INSERTION SORT (CODE)

```
#include<iostream>

 using namespace std;

void display(int arr[], int size)
 {
   for (int i = 0; i < size; i++)

                     cout << arr[i] << " ";

    cout << "\n";
 }

void insertion_sort(int arr[], int n)
 {
  int i, j ;
  int element ;

  for (i = 2; i < n; i++)
   {
    element = arr[i];
    j = i - 1;

      while (j > -1 && arr[j] > element)
       {
         arr[j + 1] = arr[j] ;
         j-- ;
       }

    arr[j + 1] = element;
  }

 }

 int main()
 {
    int n ;

            cout << "\nEnter the Number of Elements: ";
    cin >> n;

            int arr[n] ;

            cout << "\nEnter the Elements: ";

            for (int i = 0; i < n; i++)

                cin >> arr[i] ;
```

```
        cout << "\nArray before Sorting: ";

    display(arr, n);

    insertion_sort(arr, n);

    cout << "\nArray After Insertion Sorting: ";

    display(arr, n);

        return 0;
}
```

COMPLEXITY ANALYSIS

Pseudocode		Cost	NO. OF TIMES IT IS RUN
1.	for i <- 2 to length[A]	C_1	n
2.	key = A[i]	C_2	$n - 1$
3.	j <- i – 1	C_3	$n - 1$
4.	while j > 0 and A[j] > key	C_4	$\sum_{j=2}^{n} t_j$
5.	A[j + 1] = A[j]	C_5	$\sum_{j=2}^{n} t_{j-1}$
6.	j = j – 1	C_6	$\sum_{j=2}^{n} t_{j-1}$
7.	end while		
8.	A[j + 1] = key	C_8	$n - 1$
9.	end for		

Best Case Analysis

In Best Case i.e., when the array is already sorted, $\mathbf{t_j = 1}$

$$T(n) = C_1 * n + (C_2 + C_3) * (n - 1) + C_4 * (n - 1) + (C_5 + C_6) * (0) + C_8 * (n - 1)$$

$$T(n) = (C_1 + C_2 + C_3 + C_4 + C_8) * (n) - (C_2 + C_3 + C_4 + C_8)$$

$$T(n) = \Omega (n)$$

Worst Case Analysis

In Worst Case i.e., when the array is reversely sorted (in descending order), $\mathbf{t_j = j}$

$\sum_{j=2}^{n} j = 2 + 3 + \ldots\ldots + n$

$= 1 + 2 + 3 + \ldots\ldots + n - 1 = n * (n + 1) / 2 - 1$

$\sum_{j=2}^{n} (j - 1) = 1 + 2 + 3 + \ldots\ldots + n - 1$

$= n * (n - 1) / 2$

$T(n) = C_1 * n + (C_2 + C_3) * (n - 1) + C_4 *((n+1)*(n)/2 - 1) + (C_5 + C_6) * ((n - 1)*(n) / 2) + C_8 * (n - 1)$

$T(n) = ((C_4 + C_5 + C_6)/2)*(n^2) + (C_1 + C_2 + C_3 + C_4/2 - (C_5 + C_6)/2 + C_8)*(n) - (C_2 + C_3 + C_4 + C_8)$

$T(n) = O (n^2)$