Stl

Unordered set and map

Its based on the hashing while set is based on the red black tree

There cant be duplicate element in the Unordered set

Q1)normal function usage(unordered set)

All this things take O(1) time while set uses o(logn) for searching

```
#include<bits/stdc++.h>
using namespace std;
int main(){
   unordered_set<int>s;
   int n,x,m,z;
   cin>>n;
   for(int i=0;i<n;i++){
      cin>>x;
      s.insert(x);
   }
   for(auto i:s){
      cout<<i;
   }
   cout<<endl;//first method printing
   for(auto it=s.begin();it!=s.end();it++){
      cout<<*it;
   }
   cout<<endl;//second printing method
   cin>>m;
    if(s.find(m)==s.end())//find function
      cout<<"Not Found";
   else
      cout<<"Found "<<(*s.find(m));
   if(s.count(m))//count function
      cout<<"Found";
   else
      cout<<"Not Found";
   cout<<endl;
   cout<<s.size();//size of the function

   auto it=s.find(m);
   s.erase(it);//erase a part of it
   cout<<s.size();
   s.erase(s.begin(),s.end());//erase all

}
```

Q2)normal function usage(map)

```
#include<bits/stdc++.h>
using namespace std;
int main(){
   unordered_map<string,int>m;
   int n,z,y;
   string x,o;
   cin>>n;
   for(int i=0;i<n;i++){
      cin>>x>>y;
      m.insert({x,y});
   }
   for(auto i:m){
      cout<<i.first<<" "<<i.second;
   }
   cout<<endl;//first method printing
   for(auto it=m.begin();it!=m.end();it++){
      cout<<(it->first)<<" "<<(it->second);
   }
   cout<<endl;//second printing method
   cin>>o;
```

```
    auto it=m.find(o);
     if(it!=m.end())
        cout<<(it->second);//find function

    if(m.count(o)>0)
        cout<<"Found";
    else
        cout<<"Not Found";
    cout<<endl;
    cout<<m.size();//size of the function

    auto it2=m.find("o");
    m.erase(it);//erase a part of it
    cout<<m.size();
    m.erase(m.begin(),m.end());//erase all

}
```

Q3) count distinct elements

```
#include<bits/stdc++.h>
using namespace std;
int countDistinct(vector<int>arr, int n){
    unordered_set<int> us;
    for(int i = 0; i < n; i++)
      us.insert(arr[i]);

    return us.size();
}

int main(){
    int a,b,c,m,n;
    cin>>n;
    int temp;
    vector<int>v;
    for(int i=0;i<n;i++){
        cin>>temp;
        v.push_back(temp);
    }
    cout<<countDistinct(v,n);
}
```

Q4) count frequencies

```
#include<bits/stdc++.h>
using namespace std;
void countFreq(vector<int>&v, int n){
    unordered_map<int,int>h;
    for(int i=0;i<n;i++){
        h[v[i]]++;
    }
    for(auto i:h){
     cout<<i.first<<" "<<i.second;
    }
}

int main(){
    int a,b,c,m,n;
    cin>>n;
    int temp;
    vector<int>v;
    for(int i=0;i<n;i++){
        cin>>temp;
        v.push_back(temp);
    }
   countFreq(v,n);
}
```

Q5)intersection of two sorted arrays

```
#include<bits/stdc++.h>
using namespace std;
```

```
void
intersection(vector<int>&arr1,vector<int>&arr2,in
t n,int m) {
   int sum=m;
   unordered_set<int>us(arr1.begin(),arr1.end());
   for(int i = 0; i < m; i++){
      if(us.find(arr2[i])!=us.end()){
         cout<<arr2[i];
      }



}
}

int main(){
   int a,b,c,m,n;
   cin>>n;
   cin>>m;//given sum
   int temp,temp2;
   vector<int>v1;
   vector<int>v2;
   for(int i=0;i<n;i++){
      cin>>temp;
      v1.push_back(temp);
   }
    for(int i=0;i<m;i++){
      cin>>temp2;
      v2.push_back(temp2);
   }


  intersection(v1,v2,n,m);
}
```

Q6)union of two sorted arrays

```
#include<bits/stdc++.h>
using namespace std;
int
unionsize(vector<int>&arr1,vector<int>&arr2,int
m,int n) {
   unordered_set<int> us;
   for(int i = 0; i < m; i++)
    us.insert(arr1[i]);
   for(int i = 0; i < n; i++)
    us.insert(arr2[i]);


   return us.size();
}

int main(){
   int a,b,c,m,n;
   cin>>n;
   cin>>m;
   int temp;
   vector<int>v1;
   vector<int>v2;
   for(int i=0;i<n;i++){
      cin>>temp;
      v1.push_back(temp);
   }
   for(int i=0;i<m;i++){
      cin>>temp;
      v2.push_back(temp);
   }
  cout<<unionsize(v1,v2,n,m);
}
```

Q7) pair with given sum in unsorted array

```
#include<bits/stdc++.h>
using namespace std;
bool givensum(vector<int>&arr,int n,int m) {
```

```
   int sum=m;
   unordered_set<int> us;
   for(int i = 0; i < n; i++){
      if(us.find(sum-arr[i])!=us.end()){
         return true;
      }else{
us.insert(arr[i]);
}


}
return false;
}

int main(){
   int a,b,c,m,n;
   cin>>n;
   cin>>m;//given sum
   int temp;
   vector<int>v1;
   vector<int>v2;
   for(int i=0;i<n;i++){
      cin>>temp;
      v1.push_back(temp);
   }

  cout<<givensum;
}
```

Q8)subarray with given sum

```
#include<bits/stdc++.h>
using namespace std;
int sumi(vector<int>&arr,int n,int sum) {
   unordered_set<int> s;
   int pre_sum = 0;
   for(int i = 0; i < n; i++)
   {
      pre_sum += arr[i];
      if(pre_sum==sum)
         return true;
      if(s.find(pre_sum-sum) != s.end())
        return true;
      s.insert(pre_sum);
   }
   return false;
}

int main(){
   int a,b,c,m,n,sum;
   cin>>n;
   cin>>m;
   cin>>sum;
   int temp;
   vector<int>v1;
   vector<int>v2;
   for(int i=0;i<n;i++){
      cin>>temp;
      v1.push_back(temp);
   }

  cout<<sumi(v1,n,sum);
}
```

Q9)longest subarray with the given sum

```
#include<bits/stdc++.h>
using namespace std;
int sumi(vector<int>&arr,int n,int sum) {
   int prefix_sum = 0, res = 0;
```

```
    unordered_map<int, int> m;
    for(int i = 0; i < n; i++) {

        prefix_sum += arr[i];
        if(prefix_sum == sum)
            res = i + 1;
        if(m.find(prefix_sum) == m.end())
            m.insert({prefix_sum, i});
        if(m.find(prefix_sum - sum) != m.end())
        res = max(res, i-m[prefix_sum-sum]);
        }
        return res;
        }


int main(){
    int a,b,c,m,n,sum;
    cin>>n;
    cin>>sum;
    int temp;
    vector<int>v1;
    vector<int>v2;
    for(int i=0;i<n;i++){
        cin>>temp;
        v1.push_back(temp);
    }

   cout<<sumi(v1,n,sum);
}
```

Q10)longest subarray with equal 0's and 1's

The question is same as the longest subarray question just that it should be changed 0 to -1

Q11)longest consecutive consequence

```
#include<bits/stdc++.h>
using namespace std;
void solve(vector<int>&v){
    unordered_set<int>s(v.begin(),v.end());
    int res=0;
    for(int i=0;i<v.size();i++){
    if(s.find(v[i]-1)==s.end()){
        int count=1;
        while(s.find(v[i]+count)!=s.end()){
            count++;
            res=max(res,count);
        }
    }
    }
    cout<<res;

}
 int main(){
     int temp1,temp2,n;
     cin>>n;
     vector<int>v;
     for(int i=0;i<n;i++){
         cin>>temp1;
         v.push_back(temp1);
     }
     solve(v);
 }
```

Q12)longest common span with spaces in binary array

Here the logic is substract the array1 with another array and find the longest span of sum 0…

```
#include<bits/stdc++.h>
using namespace std;
```

```
void solve(vector<int>&v1,vector<int>v2){
   int n=v1.size();
   int v[n];
   for(int i=0;i<n;i++){
      v[i]=v1[i]-v2[i];
   }
   int sum=0,maxlen=0;
   unordered_map<int,int>m;
   for(int i=0;i<n;i++){
      sum=sum+v[i];
      if(sum==0){
         maxlen=i+1;
      }
      if(m.find(sum)==m.end()){
         m[sum]=i;
      }
      if(m.find(sum)!=m.end()){
         maxlen=max(maxlen,i-m[sum]);
      }
   }

  cout<<maxlen;

}
 int main(){
    int temp1,temp2,n,m;
    cin>>n;
    vector<int>v1;
    vector<int>v2;
    for(int i=0;i<n;i++){
       cin>>temp1;
       v1.push_back(temp1);
    }
    for(int i=0;i<n;i++){
      cin>>temp2;
      v2.push_back(temp2);
   }
   solve(v1,v2);
}
```

Q14)more than n/k occurrence

Use the moores voting algo or use the normal hashing method … do it yourself

Ordered map and sets

Same logic just in sorted order with higher TC..

Note:the Tc depends on .size()*logn not logn directly