Bubble Sort Algorithm

Bubble sort algorithm starts by comparing the first two elements of an array and swapping if necessary, i.e., if you want to sort the elements of array in ascending order and if the first element is greater than second then, you need to swap the elements but, if the first element is smaller than second, you mustn't swap the element. Then, again second and third elements are compared and swapped if it is necessary and this process go on until last and second last element is compared and swapped. This completes the first step of bubble sort.

If there are `n` elements to be sorted then, the process mentioned above should be repeated `n-1` times to get required result. But, for better performance, in second step, last and second last elements are not compared because, the proper element is automatically placed at last after first step. Similarly, in third step, last and second last and second last and third last elements are not compared and so on.

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**)

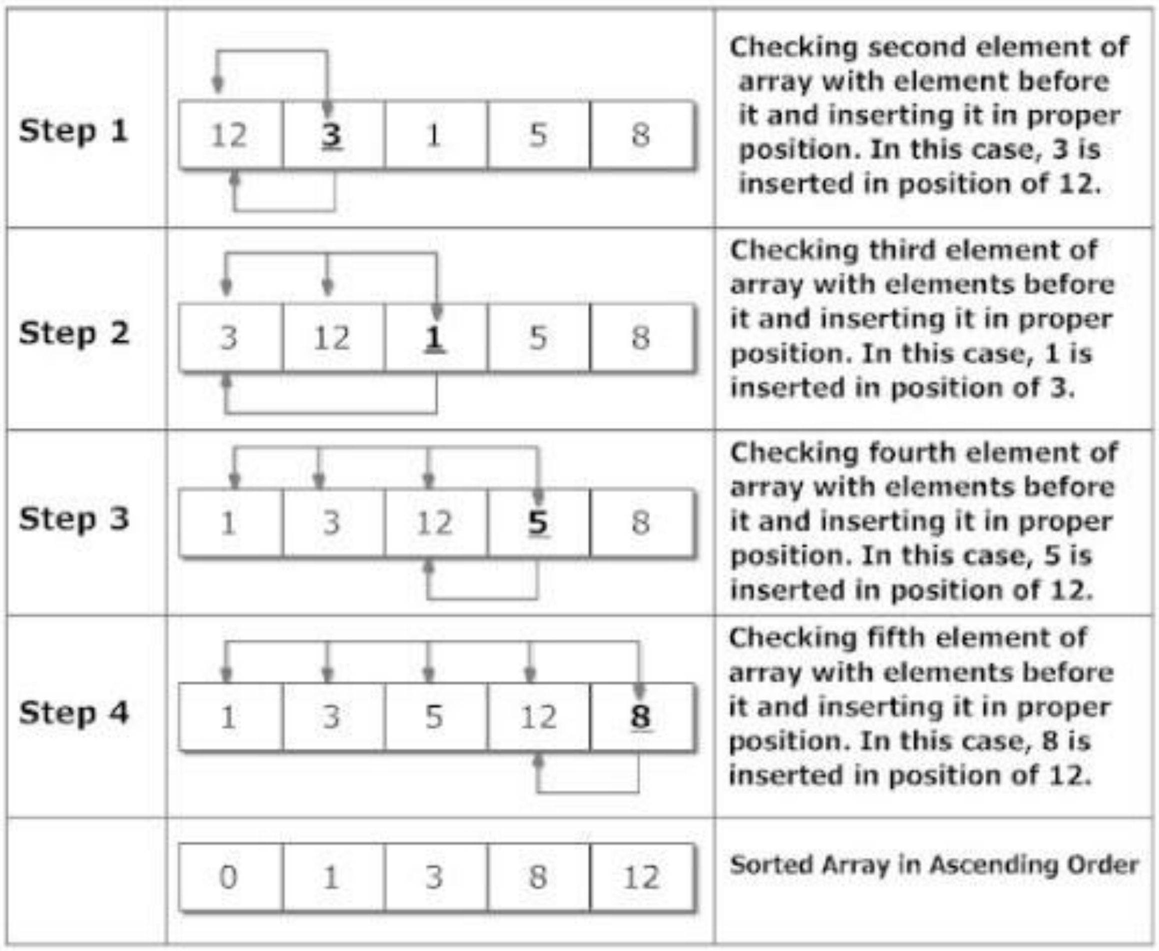
Watch this Simulation: https://www.youtube.com/watch?v=nmhjrI-aW5o

```c
/*C Program To Sort data in ascending order using bubble sort.*/
#include <stdio.h>
int main()
{
    int data[100],i,n,step,temp;
    printf("Enter the number of elements to be sorted: ");
    scanf("%d",&n);
    for(i=0;i<n;++i)
    {
        printf("%d. Enter element: ",i+1);
        scanf("%d",&data[i]);
    }

    for(step=0;step<n-1;++step)
    for(i=0;i<n-step-1;++i)
    {
        if(data[i]>data[i+1])   /* To sort in descending order, change
> to < in this line. */
        {
            temp=data[i];
            data[i]=data[i+1];
            data[i+1]=temp;
        }
    }
    printf("In ascending order: ");
    for(i=0;i<n;++i)
         printf("%d  ",data[i]);
    return 0;
}
```

Insertion Sort Algorithm

How insertion sort Algorithm works?



Step 1	12	**3**	1	5	8	Checking second element of array with element before it and inserting it in proper position. In this case, 3 is inserted in position of 12.
Step 2	3	12	**1**	5	8	Checking third element of array with elements before it and inserting it in proper position. In this case, 1 is inserted in position of 3.
Step 3	1	3	12	**5**	8	Checking fourth element of array with elements before it and inserting it in proper position. In this case, 5 is inserted in position of 12.
Step 4	1	3	5	12	**8**	Checking fifth element of array with elements before it and inserting it in proper position. In this case, 8 is inserted in position of 12.
	0	1	3	8	12	Sorted Array in Ascending Order

Figure: Sorting Array in Ascending Order Using Insertion Sort Algorithm

Explanation

Suppose, you want to sort elements in ascending as in above figure. Then,

1. Step 1: The second element of an array is compared with the elements that appears before it (only first element in this case). If the second element is smaller than first element, second element is inserted in the position of first element. After first step, first two elements of an array will be sorted.
2. Step 2: The third element of an array is compared with the elements that appears before it (first and second element). If third element is smaller than first element, it is inserted in the position of first element. If third element is larger than first element but, smaller than second element, it is inserted in the position of second element. If third element is larger than both the elements, it is kept in the position as it is. After second step, first three elements of an array will be sorted.
3. Step 3: Similarly, the fourth element of an array is compared with the elements that appears before it (first, second and third element) and the same procedure is applied and that element is inserted in the proper position. After third step, first four elements of an array will be sorted.

If there are `n` elements to be sorted. Then, this procedure is repeated `n-1` times to get sorted list of array.

Watch this Simulation: https://www.youtube.com/watch?v=OGzPmgsI-pQ

```c
/*Sorting Elements of an array in ascending order using insertion sort
algorithm*/
#include<stdio.h>
int main()
{
        int data[100],n,temp,i,j;
        printf("Enter number of terms(should be less than 100): ");
        scanf("%d",&n);
        printf("Enter elements: ");
        for(i=0;i<n;i++)
        {
                scanf("%d",&data[i]);
        }
        for(i=1;i<n;i++)
        {
                temp = data[i];
                j=i-1;
                while(temp<data[j] && j>=0)
/*To sort elements in descending order, change temp<data[j] to
temp>data[j] in above line.*/
                {
                        data[j+1] = data[j];
                        --j;
                }
                data[j+1]=temp;
        }
        printf("In ascending order: ");
        for(i=0; i<n; i++)
                printf("%d\t",data[i]);
    return 0; }
```

Selection Sort Algorithm

Selection sort algorithm starts by comparing first two elements of an array and swapping if necessary, i.e., if you want to sort the elements of array in ascending order and if the first element is greater than second then, you need to swap the elements but, if the first element is smaller than second, leave the elements as it is. Then, again first element and third element are compared and swapped if necessary. This process goes on until first and last element of an array is compared. This completes the first step of selection sort.

If there are `n` elements to be sorted then, the process mentioned above should be repeated `n-1` times to get required result. But, for better performance, in second step, comparison starts from second element because after first step, the required number is automatically placed at the first (i.e, In case of sorting in ascending order, smallest element will be at first and in case of sorting in descending order, largest element will be at first.). Similarly, in third step, comparison starts from third element and so on.

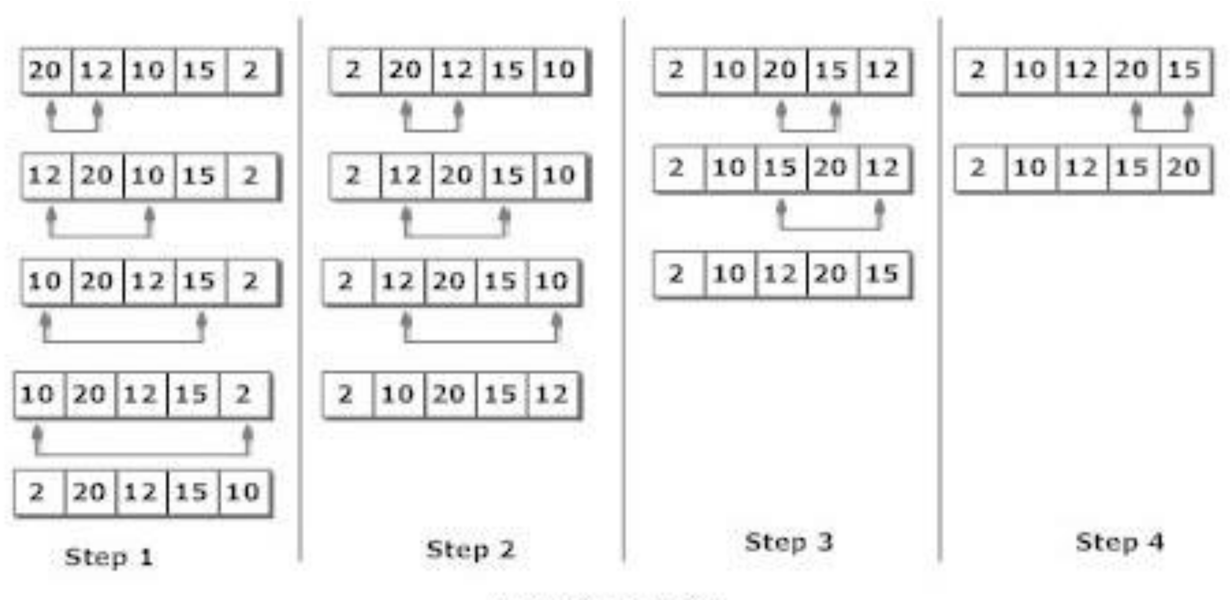


Figure: Selection Sort

Watch this Simulation: https://www.youtube.com/watch?v=xWBP4lzkoyM

```c
#include <stdio.h>
int main()
 {
    int data[100],i,n,steps,temp;
    printf("Enter the number of elements to be sorted: ");
    scanf("%d",&n);
    for(i=0;i<n;++i)
      {
       printf("%d. Enter element: ",i+1);
       scanf("%d",&data[i]);
    }
    for(steps=0;steps<n;++steps)
    for(i=steps+1;i<n;++i)
     {
         if(data[steps]>data[i])
/* To sort in descending order, change > to <. */
          {
             temp=data[steps];
             data[steps]=data[i];
             data[i]=temp;
         }
    }
    printf("In ascending order: ");
    for(i=0;i<n;++i)
        printf("%d  ",data[i]);
    return 0;
}
```