KIET Group of Institutions, Ghaziabad

Department of Computer Applications

(An ISO – 9001: 2015 Certified & 'A' Grade accredited Institution by NAAC)

Design and Analysis of Algorithm

RCA 352: Session 2020-21

DAA Lab

Experiment-No.1

Objective: Implement the COUNT sort algorithm to sort the given list of N numbers and plot graph.

Scheduled Date:	Compiled Date:	Submitted Date:
23/09/2020	23/09/2020	7/11/2020

Algorithm:

Counting-sort(A,B,K)

1. For I ← 0 to k

2. do C[i] ← 0

3. For j ← to length [A]

4. do C[A [j]] + 1

5. For I ← 1 to K

6. do C[i] ← C [i] + C [i – 1]

7. For j ← length [A] downto 1

8. do B[C [A [j]]] ← A [j]

9. C [A [j]] ← C[A [j]] – 1

```c
#include <stdio.h>
#include <stdlib.h>
int count = 0;
int sizeArray=0;
int ArraySize(int * Array)
{

  return (sizeof(Array)/sizeof(int));

}
int findMax(int a[], int n)
```

```
{
    int max = a[0];
    int i;
    for (i = 1; i < n; i++)
    {
        count++;
        if (a[i] > max)
        {
            max = a[i];
        }
    }
    return max;
}
void countSort(int a[], int c[], int m) //here n is size of array
{
    int i, j;
    // count++;
    // max=findMax(a,n);//find max element value into the original
array
    // count++;
    // c=(int*)malloc(sizeof(int)*(max+1));//dynamic memory from 1
to (max+1) = max element's size allocated
    count++;
    for (i = 0; i < m + 1; i++)
    {
        c[i] = 0; // initialize all element's value with zero
        count++;
    }
    count++;
    for (i = 0; i < sizeArray ; i++)
    {
```

```
        c[a[i]]++; //the element in terms of their values stored
into count array
        count++;
    }

    i = 0, j = 0;
    //value restored from count array to original array
    count++;
    while (j < m + 1) //check element up to max
    {
        count++;
        if (c[j] > 0)
        {
            a[i++] = j;
            count++;
            c[j]--;
            count++;
        }
        else
        {
            j++;
            count++;
        }
    }
}

int main()
{
    int a[100], i, n, *c, max;
    printf("enter the size");
    scanf("%d", &n);
    printf("enter the elements into the array");
```

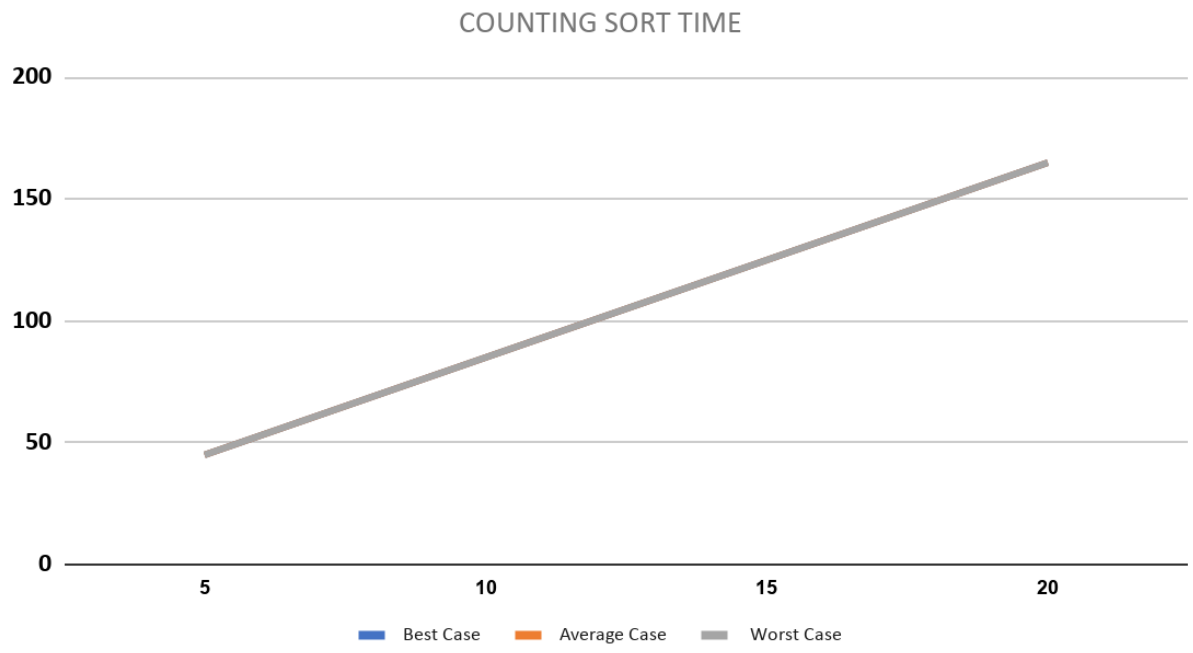
```
    for (i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
        sizeArray++;
    }

    printf("entered elements are :");
    for (i = 0; i < n; i++)
    {
        printf("%d ", a[i]);
    }

    max = findMax(a, n); //find max elemement value into the
original array
    c = (int *)malloc(sizeof(int) * (max + 1)); //dynamic memory
from 1 to (max+1) = max element's size allocated

    countSort(a,c,max); // calling counting sort
    printf("\nsorted elements are :");
    for (i = 0; i < n; i++)
    {
        printf("%d ", a[i]);
    }
    printf("Count = %d", count);
    return 0;
}
```

Design and Analysis of Algorithm
RCA 352: Session 2020-21
DAA Lab

Output

Input Size	Best Case	Average Case	Worst Case
5	45	45	45
10	85	85	85
15	125	125	125
20	165	165	165

Graph:



Conclusion

Case	Running Time : Growth of Function mathematically	Running Time : Growth of Function after observing graph
Best Case	O(n+m)	O(n+m)
Average Case	O(n+m)	O(n+m)
Worst Case	O(n+m)	O(n+m)

KIET Group of Institutions, Ghaziabad

Department of Computer Applications

(An ISO – 9001: 2015 Certified & 'A' Grade accredited Institution by NAAC)

Design and Analysis of Algorithm

RCA 352: Session 2020-21

DAA Lab