Template of segment tree and hashing

//interface

```
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;
template <typename T> using o_set = tree<T, null_type, less<T>, rb_tree_tag,
tree_order_statistics_node_update>;
//variation= ll          less_equal<ll>      (change according to need)        ordered_multiset
// ordered_set os         declare like this
// os.insert(1)           insert like this
// *os.find_by_order(k)    returns an iterator to the k-th largest element (counting from zero)
// os.order_of_key(X)      returns the number of items in a set that are strictly smaller than X
// works like set and complexity O(logN)  and saves in ascending order with input order index
//if error occurs=
c:\mingw\lib\gcc\mingw32\6.3.0\include\c++\ext\pb_ds\detail\resize_policy\hash_standard_resiz
e_policy_imp.hpp0000644. Rename that file to remove the 0000644 from the end of it.

#define long long int ll
#define endl '\n'
int main(){
ios_base:: sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);
}
```

//hashcode

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long int
int n;
int mod1=1000000007,mod2=1000000009,p1=31,p2=29;
int biyok(int a,int b,int mod){
    return ((a-b)+mod)%mod;
}
int jog(int a,int b,int mod){
    return (a+b)%mod;
}
int gun(int a,int b,int mod){
    return (1ll*(a%mod)*(b%mod))%mod;
}
int binaryexp(int a,int b,int mod){
    int result=1;
    while(b>0){
        if(b&1)result=gun(result,a,mod);
```

```
        a=gun(a,a,mod);
        b/=2;
    }
    return result;
}
void
all(vector<pair<int,int>>&pref,vector<pair<int,int>>&suff,vector<pair<int,int>>&pw,vector<pair<in
t,int>>&inv,vector<int>&arr){
pw[0]={1,1};
for(int i=1;i<n;i++){
    pw[i].first=gun(pw[i-1].first,p1,mod1);
    pw[i].second=gun(pw[i-1].second,p2,mod2);
}
inv[0]={1,1};
int inv1=binaryexp(p1,mod1-2,mod1);
int inv2=binaryexp(p2,mod2-2,mod2);
for(int i=1;i<n;i++){
    inv[i].first=gun(inv1,inv[i-1].first,mod1);
    inv[i].second=gun(inv2,inv[i-1].second,mod2);
}
pref[0]={arr[0],arr[0]};
suff[n-1]={arr[n-1],arr[n-1]};
for(int i=1;i<n;i++){
    pref[i].first=jog(gun(pw[i].first,arr[i],mod1),pref[i-1].first,mod1);
    pref[i].second=jog(gun(pw[i].second,arr[i],mod2),pref[i-1].second,mod2);
}
for(int i=n-2,j=1;i>=0;i--,j++){
    suff[i].first=jog(gun(pw[j].first,arr[i],mod1),suff[i+1].first,mod1);
    suff[i].second=jog(gun(pw[j].second,arr[i],mod2),suff[i+1].second,mod2);
}

}
int main()
{
    ios_base:: sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);

    cin>>n;
    vector<pair<int,int>>pref(n+1);
    vector<pair<int,int>>suff(n+1);
    vector<pair<int,int>>pw(n+1);
    vector<pair<int,int>>inv(n+1);
    vector<int>arr(n+1);
    for(int i=0;i<n;i++)cin>>arr[i];
    all(pref,suff,pw,inv,arr);


}
```

//Segment tree

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long int
#define endl '\n'
const int N=100001;
int n;
struct node{
   ll hsuff,hpref,subsum;
   ll totsum;
};
 vector<node>tree(4*N);
ll arr[N+1];
node merge(node l,node r){
   if(l.subsum==INT_MAX)return r;
   if(r.subsum==INT_MAX)return l;
   node ans;
   ans.subsum=max(l.subsum,r.subsum);
   ans.subsum=max(ans.subsum,l.hsuff+r.hpref);
   ans.totsum=l.totsum+r.totsum;
   ans.hsuff=max(r.hsuff,r.totsum+l.hsuff);
   ans.hpref=max(l.hpref,l.totsum+r.hpref);
   return ans;
}
void built(int nodee,int l,int r){
   if(l==r){
      tree[nodee].hsuff=tree[nodee].hpref=tree[nodee].subsum=tree[nodee].totsum=arr[l];
   }else{
      int mid=(l+r)/2;
      int lft=2*nodee,rgt=2*nodee+1;
      built(lft,l,mid);
      built(rgt,mid+1,r);
      tree[nodee]=merge(tree[lft],tree[rgt]);
   }
}
void update(int nodee,int l,int r,int i,int j,int value){
  if(r<i || l>j)return;
  if(l>=i && r<=j){
    tree[nodee].hsuff=tree[nodee].hpref=tree[nodee].subsum=tree[nodee].totsum=value;
  }else{
     int mid=(l+r)/2;
      int lft=2*nodee,rgt=2*nodee+1;
      update(lft,l,mid,i,j,value);
      update(rgt,mid+1,r,i,j,value);
      tree[nodee]=merge(tree[lft],tree[rgt]);
  }
}
node search(int nodee,int l,int r,int i,int j){
   if(r<i || l>j){
      // cout<<"haha"<<endl;
```

```
      return {INT_MAX,INT_MAX,INT_MAX,INT_MAX};
    }
    if(l>=i && r<=j){

      return tree[nodee];
    }
    else{

      int mid=(l+r)/2;
      int lft=2*nodee,rgt=2*nodee+1;
      return merge(search(lft,l,mid,i,j),search(rgt,mid+1,r,i,j));
    }
}
int main()
{
    ios_base:: sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);


cin>>n;
for(int i=0;i<n;i++){
    cin>>arr[i];
}
built(1,0,n-1);
int q;
cin>>q;

while(q--){
    int type,a,b;
    cin>>type>>a>>b;
    if(type==1){
      node ans=search(1,0,n-1,a-1,b-1);
      cout<<ans.subsum<<endl;

    }else{
      update(1,0,n-1,a-1,a-1,b);
    }

}



}
```

//segment tree with binary_search

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long int
#define endl '\n'
```

```
const int N=100001;
int arr[N+1];
vector<int>dp[4*N+1];
vector<ll>dpr[4*N+1];
void build(int node,int l,int r){
    if(l==r){
        dp[node].push_back(arr[l]);
        dpr[node].push_back(arr[l]);
        // cout<<arr[l]<<endl;
        // cout<<l<<" "<<r<<endl;
        // cout<<endl;
    }else{
        int mid=(l+r)/2;
       int left=2*node,right=2*node+1;
       build(left,l,mid);
       build(right,mid+1,r);
       int i=0,j=0;
       while(i<dp[left].size() && j<dp[right].size()){
        if(dp[left][i]<dp[right][j]){
            dp[node].push_back(dp[left][i]);
            ++i;
        }else{
            dp[node].push_back(dp[right][j]);
            ++j;
        }
       }
        while(j<dp[right].size()){
            dp[node].push_back(dp[right][j]);
            ++j;
        }

        while(i<dp[left].size()){
            dp[node].push_back(dp[left][i]);
            ++i;
       }
       dpr[node].push_back(dp[node][0]);
       for(int i=1;i<dp[node].size();i++){
         ll c=dp[node][i]+dpr[node][i-1];
         dpr[node].push_back(c);
       }

    }
}
ll ssearch(int node,int l,int r,int i,int j,ll value){
    if(l>j || r<i)return 0;
    if(l>=i && r<=j){
        ll ans=-1;
        int low=0,high=dp[node].size()-1;
        while(high>=low){
            int mid=(high+low)/2;
            if(dp[node][mid]<value){
```

```
            ans=mid;
            low=mid+1;
         }else high=mid-1;
      }
      if(ans==-1)return 0;
      else{
        return (value*(ans+1))-dpr[node][ans];
      }
   }else{
      int mid=(r+l)/2;
      return 1ll*(ssearch(2*node,l,mid,i,j,value)+ssearch(2*node+1,mid+1,r,i,j,value));
   }
}
int main()
{
  ios_base:: sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);

int n,q;
cin>>n>>q;
for(int i=0;i<n;i++)cin>>arr[i];
build(1,0,n-1);
while(q--){
   ll value;
   int a,b;
   cin>>a>>b>>value;
   cout<<ssearch(1,0,n-1,a-1,b-1,value)<<endl;
}



}
```