 **Heaven’s Light is Our Guide**

**Rajshahi University of Engineering and Technology**

**Department of Computer Science and Engineering**

**Course No:** CSE.2202

**Course Title:** Sessional based on CSE.2201 (Computer Algorithms)

**Lab Report No:** 05

**Lab Report On:** Sorting in linear time: Counting Sort.

**Submitted By** **Submitted To**

Md. Ariful Islam Biprodip Pal

Roll No: 1803046 Assistant Professor

Section: A Dept. of CSE,RUET

Department: CSE

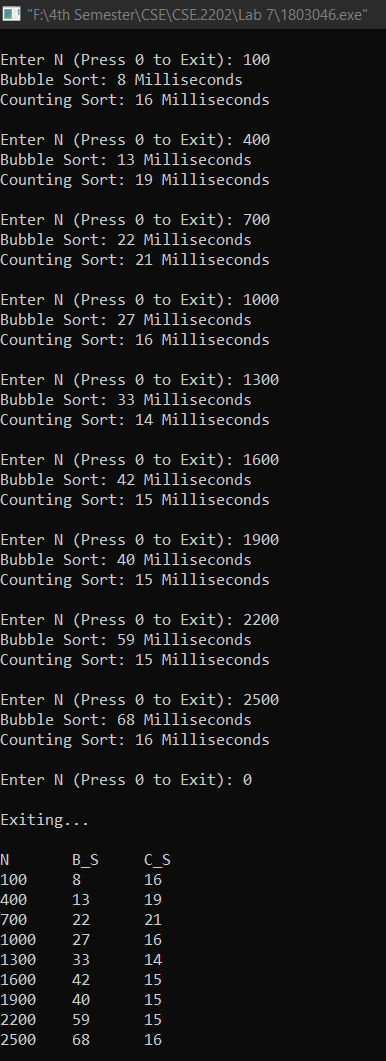
**Date: 26-06-2021**

* **Problem Statement:** The problem is observe sorting in linear time with counting sort. That requires

1. To generate **N** random integers within range 0 to 10000 in a file named **input.txt**.
2. To implement bubble sort to sort the numbers (from input.txt) and count the time.
3. To implement counting sort to sort the numbers (from input.txt) and count the time.
4. To increase the value of N and to plot the performance curve for sufficiently large N to see distinguishable performance.

* **Implemented Code:**

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;  using namespace std::chrono;  typedef long long ll;  #define M 10001  void menu(){  cout<<"\nEnter N (Press 0 to Exit): ";  }  int main(){  ll n,i,j;  vector<ll>cn,cb,cc;  while(1){  ll a;  menu();  cin>>a;  if(a<0) {  cout<<"Invalid Input"<<endl;  continue;  }  if(a==0){  cout<<"\nExiting..."<<endl;  break;  }  ll mx,mn,x;  vector<ll>bsort\_array,pos\_csort,csort\_array;  map<ll,ll>mp;  **//Creating File**  ofstream f1;  ifstream f2;  f1.open("input.txt");  n=a;  cn.push\_back(n);  srand(time(0));  x=rand()%M;  mx=x;  mn=x;  f1<<x;  csort\_array.push\_back(-1);  for(i=1;i<n;i++){  x=rand()%M;  f1<<" ";  f1<<x;  mx=max(mx,x);  mn=min(mn,x);  csort\_array.push\_back(-1);  }  f1.close();  **//Counting time for bubble sort**  auto start = high\_resolution\_clock::now();  f2.open("input.txt");  while(!f2.eof()){  f2>>x;  bsort\_array.push\_back(x);  }  f2.close();  **// Bubble sort**  for(i=0;i<n-1;i++){  for(j=i+1;j<n;j++){  if(bsort\_array[i]>bsort\_array[j]){  swap(bsort\_array[i],bsort\_array[j]);  }  }  }  auto stop = high\_resolution\_clock::now();  auto duration = duration\_cast<milliseconds>(stop - start);  cout<<"Bubble Sort: "<<duration.count()<<" Milliseconds"<<endl;  cb.push\_back(duration.count());  **//Counting time for counting sort**  start = high\_resolution\_clock::now();  f2.open("input.txt");  **//Counting sort**  while(!f2.eof()){  f2>>x;  mp[x]+=1;  }  f2.close();  for(i=mn;i<=mx;i++){  if(i==mn){  pos\_csort.push\_back(mp[i]);  continue;  }  pos\_csort.push\_back(mp[i]+pos\_csort[i-mn-1]);  }  f2.open("input.txt");  while(!f2.eof()){  f2>>x;  csort\_array[pos\_csort[x-mn]-1]=x;  pos\_csort[x-mn]-=1;  }  stop = high\_resolution\_clock::now();  duration = duration\_cast<milliseconds>(stop - start);  cout<<"Counting Sort: "<<duration.count()<<" Milliseconds"<<endl;  cc.push\_back(duration.count());  f2.close();  }  for(i=0;i<cn.size()-1;i++){  for(j=i+1;j<cn.size();j++){  if(cn[i]>cn[j]){  swap(cn[i],cn[j]);  swap(cb[i],cb[j]);  swap(cc[i],cc[j]);  }  }  }  cout<<"\nN\tB\_S\tC\_S"<<endl;  for(i=0;i<cn.size();i++){  cout<<cn[i]<<"\t"<<cb[i]<<"\t"<<cc[i]<<endl;  }  return 0;  } |

* **Output:**
* **Performance Curve of Bubble Sort & Counting Sort:**
* **Discussion & Conclusion:** From the output and graph we saw that for little value of **N** the sorting time of bubble sort was lower than counting sort. But as the value of **N** increases, the sorting time for bubble sort increases more than counting sort. For the higher value of **N** the Counting sort runs faster than the Bubble sort.

**# END #**