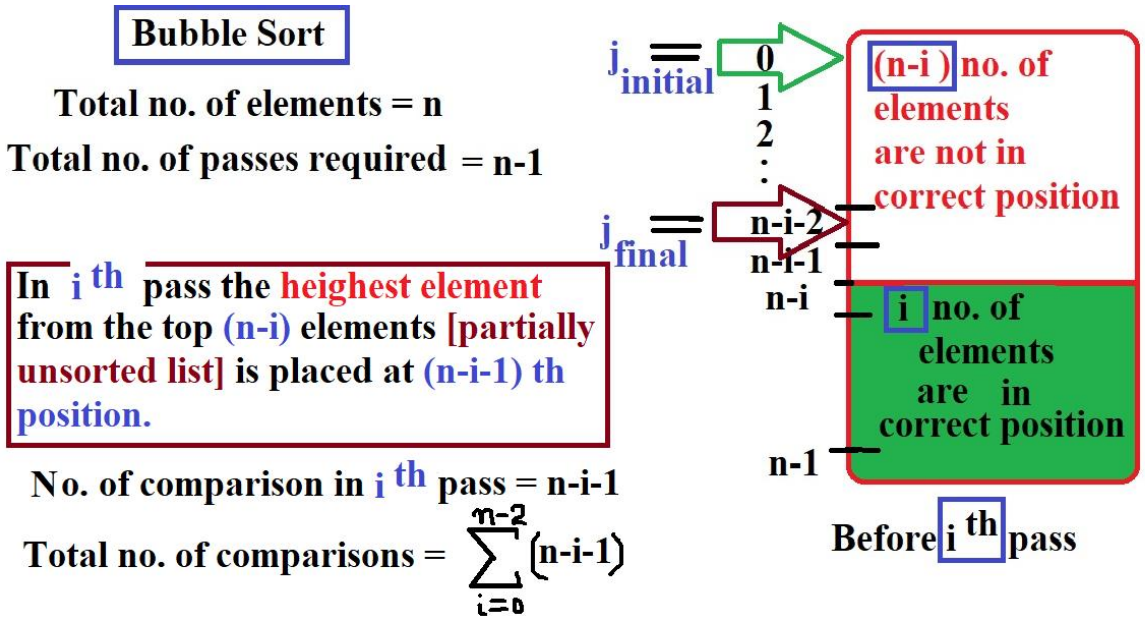
Bubble Sort

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**)



```
#include<stdio.h>
#include<conio.h>

#define MAX  10

void  input_elements( int *, int );
void  display(int *,  int);
void  bubble_sort(int *,  int);



void main()
{
   int *a, N;
   clrscr( );
   printf("\n Enter the number of elements:");
   scanf("%d", &N);
   a=(int *) malloc(sizeof(int)*N);
   input_elements( a, N);
   printf("\n Elements before sorting:");
   display(a, N);
   bubble_sort (a, N);
   printf("\n Elements after sorting:");
   display(a, N);
   getchar( );
}
```

```
void  input_elements( int *x, int  n)
{
     int i;
     for(i=0;i <n; i++)
      {
          printf("\n Enter the Element[%d]:", i+1);
          fflush(stdin);
          scanf("%d",x+i);
       }
}

void  display(int *x,  int n)
{
    int i;
   printf("\n");
    for(i=0;i<n;i++)
         printf("\t %d", x[i]);
}

void  bubble_sort (int *x,  int  n)
{
     int  i, j, t, flag;
    /* ============================
i: index of pass varies from 0 to n-2,
```

j: comparison index varies from 0 to (n-i-1) in i^{th} pass.
flag : an indicator variable
Condition (flag = = 1) is TRUE for an unsorted array.
Condition (flag = = 0) is TRUE for a sorted array.
Here, flag is initially set to zero.
If there is not a single swap in a pass then the given array is a sorted array. Under this condition flag will not be 1, it will remain 0 and next pass will not be executed due to failure of the condition (i<n-1 && flag) .
==========*/

```
    for(i=0, flag=1; i<n-1 && flag; i++)
        for(j=0, flag=0; j<n-i-1; j++)
            if(x[j]>x[j+1])
             {
                flag=1;
                t=x[j];
                x[j]=x[j+1];
                x[j+1]=t;
             }
}
```

Complexity Analysis:

No. of comparison in i^{th} pass = n-i-1

$$\text{Total no. of comparisons} = \sum_{i=0}^{n-2} (n-i-1)$$

=(n-1)+(n-2)+(n-3)+…+3+2+1

=n(n-1)/2

=n^2/2-n/2

=O(n^2)