



**Queensland University  
of Technology**

Queensland University of Technology

# IFN 505 PROGRAMING ANALYSIS

Assignment 2

Darsheel Deshpande  
N10287213

## **SUMMARY:**

This report will analyse two algorithms of sorting of number in an ascending order [bubble sort and distribution counting] in an array of elements both theoretically and experimentally on basis of efficiency, partial correctness and total correctness by executable programs, measuring their runtime characteristic and comparison of the result with theoretical calculation we have got. These two algorithms are introduced by using c# programming language.

## **TASK 1: DESCRIPTION OF ALGORITHM:**

The sorting algorithms which are bubble sorting and distribution counting are designed to arrange the number in array in non-decreasing order format with the help of different techniques. Both the algorithm will take the array in form of static data provided by user or through dynamic data from random number generation with help of processor. The bubble sort algorithm uses the method of comparing two elements and swapping whereas the distribution counting algorithm is based on frequency it uses cumulative frequency for sorting of elements in array. bubble sort algorithm is better and quick smaller number but when the input from user increases distribution counting is better than bubble sort. Therefore, bubble sort is good for a smaller number of input and distribution array is better for larger number of inputs.

### **ALGORITHM 1: BUBBLE SORT:**

The bubble sort algorithm will sort the elements in an given array in non-decreasing order .after taking the input array and storing the value it will go to do condition initially set the value of swapped as Boolean false .In the loop from  $i = 0$  to  $n-2$  where the value of  $n$  will be the length of an array which is given .initial step it will compare the value at  $a[0]$  and  $a[1]$  since the loop start with 0 and the next iteration will be 1 .After comparing two value the smallest value will be assigned to smallest iteration. In the loop iteration  $i$  will be incremented and the next element in array will be fetched and compare with previous element and will be swapped if it is smaller then the element at previous iteration.IN the loop the same procedure will continue till  $i$  is greater than or equal to value of  $n-2$  .once the value of  $i$  become equal to  $n-2$  or grater then it the loop will terminate. The value of swapped will be true in loop till swapping is performed. once the loop is terminated it will come go to while condition and later return the array with the value in it arranged in non-decreasing order.

### **ALGORITHM 2: DISTRIBUTION COUNTING:**

The distribution counting algorithm will sort the element in array in non-decreasing format with the help of cumulative frequency method. It will initially ask the user with the value of  $L$  and  $U$  which are lower and upper limit respectively. Where lower limit should be less than the upper limit .after getting the input array  $A[i]$  it will initialise two array one is  $d[]$  for frequency whose length will be  $U+1$  and  $s[]$  which will be sorted array whose length will be equal to length of input array .For the first loop  $j = 0$  to  $U-L$  we give the value 0 to  $d[j]$  that is the element in frequency array will have value of 0 initially . The second loop is to calculate the frequency of element in input array which start from  $i=0$  till  $i=n-1$  the operation performed in  $d[A[i]-L]=d[A[i]-L]+1$  it computes the histogram of element for each occurrence in input array .for the third loop  $j=1$  till  $U-L$  which does the operation  $d[j]=d[j-1]+d[j]$  will perform prefix sum computation on count to determine which position the element having a particular key should be placed. IN the fourth loop element  $i=n-1$  till 0 operation done are  $j=A[i]-L$ ,  $S[D[j]-1]=A[i]$ ,  $d[j]=d[j]-1$  .Elements will be moved to sorted position in output array since in fourth loop take iteration from back to front the elements are arranged in proper order. The input array will be sorted in non-decreasing order and return in  $s []$  in sorted form.

## THEORETICAL ANALYSIS OF ALGORITHM:

### FOR BUBBLE SORT:

#### PARTIAL CORRECTNESS:

The loop invariant (outer loop):

$\forall i: 0 \dots n-1 \wedge \forall m: 0 \dots i-1 A[m] < A[i]$

- The value of m is approaching value of i
- The value of A[m] will be lower when compare to the value A[i] at the end of the loop.
- Initially there is no element on left hand side therefore the initial variant is true for A [0].
- After iteration A[i] will be compared with A[i+1]. if A[i] is higher than A[i+1] it will be swapped due to condition specified in loop. Since the left side element are sorted and lower than value is iterated the invariant will still hold.
- After termination at i=n-1, the last value that is A[n] and A[n-1] will be compared the big value is put on the right side and the small value will be put on the left side. Thus, the invariant still holds the condition.

#### TOTAL CORRECTNESS:

In loop condition i is in range from 0 to n-2 and kept on increasing with iteration. So at a certain point it is tend to reach at i=n-2 which will terminate the loop. the loop can be terminated within few iterations depending on value of n therefore the algorithm will also be terminated at same number of iterations.

### FOR DISTRIBUTIONCOUNTING:

#### PARTIAL CORRECTNESS:

The invariant loop (outer loop)

$\forall i: n-1 \dots 0 \wedge \forall m \in I \text{ such that } m < i s[m] < s[i]$

- Invariant proposed above as i decreases till 0, so the values in array will always be sorted in non - descending order.
- Initially only a single value is assigned at i=0, therefore the invariant is true as the array is already sort in non-descending order.
- In next iteration if the value enter is less than the value presented it will hold position on left if the value is higher then the value present in array then it will be hold right position of value present .thus the invariant is rue here as for any value m smaller then i  $s[m] < s[i]$ .

#### TOTAL CORRECTNESS:

loop condition starts from i=n-1 to 0 by decrementing i on every iteration.so when  $i \leq 0$  the loop will get terminated as condition become true and thus loop will stop. Thus, after some iteration depending on value of n the loop will terminate thus the algorithm will also terminate with loop.

### TASK 3:

#### FOR BUBBLESORT:

BASIC OPERATION:  $A[i] > A[i+1]$

The basic operation compare the value i th iteration with that of value at (i+1 )th position.

PROBLEM Size: n

EFFICENCY:

BEST CASE: the best case for basic operation will be when the number in array are already in ascending order. Therefore, number basic operation performed will be equal to n-1.

For example =A[1,2,3]=

First it will compare (1&2) no swapping will take place since  $1 < 2$ .

Next it will compare (2&3) no swapping will take place since  $2 < 3$

Therefore, number of comparisons =2, no of element =3

Therefore, in equation:

$$\sum_{k=0}^{n-1} 1 = (n - 1) - 0 + 1 = n - 1 - 0 + 1 = n$$

Therefore:  $c_{\text{best}} = O(n)$

#### WORST CASE:

The worst case will be when the number in array will be in descending order.

Example :

A[3,2,1]=

First it will compare (3&2) swapping will take place since  $3 > 2$  therefore array will become =A[2,3,1]

Second it will compare (3&1) swapping will take place since  $3 > 1$  therefore array will become =A[2,1,3]

Now it will compare (2&1) swapping will take place since  $2 > 1$  therefore array will become =A[1,2,3]

Now it will compare (2&3) no swapping will take place since  $3 > 2$ .

Therefore, total no of comparison =4 no of elements =3.

Therefore for A[n]=A [n,n-1,n-2.....]

No of comparison=(n-1) +(n-2) .....1

Therefore, in equation of efficiency:

$$\sum_{i=1}^n (n - 1) = \sum_{i=1}^n \frac{n(n+1)}{2} = O(n^2)$$

$C_{\text{worst}} = O(n^2)$

#### AVERAGE CASE:

The efficiency average case of bubble sort will be same as that of worst case  $C_{\text{average}} = O(n^2)$ .

## FOR ALGORITHM2:

### FOR DISTRIBUTIONCOUNTING:

BASIC OPERATION:  $D[A(i)-L]=D[A(i)-L]+1$

The loop condition of for basic operation is  $i=0....n-1$  thus it is ran for entire length of an array. since the array is executed for entire length of array the efficiency will remain same even if array is not sorted.

Therefore worst, average and best case will be same for algorithm2.

Therefore, in equation

$$\sum_{k=0}^{n-1} 1 = (n-1) - 0 + 1 = n - 1 - 0 + 1 = n$$

Efficiency of basic operation will be  $=O(n)$

For efficiency of whole algorithm we need to add efficiency of all loops involved in algorithm.

Efficiency of  $j=1$  to  $U-L$  do  $d[j]=0$  is  $O(k)$

Efficiency of  $j=1$  to  $U-L$  do  $d[j]=d[j-1]+d[j]$  is  $O(k)$

Efficiency of for

$i=n-1$  down to 0 do

$j=A[i]-L$

$s[d[j]-1]=A[i]$

$d[j]=d[j]-1$

Therefore, efficiency of this loop is  $O(n)$

The efficiency of algorithm  $=O(n)+O(n)+O(k)+O(k)=O(k+n)$

## TASK 4 :

C# was the programming language and the software which was used to run the code is "visual studio 2019" .c# was first introduced by Microsoft when they launched .net framework .From then c# has gain popularity among users due to easy to learn ,it has retain the functions and characteristic which are used in c++.This features make c# popular among the users.(Clark & Sanders, 2011).

The environment and operating system used to run program was windows 10 pro ,intel core i5 7200,64 bit operating system,x64 processor. The software used for running program is "visual studio" 2019 version.

### USE OF PROGRAMMING LANGUAGE AND SOFTWARE TO IMPLEMENT ALGORITHM:

- The algorithm was implemented in form of program in two type one with static input data and other with dynamic input data.
- Static program for bubble sort was coded to calculate the efficiency of algorithm by testing worst case by giving input number in descending order, average case by giving input as random number and best case by giving input number in ascending order. counter was used to check the efficiency of program. It was also used for checking the correctness of algorithm.

- Static program for distribution counting was coded to calculate the efficiency of algorithm by testing worst case by giving input number in descending order, average case by giving input as random number elements and best case by giving input number in ascending order. counter was used to check the efficiency of program. It was also used for checking the correctness of algorithm
- Dynamic program for bubble sort was used to check efficiency for longer array as well as different length of an array with use of randomly generated array based on upper limit and lower limit specified by user. The length of array varied from 0 to 50. It helps us to understand how the program react to different number of array and different length of array.
- Dynamic program for distribution counting was used to check efficiency for longer array as well as different length of an array with use of randomly generated array based on upper limit and lower limit specified by user. The length of array varied from 0 to 50. It helps us to understand how the program react to different number of array and different length of array.
- Time was measured in dynamic program of both algorithms. IN bubble sort time was calculated by starting timer at start of for loop and ending it after the while. For distribution array the timer was started at first for loop and stopped at the end of last loop. This will help was to find the time efficiency of both algorithm and compare them with each other.
- Counter was used to count how many times did the basic operation was performed in a program.
- For input of upper limit and lower limit the value for upper limit should be greater than the lower limit and the value for upper limit should be maximum value and lower limit should be minimum.

## TASK 5:FUNCTIONAL TESTING:

I have done four test on each of static program which are based on best case, worst case and 2 average case .Data used is same for both bubble and distribution counting program

### ALGORITHM 1 BUBBLE SORT:

#### TEST1:

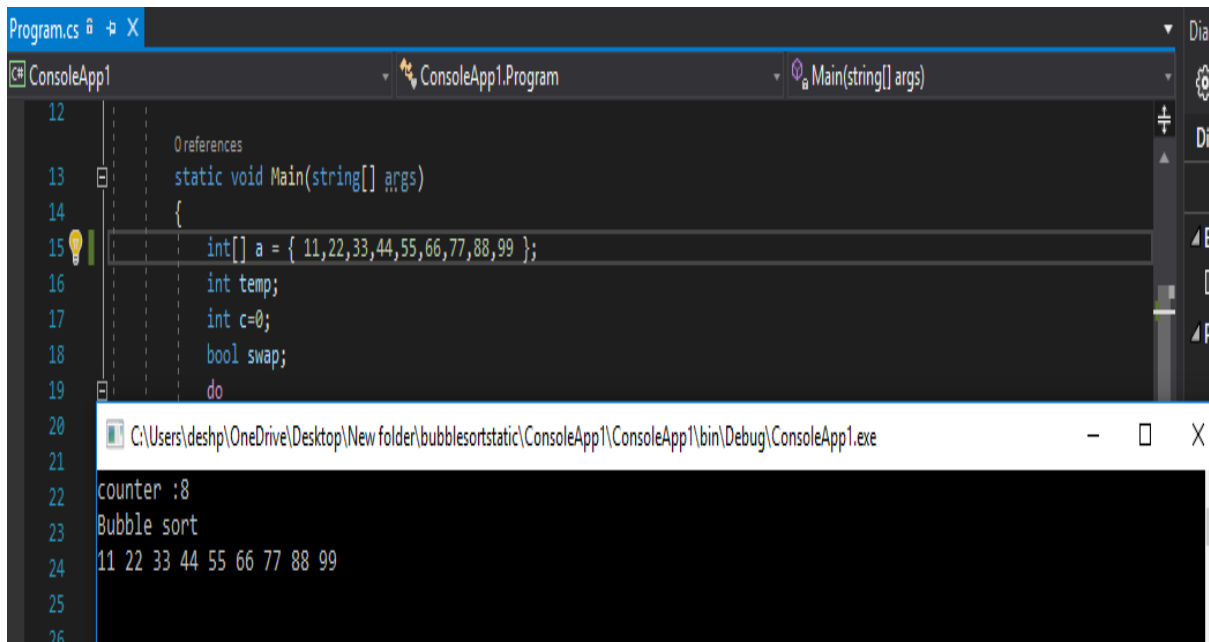
For best case: input a [] ={11,22,33,44,55,66,77,88,99} size of array =9

```

12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
References
static void Main(string[] args)
{
    int[] a = { 11, 22, 33, 44, 55, 66, 77, 88, 99 };
    int temp;
    int c=0;
    bool swap;
    do
    {
        swap = false;
        for (int i = 0; i <= a.Length - 2; i++)
        {
            c++;
            if (a[i] > a[i + 1])
            {
                temp = a[i];
                a[i] = a[i + 1];
                a[i + 1] = temp;
                swap = true;
            }
        }
    }
    while (swap);
    Console.WriteLine("counter : " + c);
    Console.WriteLine("Bubble sort");
    foreach (int num in a)
        Console.Write(num + " ");
    Console.Read();
}

```

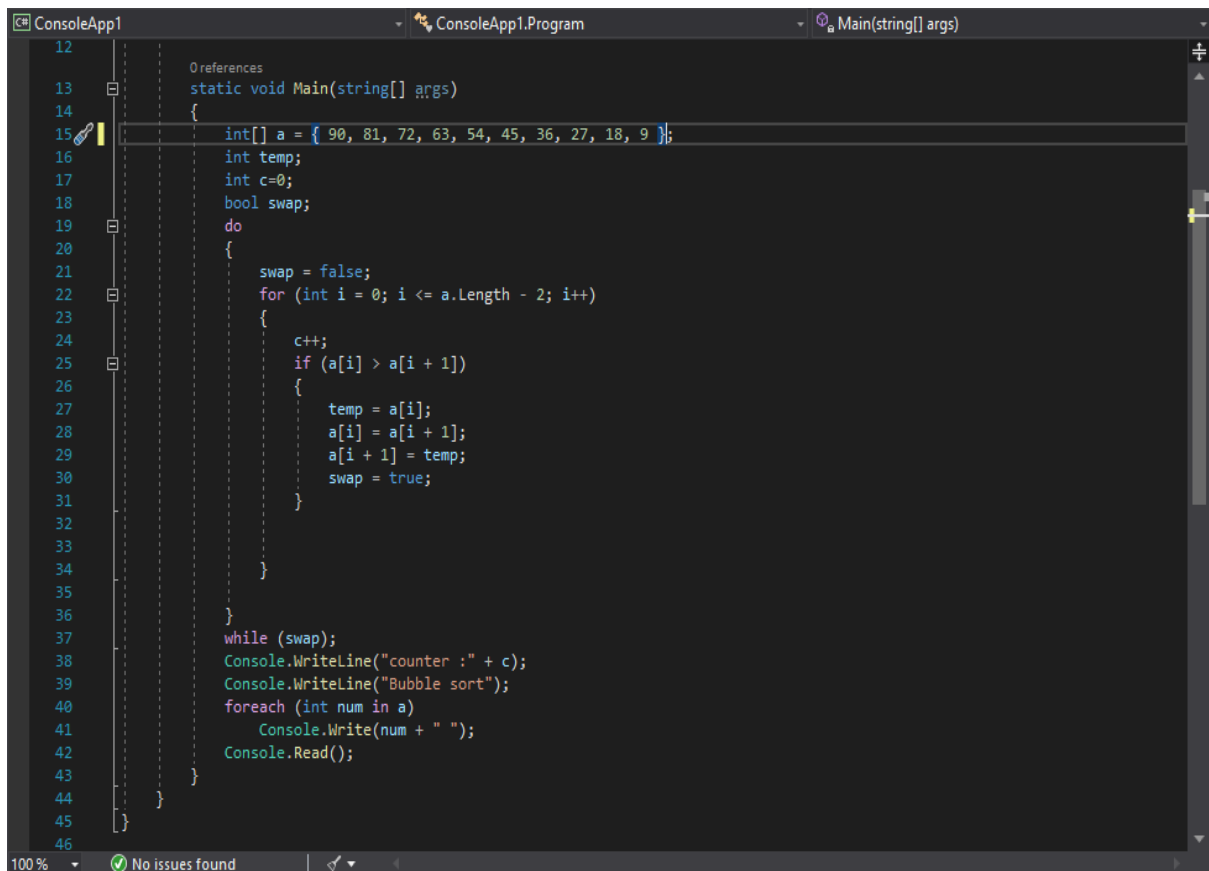
## output



```
Program.cs X
ConsoleApp1 ConsoleApp1.Program Main(string[] args)
12
13 static void Main(string[] args)
14 {
15     int[] a = { 11,22,33,44,55,66,77,88,99 };
16     int temp;
17     int c=0;
18     bool swap;
19     do
20
21
22 counter :8
23 Bubble sort
24 11 22 33 44 55 66 77 88 99
25
26
```

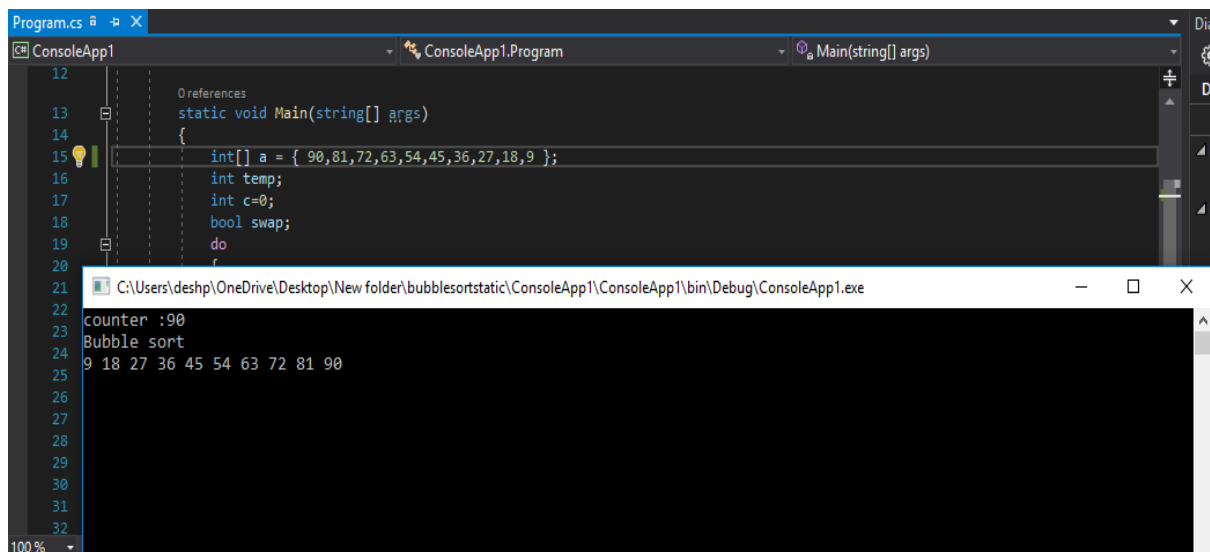
## TEST 2:

For worst case :input a [ ]={90,81,72,63,54,45,36,27,18,9} size of array =10



```
ConsoleApp1 ConsoleApp1.Program Main(string[] args)
12
13 static void Main(string[] args)
14 {
15     int[] a = { 90, 81, 72, 63, 54, 45, 36, 27, 18, 9 };
16     int temp;
17     int c=0;
18     bool swap;
19     do
20     {
21         swap = false;
22         for (int i = 0; i <= a.Length - 2; i++)
23         {
24             c++;
25             if (a[i] > a[i + 1])
26             {
27                 temp = a[i];
28                 a[i] = a[i + 1];
29                 a[i + 1] = temp;
30                 swap = true;
31             }
32         }
33     } while (swap);
34     Console.WriteLine("counter :" + c);
35     Console.WriteLine("Bubble sort");
36     foreach (int num in a)
37     {
38         Console.Write(num + " ");
39     }
40     Console.Read();
41 }
42
43
44
45
46
```

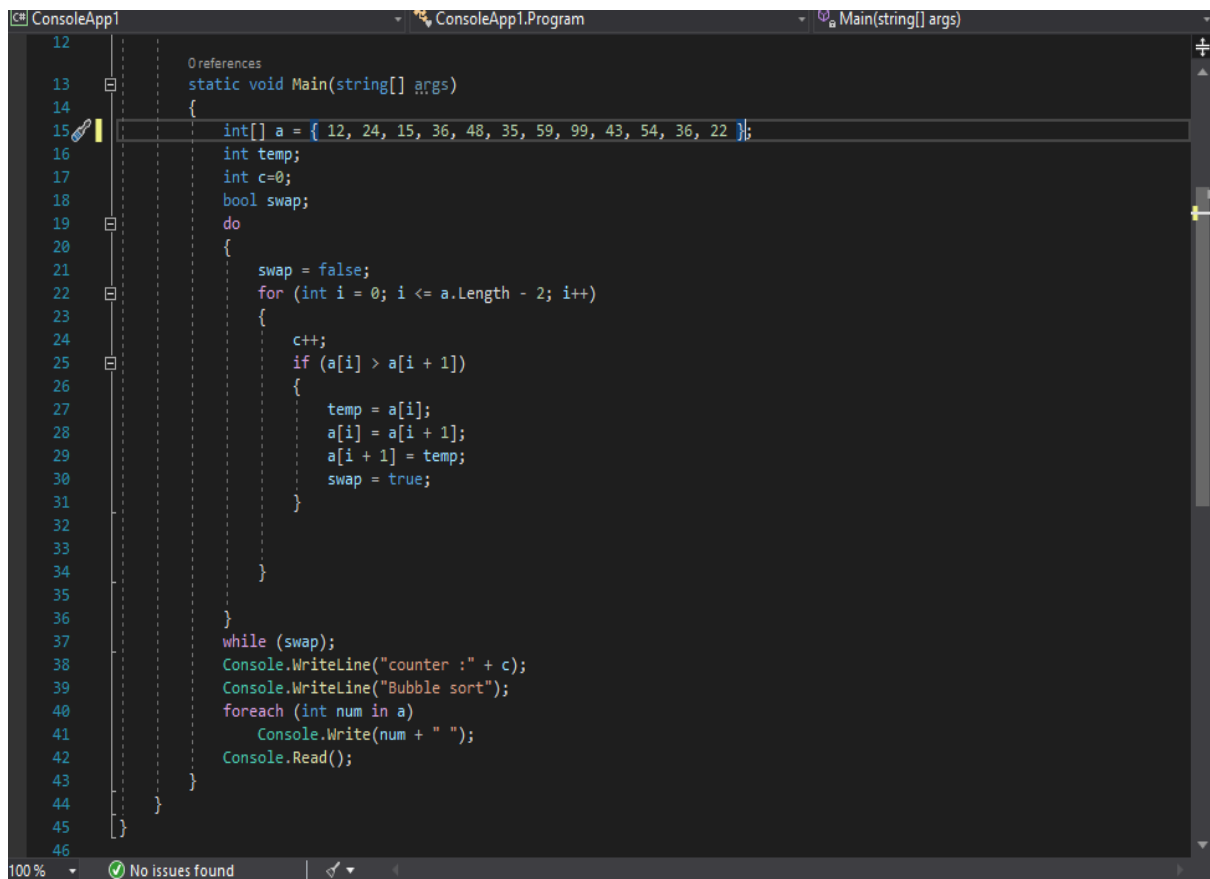
## Output:



```
Program.cs | ConsoleApp1 | Main(string[] args)
12
13 References
14 static void Main(string[] args)
15 {
16     int[] a = { 90,81,72,63,54,45,36,27,18,9 };
17     int temp;
18     int c=0;
19     bool swap;
20     do
21     {
22         counter :90
23         Bubble sort
24         9 18 27 36 45 54 63 72 81 90
25     }
26
27
28
29
30
31
32
100%
```

## TEST 3 :

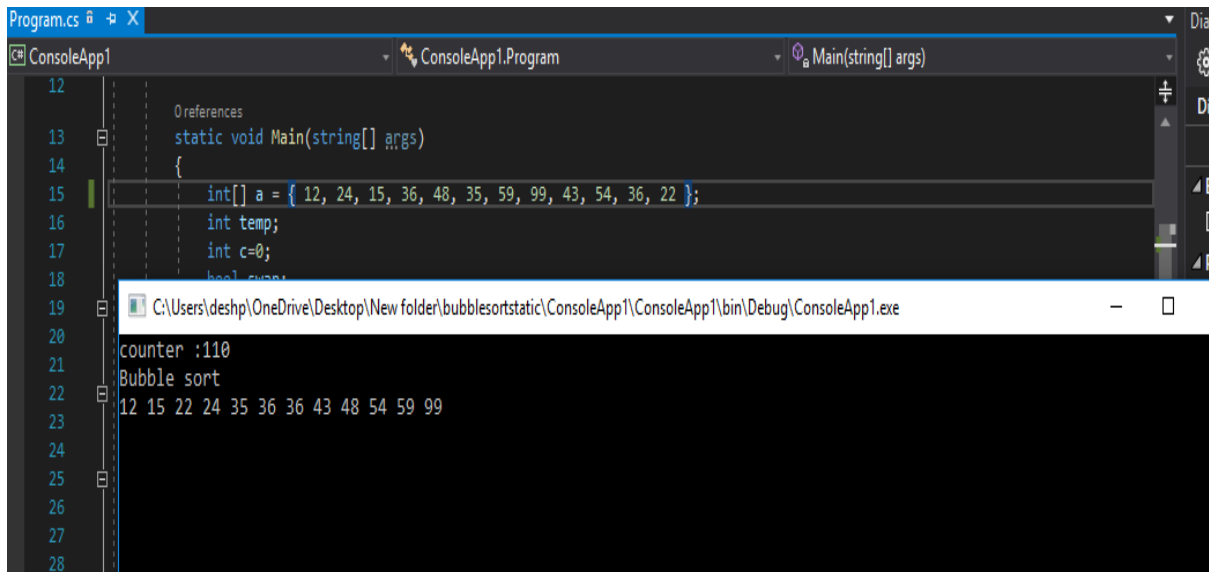
For average case scenario : input a []={12,24,15,36,48,35,59,99,43,54,36,22} size of array =12



```
ConsoleApp1 | ConsoleApp1.Program | Main(string[] args)
12
13 References
14 static void Main(string[] args)
15 {
16     int[] a = { 12, 24, 15, 36, 48, 35, 59, 99, 43, 54, 36, 22 };
17     int temp;
18     int c=0;
19     bool swap;
20     do
21     {
22         swap = false;
23         for (int i = 0; i <= a.Length - 2; i++)
24         {
25             c++;
26             if (a[i] > a[i + 1])
27             {
28                 temp = a[i];
29                 a[i] = a[i + 1];
30                 a[i + 1] = temp;
31                 swap = true;
32             }
33         }
34     }
35     while (swap);
36     Console.WriteLine("counter :" + c);
37     Console.WriteLine("Bubble sort");
38     foreach (int num in a)
39     {
40         Console.Write(num + " ");
41     }
42     Console.Read();
43 }
44
45
46
100% No issues found
```



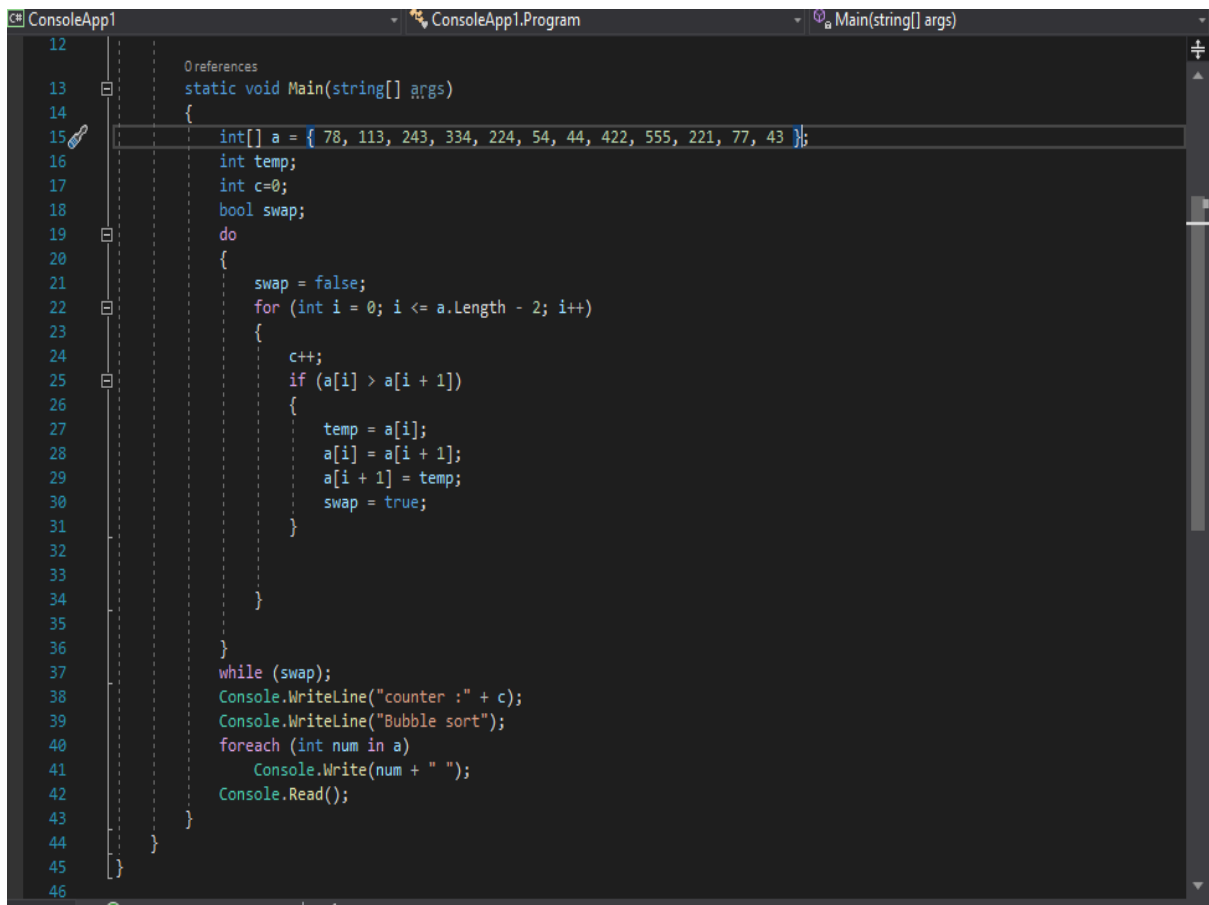
## Output:



```
Program.cs X
ConsoleApp1 ConsoleApp1.Program Main(string[] args)
12
13 static void Main(string[] args)
14 {
15     int[] a = { 12, 24, 15, 36, 48, 35, 59, 99, 43, 54, 36, 22 };
16     int temp;
17     int c=0;
18     bool swap;
19
20     counter :110
21     Bubble sort
22     12 15 22 24 35 36 36 43 48 54 59 99
23
24
25
26
27
28
```

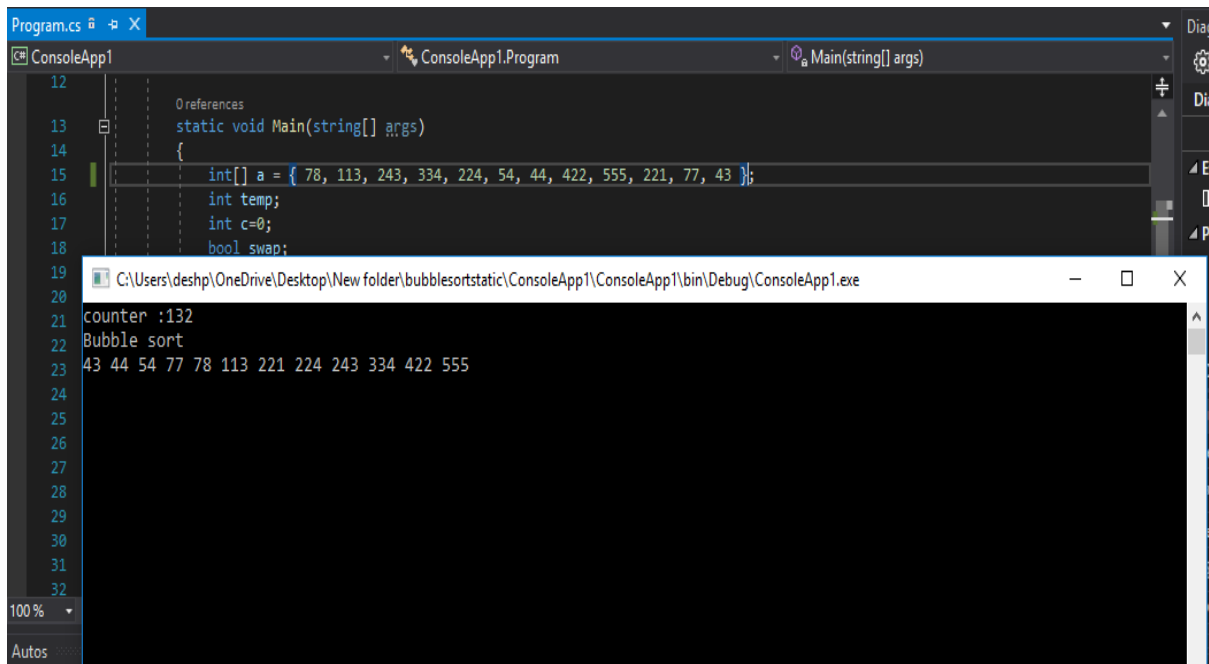
## TEST 4:

For average case : input array []={78,113,243,334,224,54,44,422,555,221,77,43} size of array :12



```
ConsoleApp1 ConsoleApp1.Program Main(string[] args)
12
13 static void Main(string[] args)
14 {
15     int[] a = { 78, 113, 243, 334, 224, 54, 44, 422, 555, 221, 77, 43 };
16     int temp;
17     int c=0;
18     bool swap;
19     do
20     {
21         swap = false;
22         for (int i = 0; i <= a.Length - 2; i++)
23         {
24             c++;
25             if (a[i] > a[i + 1])
26             {
27                 temp = a[i];
28                 a[i] = a[i + 1];
29                 a[i + 1] = temp;
30                 swap = true;
31             }
32         }
33     } while (swap);
34
35     Console.WriteLine("counter :" + c);
36     Console.WriteLine("Bubble sort");
37     foreach (int num in a)
38     {
39         Console.Write(num + " ");
40     }
41     Console.Read();
42 }
43
44
45
46
```

Output:

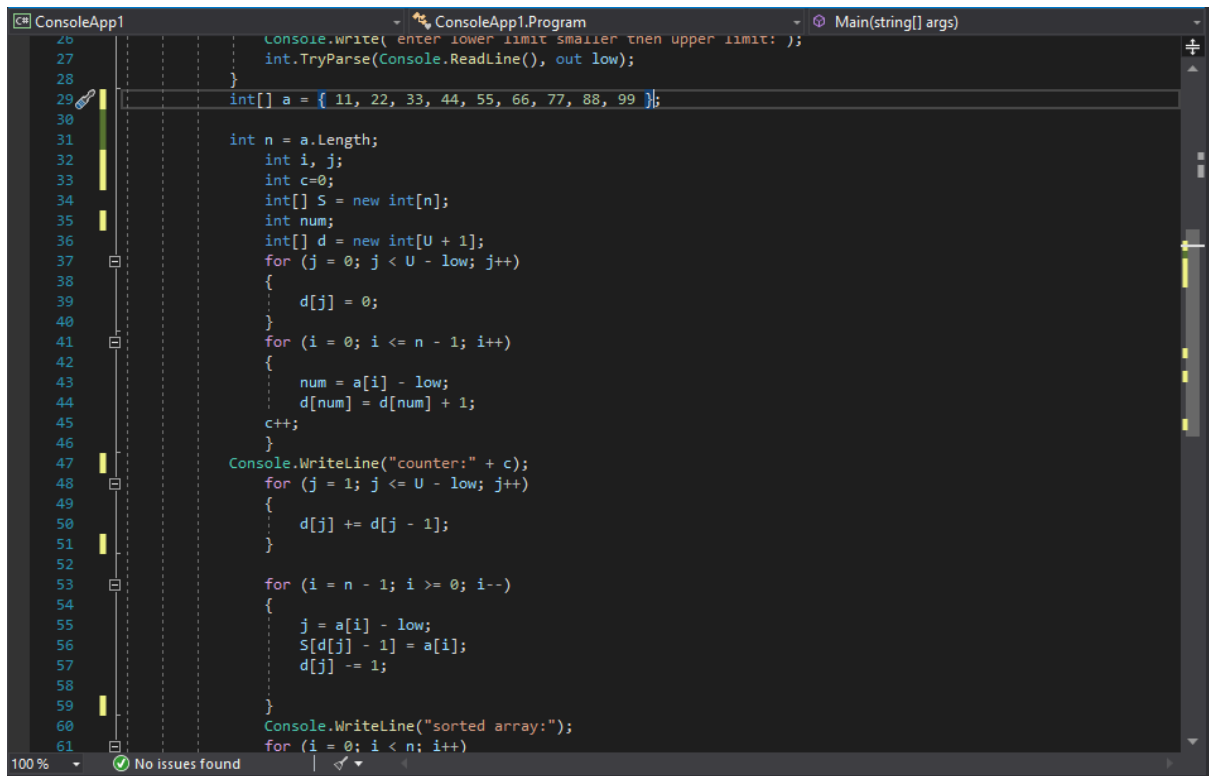


```
Program.cs 1 X
ConsoleApp1 ConsoleApp1.Program Main(string[] args)
12
13 static void Main(string[] args)
14 {
15     int[] a = { 78, 113, 243, 334, 224, 54, 44, 422, 555, 221, 77, 43 };
16     int temp;
17     int c=0;
18     bool swap;
19
20
21 counter :132
22 Bubble sort
23 43 44 54 77 78 113 221 224 243 334 422 555
24
25
26
27
28
29
30
31
32
100 %
Autos
```

## ALGORITHM 2: DISTRIBUTION COUNTING

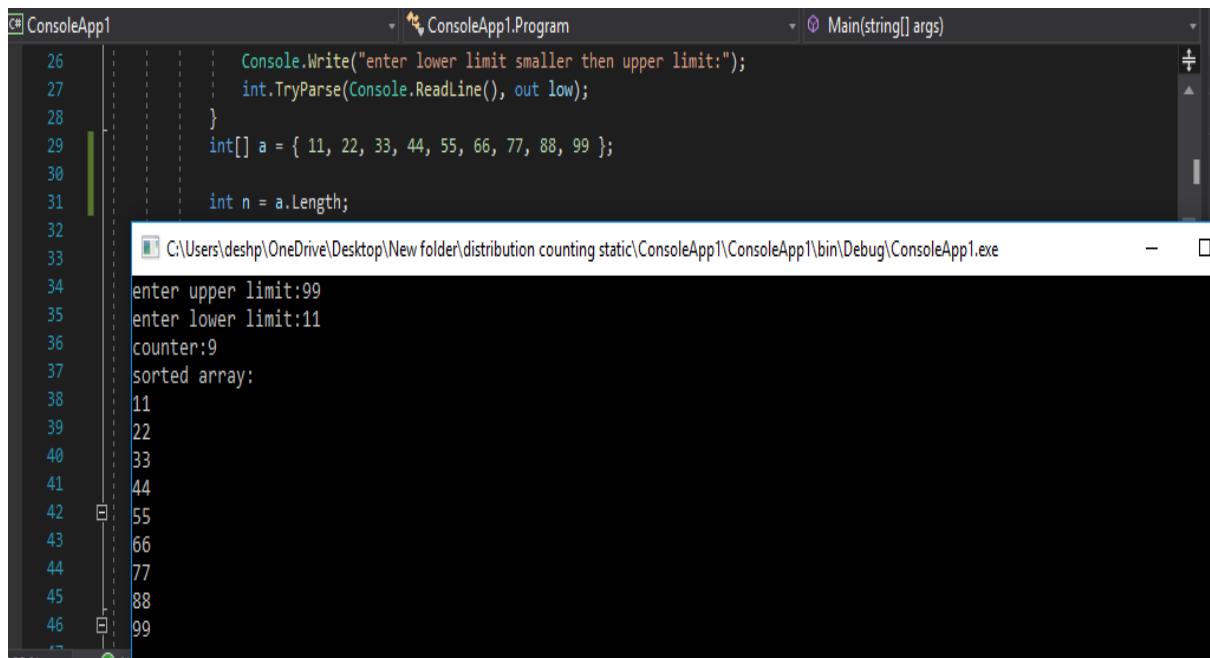
TEST1:

For best case: input a [] = {11,22,33,44,55,66,77,88,99} size of array =9



```
ConsoleApp1 ConsoleApp1.Program Main(string[] args)
26 Console.WriteLine( "enter lower limit smaller then upper limit: ");
27 int.TryParse(Console.ReadLine(), out low);
28
29 int[] a = { 11, 22, 33, 44, 55, 66, 77, 88, 99 };
30
31 int n = a.Length;
32 int i, j;
33 int c=0;
34 int[] S = new int[n];
35 int num;
36 int[] d = new int[U + 1];
37 for (j = 0; j < U - low; j++)
38 {
39     d[j] = 0;
40 }
41 for (i = 0; i <= n - 1; i++)
42 {
43     num = a[i] - low;
44     d[num] = d[num] + 1;
45     c++;
46 }
47 Console.WriteLine("counter:" + c);
48 for (j = 1; j <= U - low; j++)
49 {
50     d[j] += d[j - 1];
51 }
52
53 for (i = n - 1; i >= 0; i--)
54 {
55     j = a[i] - low;
56     S[d[j] - 1] = a[i];
57     d[j] -= 1;
58 }
59 Console.WriteLine("sorted array:");
60 for (i = 0; i < n; i++)
61 {
100 % No issues found
```

Output :



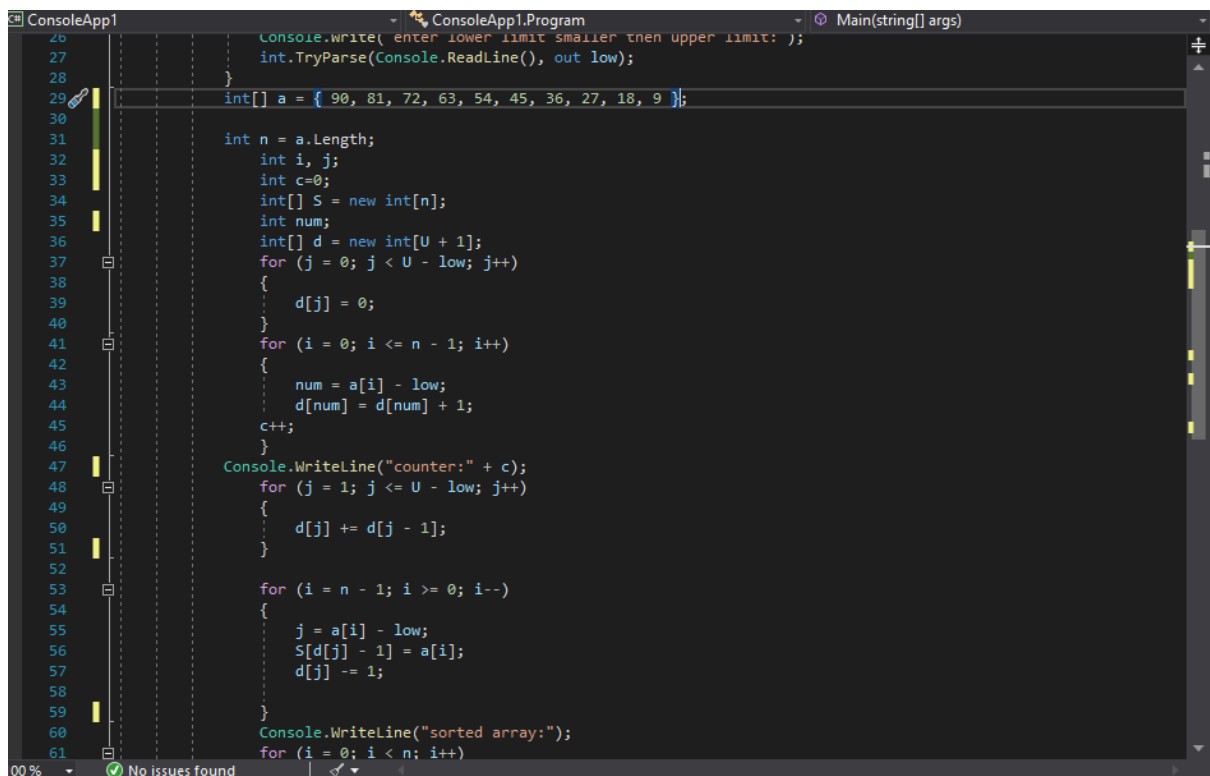
```
26 Console.WriteLine("enter lower limit smaller then upper limit:");
27 int.TryParse(Console.ReadLine(), out low);
28 }
29 int[] a = { 11, 22, 33, 44, 55, 66, 77, 88, 99 };
30
31 int n = a.Length;
```

C:\Users\deshp\OneDrive\Desktop\New folder\distribution counting static\ConsoleApp1\ConsoleApp1\bin\Debug\ConsoleApp1.exe

enter upper limit:99  
enter lower limit:11  
counter:9  
sorted array:  
11  
22  
33  
44  
55  
66  
77  
88  
99

TEST 2:

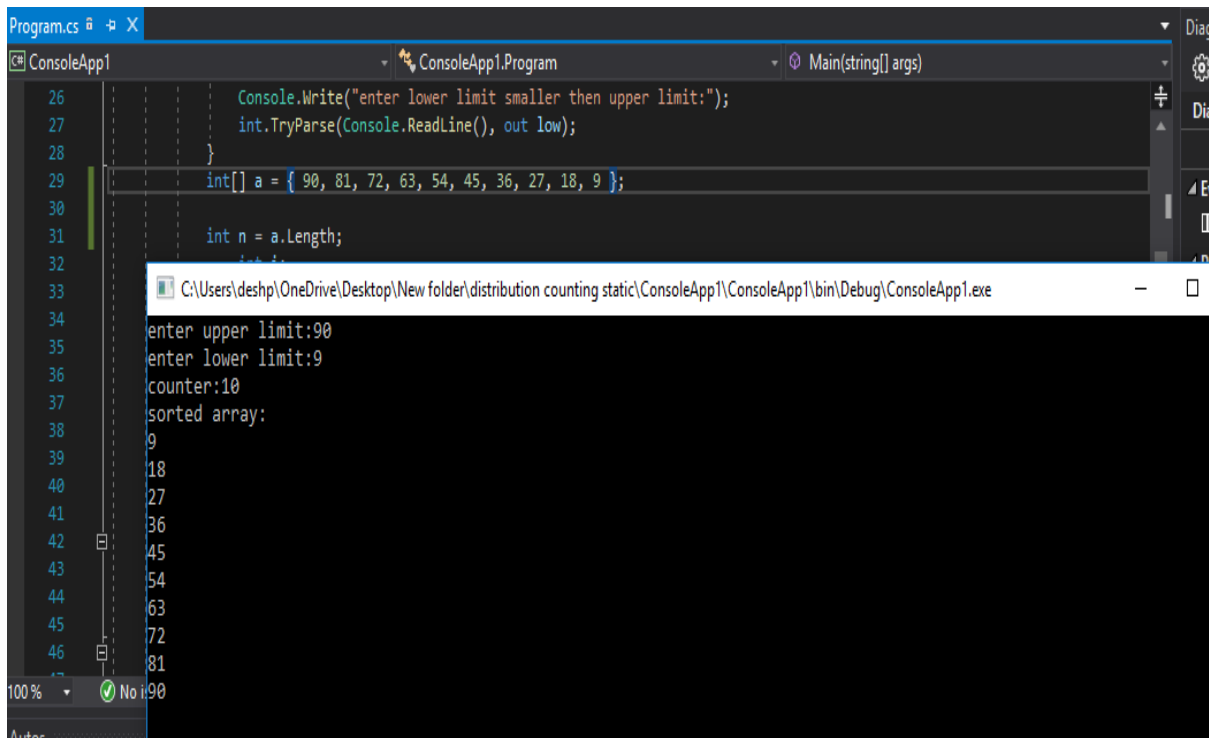
For worst case: input a [ ]={90,81,72,63,54,45,36,27,18,9} size of array =10



```
26 Console.WriteLine("enter lower limit smaller then upper limit: ");
27 int.TryParse(Console.ReadLine(), out low);
28 }
29 int[] a = { 90, 81, 72, 63, 54, 45, 36, 27, 18, 9 };
30
31 int n = a.Length;
32 int i, j;
33 int c=0;
34 int[] S = new int[n];
35 int num;
36 int[] d = new int[U + 1];
37 for (j = 0; j < U - low; j++)
38 {
39     d[j] = 0;
40 }
41 for (i = 0; i <= n - 1; i++)
42 {
43     num = a[i] - low;
44     d[num] = d[num] + 1;
45     c++;
46 }
47 Console.WriteLine("counter:" + c);
48 for (j = 1; j <= U - low; j++)
49 {
50     d[j] += d[j - 1];
51 }
52
53 for (i = n - 1; i >= 0; i--)
54 {
55     j = a[i] - low;
56     S[d[j] - 1] = a[i];
57     d[j] -= 1;
58 }
59
60 Console.WriteLine("sorted array:");
61 for (i = 0; i < n; i++)
```

00% No issues found

## Output:



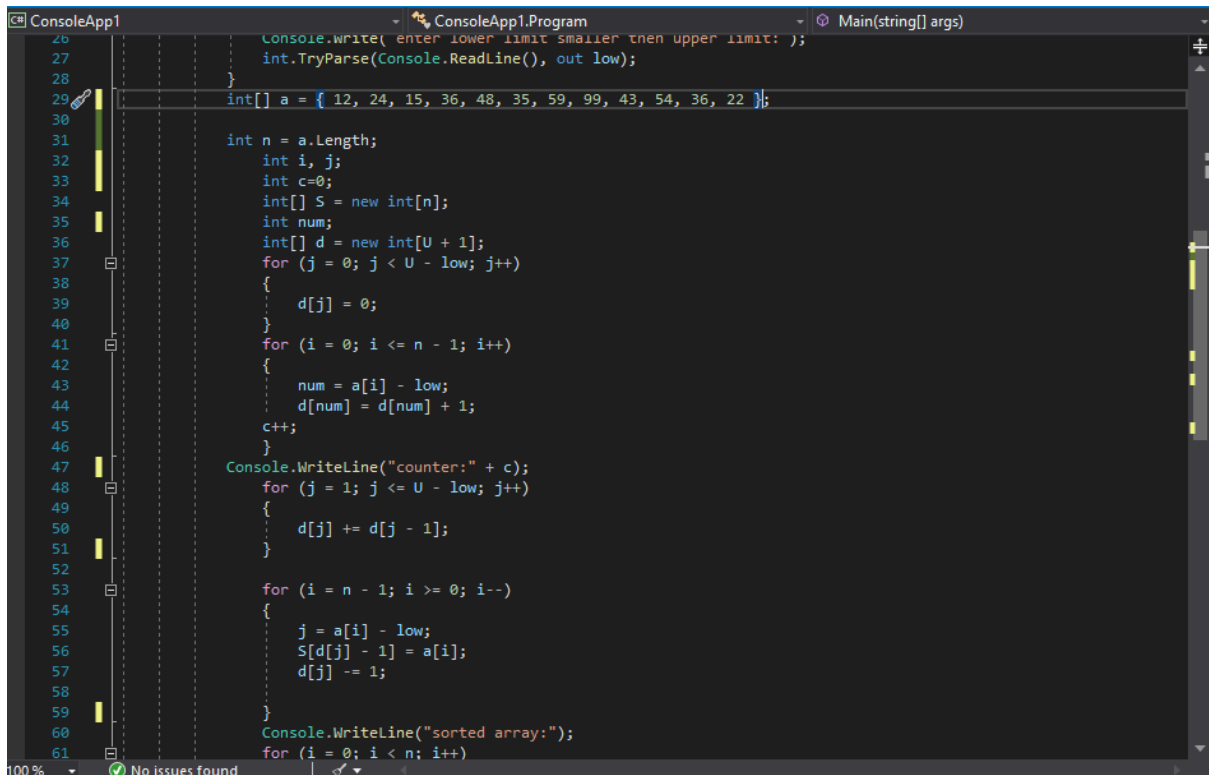
```
Program.cs X
ConsoleApp1 ConsoleApp1.Program Main(string[] args)
26 Console.WriteLine("enter lower limit smaller then upper limit:");
27 int.TryParse(Console.ReadLine(), out low);
28 }
29 int[] a = { 90, 81, 72, 63, 54, 45, 36, 27, 18, 9 };
30
31 int n = a.Length;
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
100% No issues found
```

C:\Users\deshp\OneDrive\Desktop\New folder\distribution counting static\ConsoleApp1\ConsoleApp1\bin\Debug\ConsoleApp1.exe

enter upper limit:90  
enter lower limit:9  
counter:10  
sorted array:  
9  
18  
27  
36  
45  
54  
63  
72  
81  
90

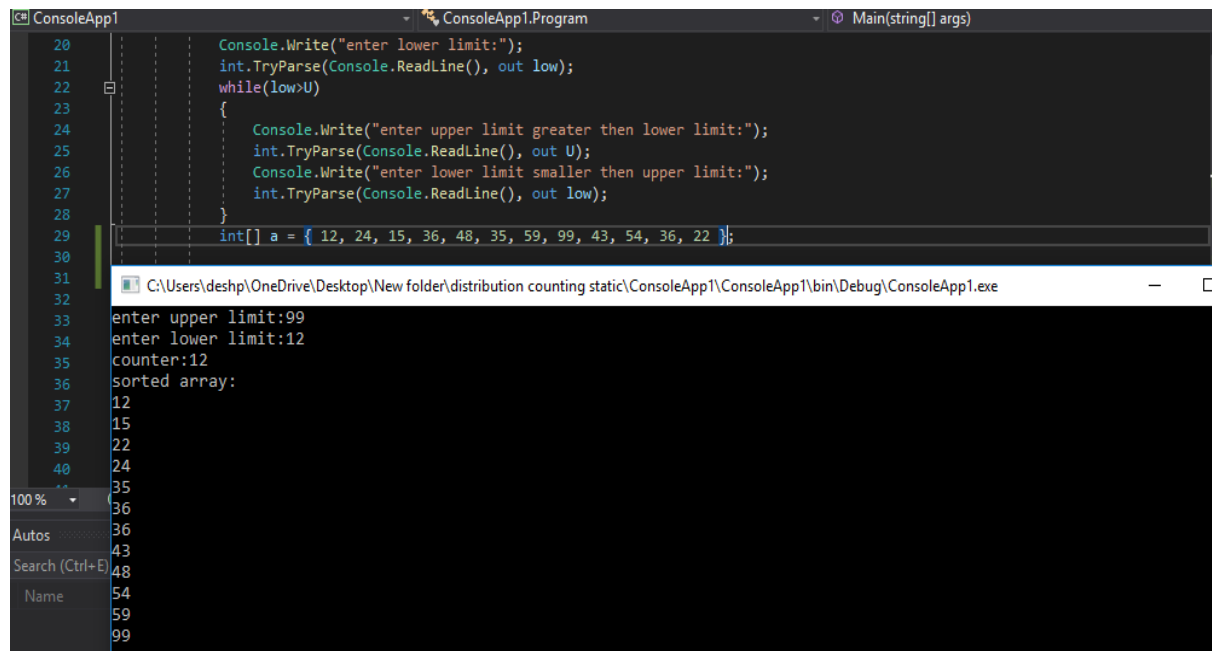
## TEST 3 :

For average case scenario : input a []={12,24,15,36,48,35,59,99,43,54,36,22} size of array =12



```
ConsoleApp1 ConsoleApp1.Program Main(string[] args)
26 Console.WriteLine("enter lower limit smaller then upper limit: ");
27 int.TryParse(Console.ReadLine(), out low);
28 }
29 int[] a = { 12, 24, 15, 36, 48, 35, 59, 99, 43, 54, 36, 22 };
30
31 int n = a.Length;
32 int i, j;
33 int c=0;
34 int[] S = new int[n];
35 int num;
36 int[] d = new int[U + 1];
37 for (j = 0; j < U - low; j++)
38 {
39     d[j] = 0;
40 }
41 for (i = 0; i <= n - 1; i++)
42 {
43     num = a[i] - low;
44     d[num] = d[num] + 1;
45     c++;
46 }
47 Console.WriteLine("counter:" + c);
48 for (j = 1; j <= U - low; j++)
49 {
50     d[j] += d[j - 1];
51 }
52
53 for (i = n - 1; i >= 0; i--)
54 {
55     j = a[i] - low;
56     S[d[j] - 1] = a[i];
57     d[j] -= 1;
58 }
59 Console.WriteLine("sorted array:");
60 for (i = 0; i < n; i++)
61 {
62 }
100% No issues found
```

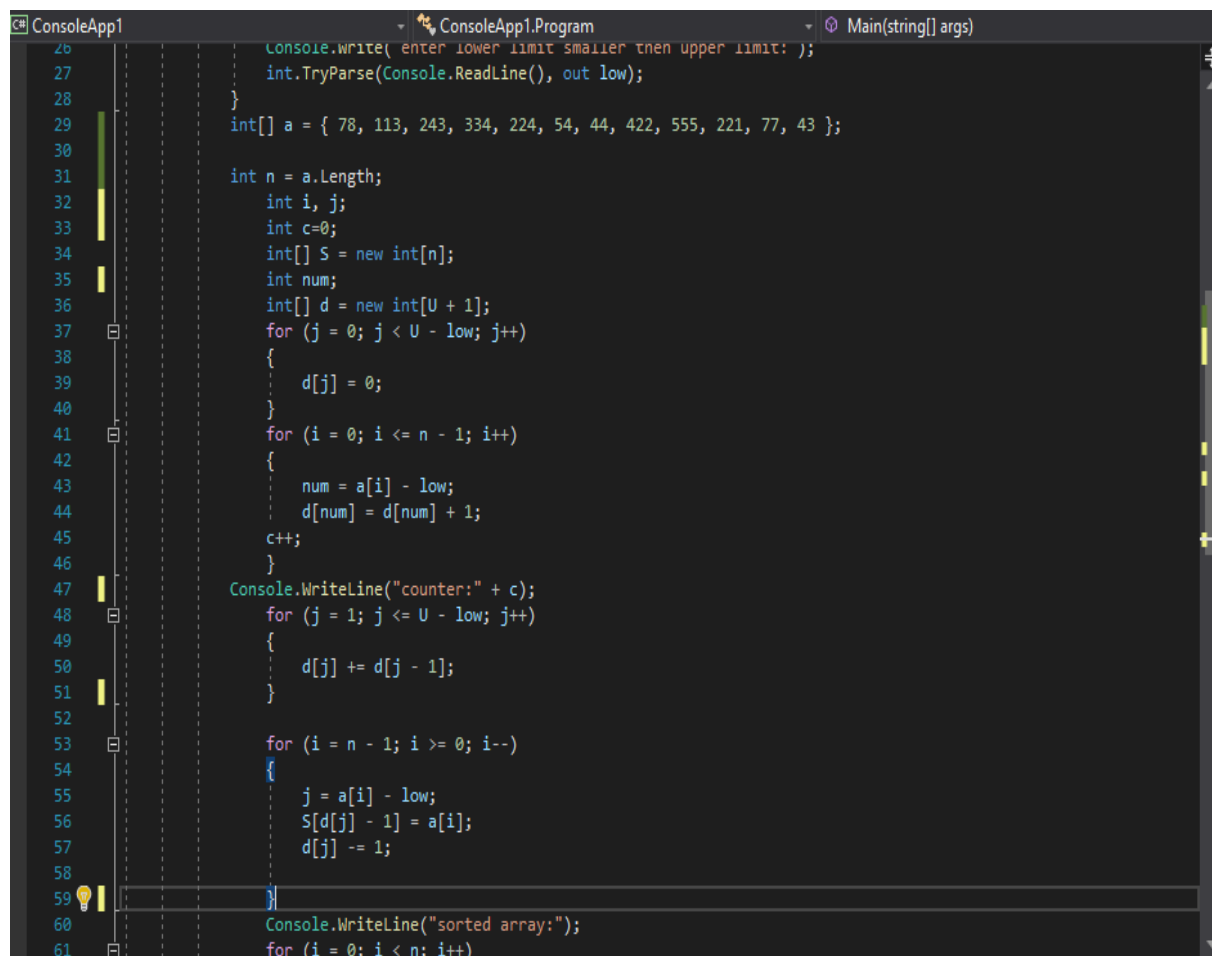
## Output:



The screenshot shows a C# console application named 'ConsoleApp1'. The code defines a method 'Main' that prompts the user for a lower limit and an upper limit, ensuring the lower limit is smaller than the upper limit. It then defines an array 'a' with 12 elements: {12, 24, 15, 36, 48, 35, 59, 99, 43, 54, 36, 22}. The output shows the user entering '99' for the upper limit and '12' for the lower limit. The program then prints the sorted array: 12, 15, 22, 24, 35, 36, 43, 48, 54, 59, 99.

```
20 Console.WriteLine("enter lower limit:");
21 int.TryParse(Console.ReadLine(), out low);
22 while(low > U)
23 {
24     Console.WriteLine("enter upper limit greater then lower limit:");
25     int.TryParse(Console.ReadLine(), out U);
26     Console.WriteLine("enter lower limit smaller then upper limit:");
27     int.TryParse(Console.ReadLine(), out low);
28 }
29 int[] a = { 12, 24, 15, 36, 48, 35, 59, 99, 43, 54, 36, 22 };
30
31
32
33 enter upper limit:99
34 enter lower limit:12
35 counter:12
36 sorted array:
37 12
38 15
39 22
40 24
41 35
42 36
43 43
44 48
45 54
46 59
47 99
```

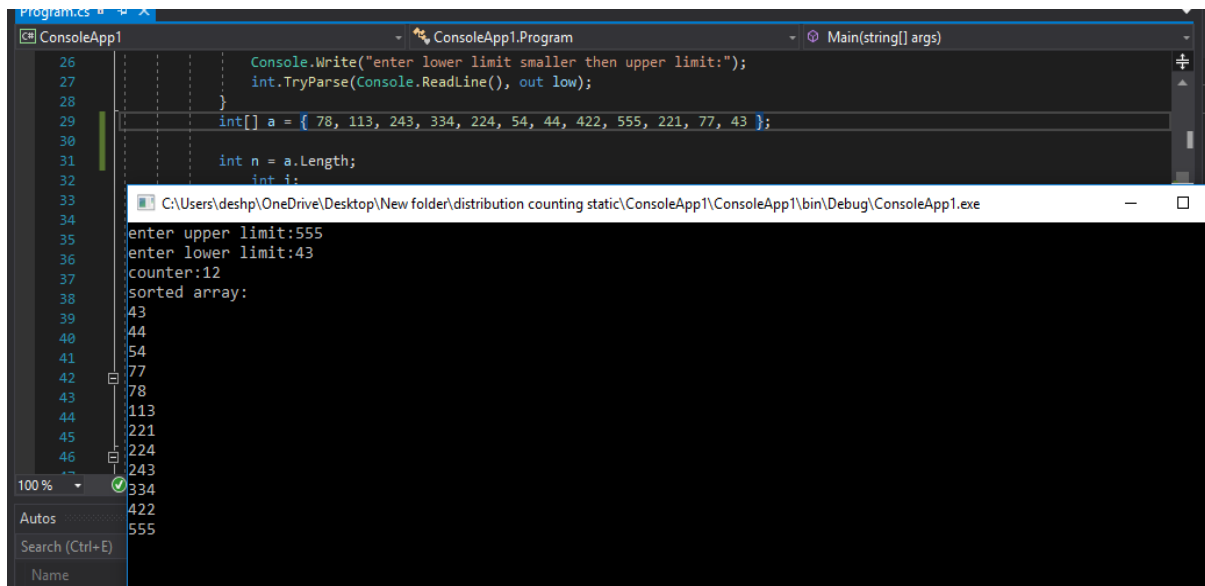
For average case : input array []={78,113,243,334,224,54,44,422,555,221,77,43} size of array :12



The screenshot shows the same C# console application, but with a different input array 'a' for the average case: {78, 113, 243, 334, 224, 54, 44, 422, 555, 221, 77, 43}. The code calculates the length of the array 'n' and initializes a counter 'c'. It then uses a counting sort algorithm to sort the array. The output shows the sorted array: 43, 77, 113, 221, 243, 334, 422, 555, 78, 113, 243, 334.

```
26 Console.WriteLine("enter lower limit smaller then upper limit: ");
27 int.TryParse(Console.ReadLine(), out low);
28 }
29 int[] a = { 78, 113, 243, 334, 224, 54, 44, 422, 555, 221, 77, 43 };
30
31 int n = a.Length;
32 int i, j;
33 int c=0;
34 int[] S = new int[n];
35 int num;
36 int[] d = new int[U + 1];
37 for (j = 0; j < U - low; j++)
38 {
39     d[j] = 0;
40 }
41 for (i = 0; i <= n - 1; i++)
42 {
43     num = a[i] - low;
44     d[num] = d[num] + 1;
45     c++;
46 }
47 Console.WriteLine("counter:" + c);
48 for (j = 1; j <= U - low; j++)
49 {
50     d[j] += d[j - 1];
51 }
52
53 for (i = n - 1; i >= 0; i--)
54 {
55     j = a[i] - low;
56     S[d[j] - 1] = a[i];
57     d[j] -= 1;
58 }
59
60 Console.WriteLine("sorted array:");
61 for (i = 0; i < n; i++)
```

## OUTPUT



```
Program.cs - ConsoleApp1
ConsoleApp1
ConsoleApp1.Program
Main(string[] args)
26 Console.WriteLine("enter lower limit smaller then upper limit:");
27 int.TryParse(Console.ReadLine(), out low);
28
29 int[] a = { 78, 113, 243, 334, 224, 54, 44, 422, 555, 221, 77, 43 };
30
31 int n = a.Length;
32 int i;
33
34
35 enter upper limit:555
36 enter lower limit:43
37 counter:12
38 sorted array:
39 43
40 44
41 54
42 77
43 78
44 113
45 221
46 224
47 243
48 334
49 422
50 555
```

Thus the experimental finding shows the same thing which we have find out in theortical finding. when the data we tests average case ,best case , and worst case for buublesort we get different counter depending on the input and basic operation performed .but when we do best case ,average case and worst case scenrio in distribution counting we get counter depending on the length of an array that is the basic operation performed is equal to the length of an array in input.

## EFFECENCY TESTING

For Efficiency Testing I have implemented the code to generate array of random numbers of range from 0 to 50 and execution timings are in Milliseconds.

To generate the random numbers, and to calculate Execution time I have used the following code in both the algorithms.

For bubble sort algorithm:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Diagnostics;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            int U;
            int low;
            Console.WriteLine("enter upper limit:");
            int.TryParse(Console.ReadLine(), out U);
            Console.WriteLine("enter lower limit:");
```

```

int.TryParse(Console.ReadLine(), out low);
while (low > U)
{
    Console.WriteLine("enter upper limit greater than lower limit:");
    int.TryParse(Console.ReadLine(), out U);
    Console.WriteLine("enter lower limit smaller than upper limit:");
    int.TryParse(Console.ReadLine(), out low);
}

for (int r = 1; r <= 50; r++)
{
    int[] bubblesort = new int[r];
    Random R = new Random();

    int i;
    for (i = 0; i < bubblesort.Length; i++)
    {
        bubblesort[i] = R.Next(low, U);
    }

    Stopwatch timer = new Stopwatch();

    timer.Start();
    int c=0;
    int temp;
    bool swap;
    do
    {
        swap = false;
        for (i = 0; i <= bubblesort.Length - 2; i++)
        {
            c++;
            if (bubblesort[i] > bubblesort[i + 1])
            {
                temp = bubblesort[i];
                bubblesort[i] = bubblesort[i + 1];
                bubblesort[i + 1] = temp;
                swap = true;
            }
        }
    }
    while (swap);
    Console.WriteLine(" " + c);
    /* Console.WriteLine("Bubble sort");
    foreach (int num in bubblesort)
        Console.Write(num + " ");
    Console.Read();*/

    timer.Stop();
    Console.WriteLine("{0};{1} " ,r,timer.Elapsed);
    timer.Reset();
    Console.ReadKey();
}
}

```

```
}  
}
```

For distribution counting:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Diagnostics;  
  
namespace ConsoleApp2  
{  
    class Program  
    {  
  
        public static void Main(string[] args)  
        {  
            int U;  
            int low;  
            Console.Write("enter upper limit:");  
            int.TryParse(Console.ReadLine(), out U);  
            Console.Write("enter lower limit:");  
            int.TryParse(Console.ReadLine(), out low);  
            while (low > U)  
            {  
                Console.Write("enter upper limit greater then lower limit:");  
                int.TryParse(Console.ReadLine(), out U);  
                Console.Write("enter lower limit smaller then upper limit:");  
                int.TryParse(Console.ReadLine(), out low);  
            }  
            for (int r = 1; r <= 50; r++)  
            {  
                int[] distcount = new int[r];  
                Random R = new Random();  
  
                int c = 0;  
                int i;  
                for (i = 0; i < distcount.Length; i++)  
                {  
                    distcount[i] = R.Next(low, U);  
                }  
                /*for (i = 0; i < distcount.Length; i++)  
                {  
                    Console.Write(distcount[i].ToString() + " ");  
                }*/  
  
                int j;  
                //int low,up;  
  
                int[] S = new int[distcount.Length];  
                int num;  
                int[] d = new int[U + 1];  
                Stopwatch timer = new Stopwatch();
```



```

timer.Start();
for (j = 0; j < U - low; j++)
{
    d[j] = 0;
}

for (i = 0; i <= distcount.Length - 1; i++)
{
    num = distcount[i] - low;
    d[num] = d[num] + 1;
    c++;
}
/* Console.WriteLine("counte :" + c);

/* for (i = 0; i < (U - low) + 1; i++)
{
    Console.WriteLine(d[i].ToString() + " ");
}*/

Console.WriteLine();
for (j = 1; j <= U - low; j++)
{
    d[j] += d[j - 1];
}

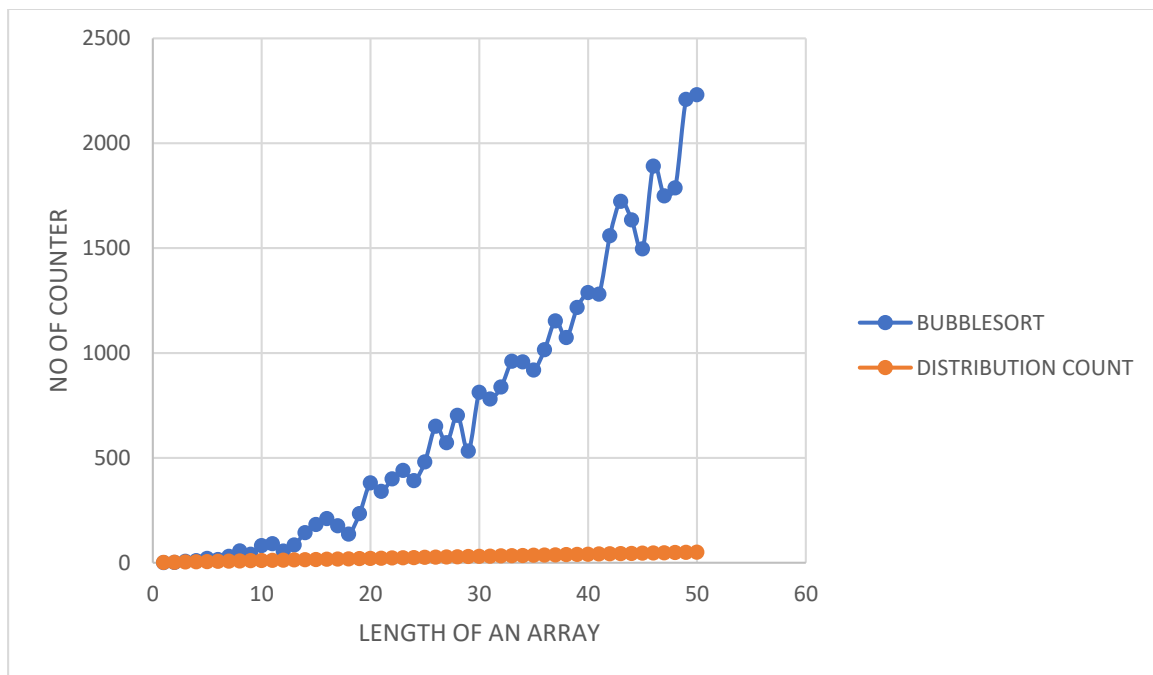
/*for (i = 0; i < (U - low) + 1; i++)
{
    Console.WriteLine(d[i].ToString() + " ");
}*/

for (i = distcount.Length - 1; i >= 0; i--)
{
    j = distcount[i] - low;
    S[d[j] - 1] = distcount[i];
    d[j] -= 1;
}
/* Console.WriteLine("sorted array :");
for (i = 0; i < distcount.Length; i++)
{
    Console.Write(S[i].ToString() + " ");
}*/
timer.Stop();
Console.WriteLine("{0}          ", timer.Elapsed);
timer.Reset();
}
Console.ReadLine();
Console.ReadKey();
}
}
}

```

## GRAPH 1: NUMBER OF TIMES BASIC OPERATION

The graph shows that number of time the basic operation is performed in bubble sort and distribution counting vs the length of array. It shows the same thing which is derived in theoretical calculation



## GRAPH 2 : EXECUTION TIME

It displays the time taken by bubblesort algorithm and distribution count algorithm for the different length for an array.



## PROGRAM FOR STATIC:

### FOR BUBBLESORT:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] a = { 11, 22, 33, 44, 55, 66, 77, 88, 99 };
            int temp;
            int c=0;
            bool swap;
            do
            {
                swap = false;
                for (int i = 0; i <= a.Length - 2; i++)
                {
                    c++;
                    if (a[i] > a[i + 1])
                    {
                        temp = a[i];
                        a[i] = a[i + 1];
                        a[i + 1] = temp;
                        swap = true;
                    }
                }
            }
            while (swap);
            Console.WriteLine("counter :" + c);
            Console.WriteLine("Bubble sort");
            foreach (int num in a)
                Console.Write(num + " ");
            Console.Read();
        }
    }
}
```

## PRORAM FOR DISTRIBUTION COUNTING STATIC:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class Program
```

```
{
```

```
public static void Main(string[] args)
{
    int U;
    int low;
    Console.Write("enter upper limit:");
    int.TryParse(Console.ReadLine(), out U);
    Console.Write("enter lower limit:");
    int.TryParse(Console.ReadLine(), out low);
    while(low>U)
    {
        Console.Write("enter upper limit greater then lower limit:");
        int.TryParse(Console.ReadLine(), out U);
        Console.Write("enter lower limit smaller then upper limit:");
        int.TryParse(Console.ReadLine(), out low);
    }
    int[] a = { 11, 22, 33, 44, 55, 66, 77, 88, 99 };

    int n = a.Length;
    int i, j;
    int c=0;
    int[] S = new int[n];
    int num;
    int[] d = new int[U + 1];
    for (j = 0; j < U - low; j++)
    {
        d[j] = 0;
    }
    for (i = 0; i <= n - 1; i++)
    {
        num = a[i] - low;
        d[num] = d[num] + 1;
        c++;
    }
    Console.WriteLine("counter:" + c);
    for (j = 1; j <= U - low; j++)
    {
        d[j] += d[j - 1];
    }

    for (i = n - 1; i >= 0; i--)
    {
        j = a[i] - low;
        S[d[j] - 1] = a[i];
        d[j] -= 1;
    }
    Console.WriteLine("sorted array:");
    for (i = 0; i < n; i++)
    {
        Console.WriteLine(S[i].ToString() + " ");
    }
    Console.ReadKey();
}
```

```
}
}
```

}

### **Conclusion:**

Thus the report explains the theoretical and experimental finding of efficiency, best case, worst case for both BUBBLESORT AND DISTRIBUTION COUNTING ALGORITHM and compare both for same.

### **REFERENCES:**

Clark, D., & Sanders, J. (2011). *Beginning C# object-oriented programming*: Springer.