segment_tree

Wednesday, 4 November 2015

**KQUERY - K-query(offline + segment tree)

KQUERY - K-query

Given a sequence of n numbers a_1, a_2, ..., a_n and a number of k- queries. A k-query is a triple (i, j, k) (1 ≤ i ≤ j ≤ n). For each k-query (i, j, k), you have to return the number of elements greater than k in the subsequence a_i, a_{i+1}, ..., a_j.

Input

- Line 1: n (1 ≤ n ≤ 30000).
- Line 2: n numbers a_1, a_2, ..., a_n (1 ≤ a_i ≤ 10^9).
- Line 3: q (1 ≤ q ≤ 200000), the number of k- queries.
- In the next q lines, each line contains 3 numbers i, j, k representing a k-query (1 ≤ i ≤ j ≤ n, 1 ≤ k ≤ 10^9).

Output

- For each k-query (i, j, k), print the number of elements greater than k in the subsequence a_i, a_{i+1}, ..., a_j in a single line.

Example

```
Input
5
5 1 2 3 4
3
2 4 1
4 4 4
1 5 2

Output
2
0
3
```

```
-------------------------------------------editorial------------------------------
```

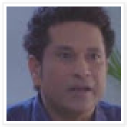
Imagine we have an array $b_1, b_2, ..., b_n$ which, $b_i \in 0, 1$ and $b_i = 1$ if an only if $a_i > k$, then we can easily answer the query (i, j, k) in$O(log(n))$ using a simple segment tree (answer is $b_i + b_{i+1} + ... + b_j$).

We can do this ! We can answer the queries offline.

First of all, read all the queries and save them somewhere, then sort them in increasing order of k and also the array a in increasing order (compute the permutation $p_1, p_2, ..., p_n$ where $a_{p_1} \le a_{p_2} \le ... \le a_{p_n}$)

At first we'll set all array b to 1 and we will set all of them to 0 one by one.

About Me



deepak gautam

Follow 9

View my complete profile

Blog Archive

- ► 2016 (25)
- ▼ 2015 (25)
  - ► December (1)
  - ▼ November (3)
    - KQUERYO - K-Query Online(vector in segtree , mereg...
    - **KQUERY - K-query(offline + segment tree)
    - **Sereja and Brackets(div2 E)
  - ► October (6)
  - ► September (3)
  - ► August (1)
  - ► July (11)

Consider after sorting the queries in increasing order of their k, we have a permutation $w_1, w_2, ..., w_q$ (of $1, 2, ..., q$) where$k_{w_1} \le k_{w_2} \le k_{w_2} \le ... \le k_{w_q}$ (we keep t he answer to the i - th query in ans_i .

Pseudo code : (all 0-based)

```
po = 0
for j = 0 to q-1
 while po < n and a[p[po]] <= k[w[j]]
  b[p[po]] = 0, po = po + 1
```

So, build function would be like this ($s[x]$ is the sum of b in the interval of node x) :

```
void build(int id = 1,int l = 0,int r = n){
 if(r - l < 2){
  s[id] = 1;
  return ;
 }
 int mid = (l+r)/2;
 build(2 * id, l, mid);
 build(2 * id + 1, mid, r);
 s[id] = s[2 * id] + s[2 * id + 1];
}
```

et An update function for when we want to st `b[p[po]] = 0` to update the segment tree:

```
void update(int p,int id = 1,int l = 0,int r = n){
 if(r - l < 2){
  s[id] = 0;
  return ;
 }
 int mid = (l+r)/2;
 if(p < mid)
  update(p, 2 * id, l, mid);
 else
  update(p, 2 * id + 1, mid, r);
 s[id] = s[2 * id] + s[2 * id + 1];
}
```

Finally, a function for sum of an interval

```
int sum(int x,int y,int id = 1,int l = 0,int r = n){// [x, y)
 if(x >= r or l >= y) return 0;// [x, y) intersection [l,r) = empty
 if(x <= l && r <= y) // [l,r) is a subset of [x,y)
  return s[id];
 int mid = (l + r)/2;
 return sum(x, y, id * 2, l, mid) +
        sum(x, y, id*2+1, mid, r) ;
}
```

So, in main function instead of that pseudo code, we will use this :

```
build();
int po = 0;
for(int y = 0;y < q;++ y){
 int x = w[y];
 while(po < n && a[p[po]] <= k[x])
  update(p[po ++]);
 ans[x] = sum(i[x], j[x] + 1); // the interval [i[x], j[x] + 1)
}


-------------------------------------------code--------------------
------------------



#include<iostream>
#include<string.h>
#include<bits/stdc++.h>
 int nn[30005];
int arr[30005],seg[100000];
using namespace std;

vector<pair<int,int> > v;


int ans[200005];
struct node
 {
  int k,l,r,qn;
 } vv[200005];


 int read_int(){
 char r;
 bool start=false,neg=false;
 int ret=0;
 while(true){
  r=getchar();
  if((r-'0'<0 || r-'0'>9) && r!='-' && !start){
   continue;
  }
  if((r-'0'<0 || r-'0'>9) && r!='-' && start){
   break;
  }
  if(start)ret*=10;
  start=true;
  if(r=='-')neg=true;
  else ret+=r-'0';
 }
 if(!neg)
  return ret;
 else
  return -ret;
}

#define inf 100000001



bool compare(node n1,node n2)
 {
  if(n1.k>n2.k) return false;
  else return true;
 }



 int ups,upe,qs,qe;//qs = query start index , qe= query end index

 int val;// ups = update start index , upe =update end index


int qry(int index,int start,int end)
  {



        if(start>end || end<qs || start>qe)
       {
        return 0;
       }
        if(start>=qs && end<=qe)
         {
          return  seg[index];

         }
        int q1=qry(2*index,start,(start+end)/2);
          int q2=qry(2*index+1,((start+end)/2)+1,end);
         return q1+q2;

  }

  void build(int index,int start,int end)
   {




   if(start==end)
    {
     seg[index]=1;
     return;
 }



     build(2*index,start,(start+end)/2);
     build(2*index+1,((start+end)/2)+1,end);

     seg[index]=seg[2*index]+seg[2*index+1];
   //  cout<<" index        "<<index<<" val "<<seg[index]<<endl;
   }




void update(int index,int start,int end)
{
// cout<<"update "<<start<<" "<<end<<endl;
   if(start>end || start>upe || end<ups) return ;// if(range in com
plitly out of range sooo need not to update ;;;;)
   if(start==end && start==ups)
    {
    // cout<<" reach "<<index<<endl;
     seg[index]=0;
     return ;
 }
 //else if(start==end) return ;
```

```
     update(2*index,start,(start+end)/2);
     update(2*index+1,((start+end)/2)+1,end);

     seg[index]=seg[2*index]+seg[2*index+1];

 }



int main()
 {



        int n,q;
       n=read_int();
        for(int i=0;i<n;i++)
  {
   int a;
   a=read_int();
    v.push_back(make_pair(a,i));
     arr[i]=1;
  }

   //cout<<"build call "<<endl;
   build(1,0,n-1);
  // cout<<"build return "<<endl;

   sort(v.begin(),v.end());


     for(int i=0;i<n;i++) nn[i]=(v[i].first);
  //   cout<<" copy done "<<endl;
   q=read_int();

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

      int l,r,k;
      // cin>>l>>r>>k;
      l=read_int();
      r=read_int();
      k=read_int();
       vv[i].l=l;
       vv[i].r=r;
       vv[i].k=k;
       vv[i].qn=i;
     }
   //  cout<<" qinp done "<<endl;
     sort(vv,vv+q,compare);

    // cout<<" after sorting status of the query "<<endl;

     for(int i=0;i<q;i++)
      {
       int l,r,k,qn;
       k=vv[i].k;
       l=vv[i].l;
       r=vv[i].r;
       qn=vv[i].qn;
    //  cout<<" l "<<vv[i].l<<" r "<<vv[i].r<<" k "<<vv[i].k<<" "<<
vv[i].qn<<endl;
      // vector<int > :: iterator it;


      int *it=lower_bound(nn,nn+n,k+1);
      if(*it>k) it--;
       int pos=it-nn;
       if(pos==n)pos--;
     //cout<<"pos in sorted array is for "<<k<<" is "<<pos<<endl;
       for(int j=pos;j>=0;j--)
        {
      //      cout<<" updating "<<j<<endl;
         if(nn[j]==0) break;
         else
         {
          int place=v[j].second;
         // v[j].first=0;
              nn[j]=0;
          arr[place]=0;
          ups=place;
          upe=place;
              //  cout<<"index of update "<<place<<endl;
          update(1,0,n-1);

      }
     }
       qs=l-1;
       qe=r-1;
       ans[qn]=qry(1,0,n-1);
   }
   for(int i=0;i<q;i++) printf("%d\n",ans[i]);



  return 0;
 }



------------------------------------ direct online code is in the n
ext post kqueryo----
```

Posted by deepak gautam at 10:37

No comments:

Post a Comment

Enter your comment...

Comment as: Google Accoun ▾

Publish Preview

Newer Post Home Older Post