[L1]Class - 21 ( Policy based data structure and Practice Problem )[Bangla]

Class - 21 :

Class Link : <https://youtu.be/t9-uFN23HjQ>

**Discussed topics:**

1) Policy based data structure

2) Policy based data structure template:

Link : <https://paste.ubuntu.com/p/WvKJXBPNSj/>

3) Petya and Array (Codeforces)

Link : <https://codeforces.com/problemset/problem/1042/D>

My Code : <https://codeforces.com/contest/1042/submission/93346374>

#include**<bits/stdc++.h>  
using namespace** std;  
#define **optimize**() ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define **endl '\n'  
int** main()  
{  
 **optimize**();  
  
 set<**int**>s={1,2,3,4,5};  
 cout<<\*lower\_bound(s.begin(),s.end(),2);*//2* cout<<\*upper\_bound(s.begin(),s.end(),2);*//3* **return** 0;  
}

**Policy based data structure template:**

#include**<bits/stdc++.h>  
using namespace** std;  
  
*///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Policy based data structure Template \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///*#include **<ext/pb\_ds/assoc\_container.hpp>**#include **<ext/pb\_ds/tree\_policy.hpp>  
using namespace** \_\_gnu\_pbds;  
  
**typedef** tree< **long long**, null\_type, less<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > increasing\_set;  
**typedef** tree< **long long**, null\_type, less\_equal<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > increasing\_multiset;  
**typedef** tree< **long long**, null\_type, greater<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > decreasing\_set;  
**typedef** tree< **long long**, null\_type, greater\_equal<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > decreasing\_multiset;  
*/// change ll(long long) to any data type  
/// less\_equal for multiset increasing order  
/// less for set increasing order  
/// greater\_equal for multiset decreasing order  
/// greater for set decreasing order  
  
/// cout<<\*X.find\_by\_order(1)<<endl; // iterator to the k-th largest element  
/// cout<<X.order\_of\_key(-5)<<endl; // number of items in a set that are strictly smaller than our item  
  
///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Policy based data structure Template End \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///*#define **optimize**() ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define **endl '\n'  
int** main()  
{  
 **optimize**();  
 **return** 0;  
}

**Increasing set// Deccreasing set:**

#include**<bits/stdc++.h>  
using namespace** std;  
  
#include **<ext/pb\_ds/assoc\_container.hpp>**#include **<ext/pb\_ds/tree\_policy.hpp>  
using namespace** \_\_gnu\_pbds;  
  
**typedef** tree< **long long**, null\_type, less<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > increasing\_set;  
**typedef** tree< **long long**, null\_type, less\_equal<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > increasing\_multiset;  
**typedef** tree< **long long**, null\_type, greater<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > decreasing\_set;  
**typedef** tree< **long long**, null\_type, greater\_equal<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > decreasing\_multiset;  
  
*/// cout<<\*X.find\_by\_order(1)<<endl; // iterator to the k-th largest element  
/// cout<<X.order\_of\_key(-5)<<endl; // number of items in a set that are strictly smaller than our item*#define **optimize**() ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define **endl '\n'  
int** main()  
{  
 **optimize**();  
  
 increasing\_set ps;// decreasing\_set ps;

ps.insert(1);  
 ps.insert(2);  
 ps.insert(4);  
 ps.insert(3);  
 ps.insert(3);  
  
 cout<<ps.size()<<**endl**;  
 **for**(**auto** u:ps) cout<<u<<**" "**;  
 cout<<**endl**;  
 **return** 0;  
}

**Output:**

4

1 2 3 4

**//Output:**

4

4 3 2 1

**increasing multiset// decreasing multiset:**

#include**<bits/stdc++.h>  
using namespace** std;  
  
#include **<ext/pb\_ds/assoc\_container.hpp>**#include **<ext/pb\_ds/tree\_policy.hpp>  
using namespace** \_\_gnu\_pbds;  
  
**typedef** tree< **long long**, null\_type, less<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > increasing\_set;  
**typedef** tree< **long long**, null\_type, less\_equal<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > increasing\_multiset;  
**typedef** tree< **long long**, null\_type, greater<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > decreasing\_set;  
**typedef** tree< **long long**, null\_type, greater\_equal<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > decreasing\_multiset;  
  
*/// cout<<\*X.find\_by\_order(1)<<endl; // iterator to the k-th largest element  
/// cout<<X.order\_of\_key(-5)<<endl; // number of items in a set that are strictly smaller than our item*#define **optimize**() ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define **endl '\n'  
int** main()  
{  
 **optimize**();  
  
 increasing\_multiset ps;// decreasing\_multiset ps  
  
 ps.insert(1);  
 ps.insert(2);  
 ps.insert(4);  
 ps.insert(3);  
 ps.insert(3);  
  
 cout<<ps.size()<<**endl**;  
 **for**(**auto** u:ps) cout<<u<<**" "**;  
 cout<<**endl**;  
 **return** 0;  
}

**Output:**

4

1 2 3 3 4

**//Output:**

4

4 3 3 2 1

**Set lower\_bount**:

#include**<bits/stdc++.h>  
using namespace** std;  
  
#include **<ext/pb\_ds/assoc\_container.hpp>**#include **<ext/pb\_ds/tree\_policy.hpp>  
using namespace** \_\_gnu\_pbds;  
  
**typedef** tree< **long long**, null\_type, less<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > increasing\_set;  
**typedef** tree< **long long**, null\_type, less\_equal<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > increasing\_multiset;  
**typedef** tree< **long long**, null\_type, greater<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > decreasing\_set;  
**typedef** tree< **long long**, null\_type, greater\_equal<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > decreasing\_multiset;  
  
*/// cout<<\*X.find\_by\_order(1)<<endl; // iterator to the k-th largest element  
/// cout<<X.order\_of\_key(-5)<<endl; // number of items in a set that are strictly smaller than our item*#define **optimize**() ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define **endl '\n'  
int** main()  
{  
 **optimize**();  
  
 increasing\_set ps;  
  
 ps.insert(1);  
 ps.insert(2);  
 ps.insert(4);  
 ps.insert(3);  
 ps.insert(3);  
  
 cout<<ps.order\_of\_key(3)<<**endl**;  
 **return** 0;  
}

**Output:**

2

**partial\_sum() in C++:**

**partial\_sum(first, last, b);**

**partial\_sum(first, last, b, myfun);**

first, last : first and last element of range whose elements are to be added

b : index of array where corresponding partial sum will be stored;

myfun : a user defined function for performing any specific task

*// C++ program to demonstrate working of accumulate()*#include **<iostream>**#include **<numeric>  
using namespace** std;  
  
*//user defined function***int** myfun(**int** x, **int** y)  
{  
 *// the sum of element is twice of its   
 // adjacent element* **return** x + 2 \* y;  
}  
  
**int** main ()  
{  
 **int** a[] = {1, 2, 3, 4, 5} ;  
 **int** b[5];  
  
 *// Default function* partial\_sum(a , a+5 , b);  
  
 cout << **"Partial Sum - Using Default function: "**;  
 **for** (**int** i=0; i<5; i++)  
 cout << b[i] << **' '**;  
 cout << **'\n'**;  
  
 *// Using user defined function* partial\_sum(a , a+5 , b , myfun) ;  
  
 cout << **"Partial sum - Using user defined function: "**;  
 **for** (**int** i=0; i<5; i++)  
 cout << b[i] << **' '**;  
 cout << **'\n'**;  
  
 **return** 0;  
}

**Output:**

Partial Sum - Using Default function: 1 3 6 10 15

Partial sum - Using user defined function: 1 5 11 19 29

Petya and Array (Codeforces)

Link : <https://codeforces.com/problemset/problem/1042/D>

My Code : <https://codeforces.com/contest/1042/submission/93346374>

Time Complexity=O(nlog2n)

#include**<bits/stdc++.h>  
using namespace** std;  
  
#include **<ext/pb\_ds/assoc\_container.hpp>**#include **<ext/pb\_ds/tree\_policy.hpp>  
using namespace** \_\_gnu\_pbds;  
  
**typedef** tree< **long long**, null\_type, less<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > increasing\_set;  
**typedef** tree< **long long**, null\_type, less\_equal<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > increasing\_multiset;  
**typedef** tree< **long long**, null\_type, greater<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > decreasing\_set;  
**typedef** tree< **long long**, null\_type, greater\_equal<**long long**>, rb\_tree\_tag,tree\_order\_statistics\_node\_update > decreasing\_multiset;  
  
*/// cout<<\*X.find\_by\_order(1)<<endl; // iterator to the k-th largest element  
/// cout<<X.order\_of\_key(-5)<<endl; // number of items in a set that are strictly smaller than our item*#define **optimize**() ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
#define **endl '\n'  
  
const int** mx=2e5+123;  
**long long** a[mx],sum[mx],b[mx];  
  
**int** main()  
{  
 **optimize**();  
  
 **long long** n, t;  
 cin >> n >> t;  
 **for** ( **int** i = 1; i <= n; i++ ) cin >> a[i];  
 **for** ( **int** i = 1; i <= n; i++ ) sum[i] = sum[i-1]+a[i];  
  
 map<**long long**, **int**> cnt;  
 increasing\_multiset ps;  
  
 ps.insert ( 0 );  
 cnt[0]++;  
  
 **long long** ans = 0;  
 **for** ( **int** i = 1; i <= n; i++ ) {  
 ans += ( i - ps.order\_of\_key(sum[i]-t) - cnt[sum[i]-t] );  
 ps.insert ( sum[i] );  
 cnt[sum[i]]++;  
 }  
  
 cout << ans << **endl**;  
  
 **return** 0;  
}

**Input:**

5 4

5 -1 3 4 -1

**Output:**

5