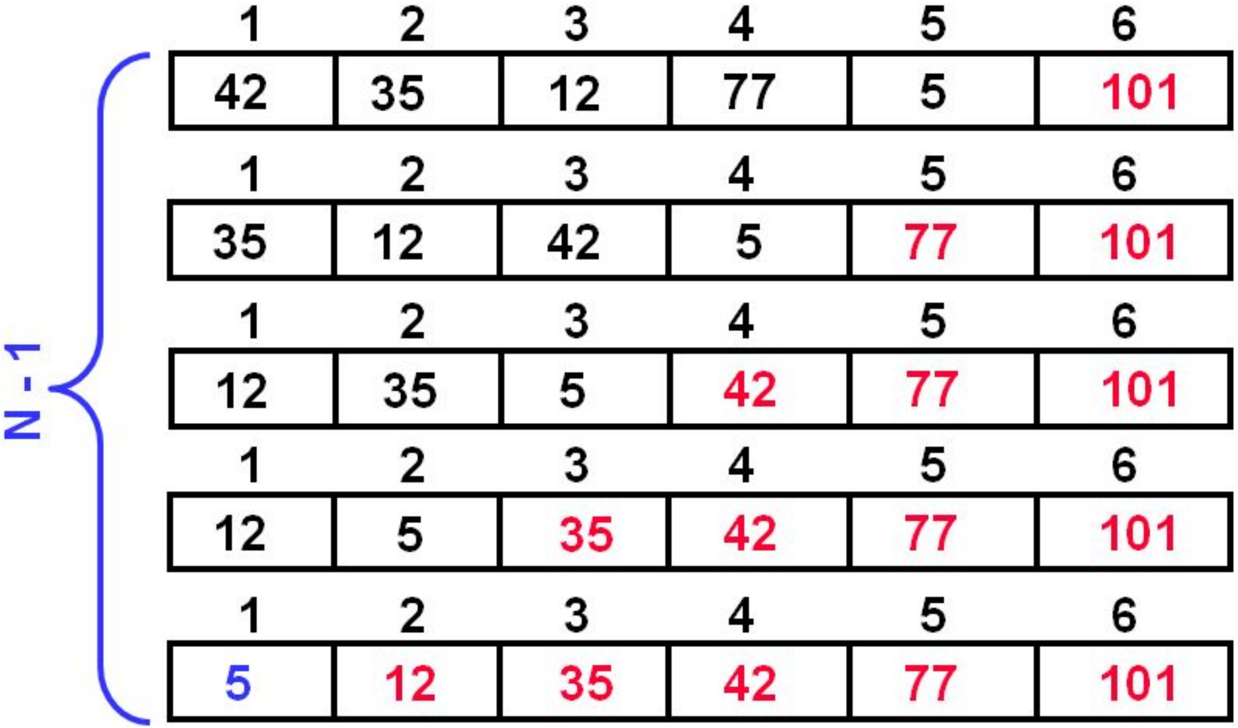
1	2	3	4	5	6
42	35	12	77	5	101

Bubble Sort

1	2	3	4	5	6
42	35	12	77	5	101

Largest value correctly placed

Bubble Sort



N - 1

1	2	3	4	5	6
42	35	12	77	5	101

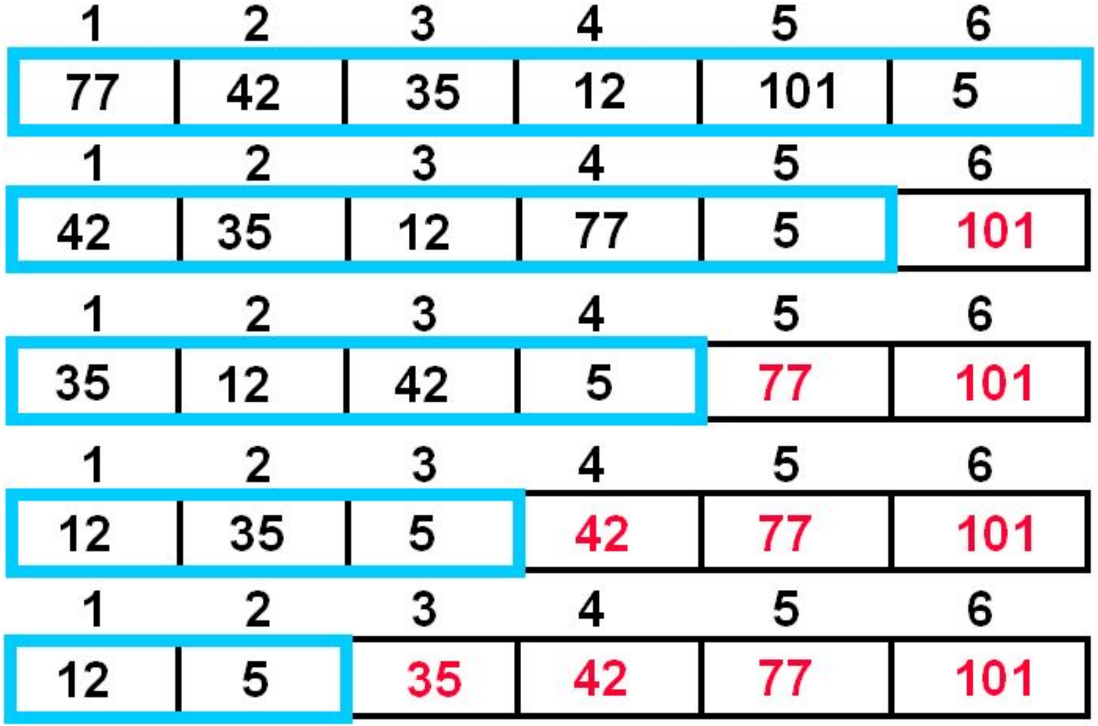
1	2	3	4	5	6
35	12	42	5	77	101

1	2	3	4	5	6
12	35	5	42	77	101

1	2	3	4	5	6
12	5	35	42	77	101

1	2	3	4	5	6
5	12	35	42	77	101

Bubble Sort



1	2	3	4	5	6
77	42	35	12	101	5

1	2	3	4	5	6
42	35	12	77	5	101

1	2	3	4	5	6
35	12	42	5	77	101

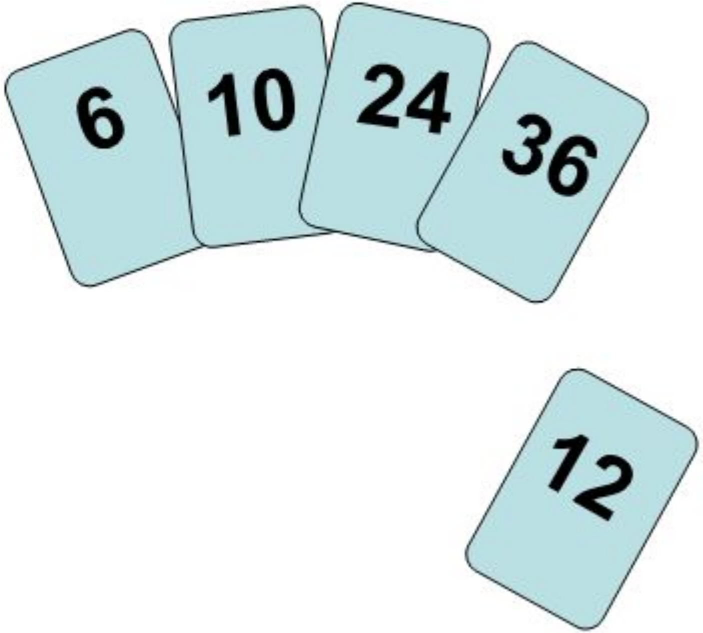
1	2	3	4	5	6
12	35	5	42	77	101

1	2	3	4	5	6
12	5	35	42	77	101

Insertion Sort

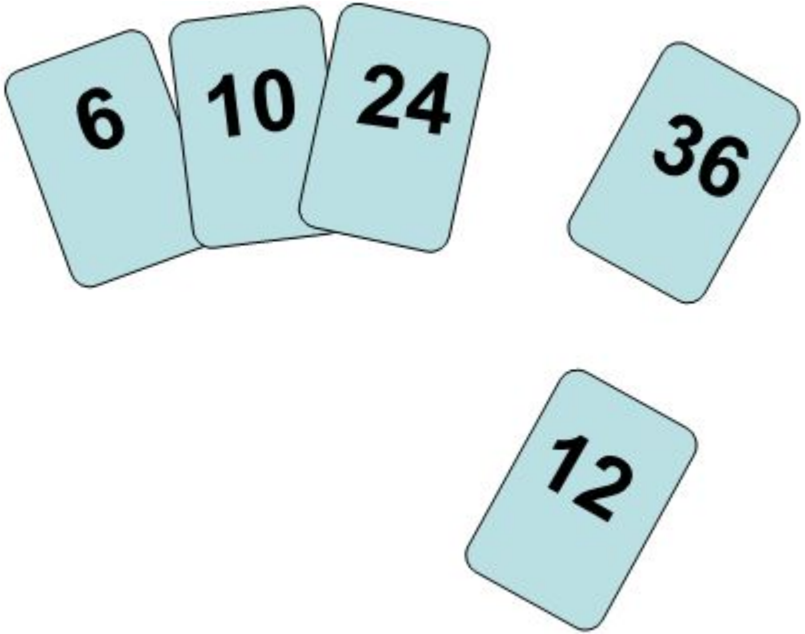
- **Idea:** like sorting a hand of playing cards
  - Start with an empty left hand and the cards facing down on the table.
  - Remove one card at a time from the table, and insert it into the correct position in the left hand
    - compare it with each of the cards already in the hand, from right to left
  - The cards held in the left hand are sorted
    - these cards were originally the top cards of the pile on the table

Insertion Sort



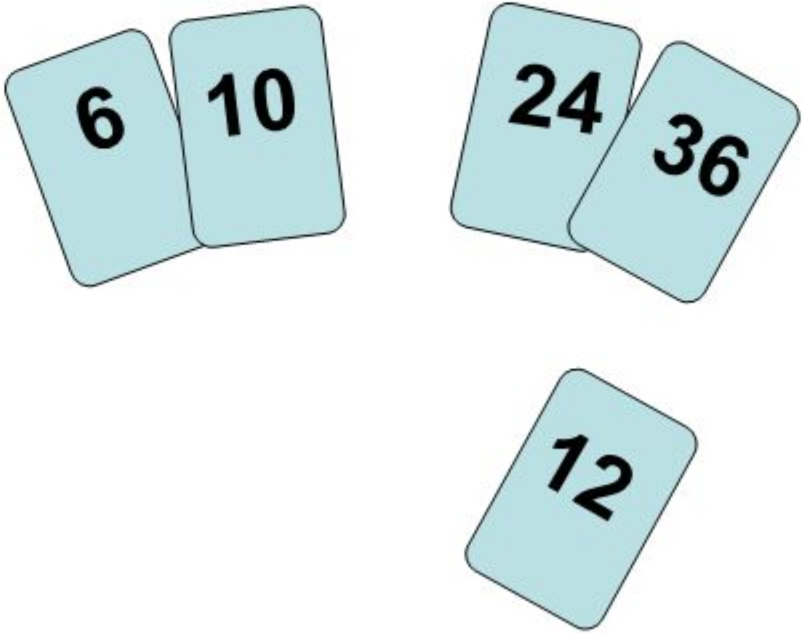
To insert 12, we need to make room for it by moving first 36 and then 24.

Insertion Sort



To insert 12, we need to make room for it by moving first 36 and then 24.

Insertion Sort



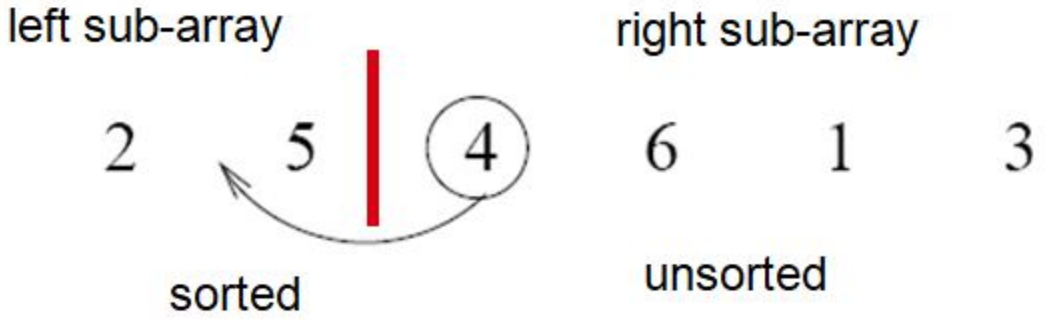
To insert 12, we need to make room for it by moving first 36 and then 24.

Insertion Sort

input array

5 2 4 6 1 3

at each iteration, the array is divided in two sub-arrays:



left sub-array

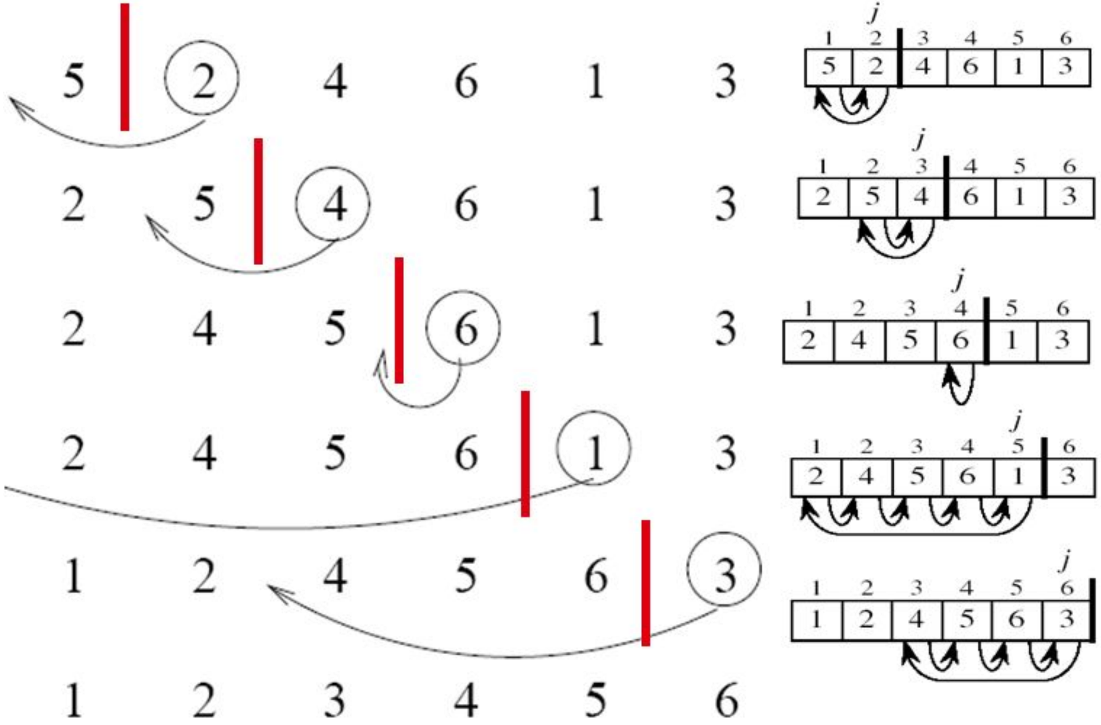
right sub-array

2 5 | 4 6 1 3

sorted

unsorted

Insertion Sort



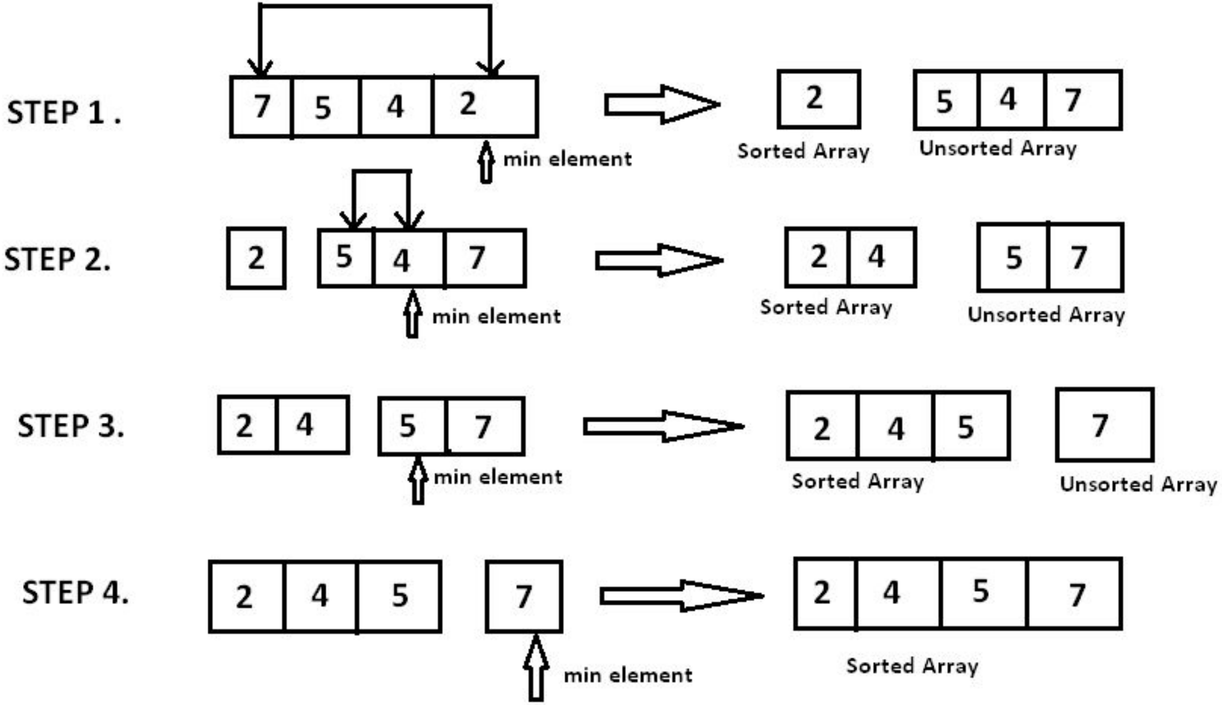
Insertion Sort

```
Void insertion (int a[10], int n)
{
  int i, j, temp;
  for ( i = 1; i <= n-1; i + + )
  {
      temp = A [i];
    j = i - 1;
    while (( j > = 0 ) && ( A [j]  > temp ))
    {
      A [ j + 1] = A [j];
      j = j - 1;
    }
    A [ j + 1 ] = temp;
}
```

Selection Sort

- **Idea:**
  - Find the smallest element in the array
  - Exchange it with the element in the first position
  - Find the second smallest element and exchange it with the element in the second position
  - Continue until the array is sorted
- **Disadvantage:**
  - Running time depends only slightly on the amount of order in the file

Selection Sort



Selection Sort

```
void selection_sort (int A[ ], int n)
{
        int minimum;        // temporary variable to store the position of minimum element
        for(int i = 0; i < n-1 ; i++)
        {
                minimum = i ;      // assuming the first element to be the minimum of the unsorted array .
                for(int j = i+1; j < n ; j++ )      // gives the effective size of the unsorted  array .
                {
                        if(A[ j ] < A[ minimum ])       //finds the minimum element
                        {
                                minimum = j ;
                        }
                }
                swap ( A[ minimum ], A[ i ]) ;       // putting minimum element on its proper position.
        }
}
```