**Selection Sort:**

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

1) The subarray which is already sorted.  
2) Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

arr[] = 64 25 12 22 11

// Find the minimum element in arr[0...4]

// and place it at beginning

**11** 25 12 22 64

// Find the minimum element in arr[1...4]

// and place it at beginning of arr[1...4]

11 **12** 25 22 64

// Find the minimum element in arr[2...4]

// and place it at beginning of arr[2...4]

11 12 **22** 25 64

// Find the minimum element in arr[3...4]

// and place it at beginning of arr[3...4]

11 12 22 **25** 64

Bubble Sort

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

Example:  
First Pass:  
( 5 1 4 2 8 ) –> ( 1 5 4 2 8 ), Here, algorithm compares the first two elements, and swaps since 5 > 1.  
( 1 5 4 2 8 ) –>  ( 1 4 5 2 8 ), Swap since 5 > 4  
( 1 4 5 2 8 ) –>  ( 1 4 2 5 8 ), Swap since 5 > 2  
( 1 4 2 5 8 ) –> ( 1 4 2 5 8 ), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:  
( 1 4 2 5 8 ) –> ( 1 4 2 5 8 )  
( 1 4 2 5 8 ) –> ( 1 2 4 5 8 ), Swap since 4 > 2  
( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )  
( 1 2 4 5 8 ) –>  ( 1 2 4 5 8 )  
Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.

Third Pass:  
( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )  
( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )  
( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )  
( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )