**PRACTICAL-7**

**AIM:Implement program of Counting Sort.**

**ALGORITHM:**

countingSort(array, n) max = find maximum element in the given array create count array

with size maximum + 1 Initialize count array

with all 0's

**for** i = 0 to n

find the count of every unique element and store that count at ith position in the count array **for** j =

1 to max

Now, find the cumulative sum and store it in count array **for** i = n to 1

Restore the array elements

Decrease the count of every restored element by 1

\end countingSort

**Code:**

#include <iostream> using namespace std; int getMax(int a[], int n) { int max = a[0]; for(int i

= 1; i<n; i++) { if(a[i]

> max)

max = a[i];

}

return max;

}

void countSort(int a[], int n)

{

int output[n+1]; int max = getMax(a, n); int count[max+1];

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

{

count[i] = 0;

}

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

{

count[a[i]]++;

}

for(int i = 1; i<=max; i++)

count[i] += count[i-1]; for (int i = n - 1; i >= 0; i--) { output[count[a[i]] - 1] = a[i];

count[a[i]]--;

}

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

a[i] = output[i];

}

}

void printArr(int a[], int n)

{ int i;

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

cout<<a[i]<<" ";

}

int main() { int a[] = { 31, 11, 42, 7, 30, 11 }; int n = sizeof(a)/sizeof(a[0]);

cout<<"Before sorting array elements are - \n"; printArr(a, n); countSort(a, n);

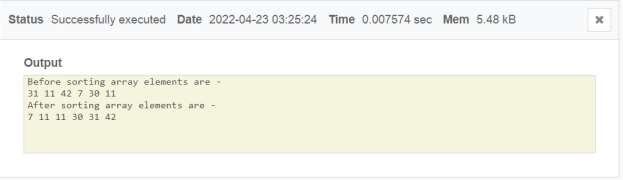
cout<<"\nAfter sorting array elements are - \n";

printArr(a, n);

return 0;

}

Output:



**ANALYSIS:**

|  |  |
| --- | --- |
| **LENGTH OF ARRAY** | **TIME COPLEXITY** |
| **5** | 0.007910 |
| **10** | 0.008162 |
| **15** | 0.007256 |
| **20** | 0.007163 |
| **25** | 0.009950 |

**Best Case Complexity -** It occurs when there is no sorting required, i.e. the array is already sorted. The best-case time complexity of counting sort is **O(n + k)**.

**Average Case Complexity -** It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending. The average case time complexity of counting sort is **O(n + k)**.

**Worst Case Complexity -** It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order. The worst-case time complexity of counting sort is **O(n + k).**

**CONCLUSION**:

**In this experiment we implemented and analyzed the counting sort and it’s Algorithm.**

**Further we analyzed it’s time complexity for worst , best and average case.**