Bubble Sort

Implementation:

```
void bubble_sort(int arr[],int n)
{
   int i,j,temp,flag;
   for(i=0; i<n-1; i++)
   {
      flag = 0;
      for(j=0; j<n-i-1; j++)
      {
         if(arr[j]>arr[j+1])
         {
            temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
            flag = 1;
         }
      }
      if(flag==0)
      {
         break;
      }
   }
}
```

Analysis:

We start analyzing the Insertion Sort procedure with the time "cost" of each statement and the number of times each statement is executed.

BUBBLE_SORT(*A*,*n*)		**Cost**	**Time**
1	for i <-1 to n-1	c1	n
2	flag <- 0	c2	n-1
3	for j <- 1 to n-i-1		
4	if A[j] > A[j+1]		
5	Swap(A[j],A[j+1])	c3	$\frac{n(n-1)}{2}$
6	Flag = 1		
7	if flag == 0	c4	n-1
	break		

For each i= 1,2,3……n, where n is the size of the array. When a for or while loop exits in the usual way (i.e., due to the test in the loop header), the test is executed one time more than the loop body.
The running time of the algorithm is the sum of running times for each statement executed; a statement that takes c_i steps to execute and executes n times will contribute $c_i * n$ times to the total running time.

If the array is in reverse sorted order—that is, in decreasing order the **worst case** occurs. For each value of i, the inner loop will execute for n-1 times.
Initially i=1,
Then j =1 and It will run for 1 to n-i-1 = n-1 times
j = 2 and It will run for 2 to n-i-1 = n-2 times
j = 3 and it will run for 3 to n-i-1 = n-3 times
…….
…….
j = n-i-1 and It will run n-i-1 to n-i-1 = 1 times

For reversely sorted array elements, if there are n elements, for all iteration the total checked condition will be (n-1) +(n-2) +(n-3) +……+3+2+1

This equation can be written as,

$$= \frac{(n-1)(n-1+1)}{2}$$

$$= \frac{n(n-1)}{2}$$ which is a polynomial equation.

Therefore, the running time of BUBBLE SORT is,

$$T(n) = c_1 n + c_2(n-1) + +c_3(\tfrac{n(n-1)}{2}) + c_4(\mathrm{n}-1)$$
$$= c_1 n + c_2 n - c_2 + \tfrac{c_3}{2}(n^2 - n) + c_4 n -$$
$$= c_1 n + c_2 n - c_2 + \frac{c_3}{2} n^2 - \frac{c_3}{2} n + c_4 n - c_4$$
$$= \tfrac{c_3}{2} n^2 + (c_1 + c_2 - \tfrac{c_3}{2} + c_4) n + (-c_2 - c_4)$$

This equation can be written as,

$T(n)$ = $an^2 + bn + c$ (where a ,b & c are some constant) **which is an quadratic function.**

In bubble sort if the array is already sorted then **best case** occurs and the inner loop will execute for only $(n-1)$ times as $\boldsymbol{flag == 0}$ remain as 0 which means no swaps occurs.

The equation can be written as,

$T(n)$ = $an + b$ **which is a linear function.**

Worst Case:

5	4	3	2	1

For reversely sorted array elements, if there are n elements, for all iteration the total checked condition will be (n-1) +(n-2) +(n-3) +……+3+2+1

This equation can be written as,

$= \frac{(n-1)(n-1+1)}{2}$

$= \frac{n(n-1)}{2}$ which is a polynomial equation. Considering the Highest order of n is 2.

Therefore, we can say that in worst case, the complexity of bubble sort is O(n^2).

Best Case:

1	2	3	4	5

If the array is already sorted, then only inner loop will execute for (n-1) times.

Therefore, in best case the complexity of bubble sort is O(n).