L21 Range Query 2

1-Tut : Coder's Rating

Send Feedback

Some of the more elite (and not-so-elite) coders around take part in a certain unnamed programming contest. In said contest, there are multiple types of competitions. Here, we consider the Open and High School competition types. For each type, each competitor receives a rating, an integer between 1 and 100000, inclusive. A coder's rating is based upon his or her level of performance in matches and is calculated using a complicated formula which, thankfully, you will not be asked to implement.

Although the Open and High School ratings for a coder who has participated in both competition types lately are usually close, this is not always the case. In particular, High School matches are more about speed, since many coders are able to solve all the problems, whereas Open matches require more thinking and there is a steeper curve in terms of problem difficulty.

Problem Statement

You are given N coders (1 ≤ N ≤ 300000), conveniently numbered from 1 to N. Each of these coders participates in both High School and Open matches. For each coder, you are also given an Open rating Ai and a High School rating Hi. Coder i is said to be better than coder j if and only if both of coder i's ratings are greater than or equal to coder j's corresponding ratings, with at least one being greater. For each coder i, determine how many coders coder i is better than.

Input Format

On the first line of input is a single integer N, as described above. N lines then follow. Line i+1 contains two space-separated integers, Ai and Hi.

Output Format

Line i should contain the number of coders that coder i is better than.

Sample Input 1:

```
8
1798 1832
862 700
1075 1089
1568 1557
2575 1984
1033 950
1656 1649
1014 1473
```

Sample Output 1:

```
6
0
2
```

4
7
1
5
1
Explanation

1st code is better than 2nd, 3rd, 4th, 5th, 6th and 7th coder.

Hence he is better than 6 coders.

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

class coder
{
public:
    int x, y, index;
};
bool sorter(coder c1, coder c2)
{
    if (c1.x == c2.x)
    {
        return c1.y < c2.y;
    }
    return c1.x < c2.x;
}
void update(int y, int *bit)
{
    for (; y <= 100000; y += y & (-y))
    {
        bit[y]++;
    }
}
int query(int y, int *bit)
{
    int count = 0;
    for (; y > 0; y -= y & (-y))
    {
        count += bit[y];
    }
    return count;
}
int main()
{
```

```
    int n;
    cin >> n;
    coder *arr = new coder[n];
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i].x >> arr[i].y;
        arr[i].index = i;
    }
    sort(arr, arr + n, sorter);
    int *bit = new int[100001];
    int *ans = new int[n];
    for (int i = 0; i < n;)
    {
        int endindex = i;
        while (endindex < n && arr[i].x == arr[endindex].x && arr[i].y == arr[endindex].y)
        {
            endindex++;
        }
        for (int j = i; j < endindex; j++)
        {
            ans[arr[j].index] = query(arr[j].y, bit);
        }
        for (int j = i; j < endindex; j++)
        {
            update(arr[j].y, bit);
        }
        i = endindex;
    }
    for (int i = 0; i < n; i++)
    {
        cout << ans[i] << endl;
    }
    return 0;
}
```

2-Tut : Distinct Query Problem

Send Feedback

Given a sequence of n numbers a1, a2, ..., an and a number of d-queries. A d-query is a pair (i, j) (1 ≤ i ≤ j ≤ n). For each d-query (i, j), you have to return the number of distinct elements in the subsequence ai, ai+1, ..., aj.

Input Format:

Line 1: n (1 ≤ n ≤ 30000).
Line 2: n numbers a1, a2, ..., an (1 ≤ ai ≤ 10^6).

Line 3: q (1 ≤ q ≤ 200000), the number of d-queries.
In the next q lines, each line contains 2 numbers i, j representing a d-query (1 ≤ i ≤ j ≤ n).

Output Format:

For each d-query (i, j), print the number of distinct elements in the subsequence ai, ai+1, ..., aj in a single line.

Sample Input 1:

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

Sample Output 1:

```
3
2

3
```

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

class event
{
public:
    int first, second, index;
};

bool compare(event a, event b)
{

    return a.second < b.second;
}

void update(int index, int value, int n, int *bit)
{
    for (; index <= n; index += (index & (-index)))
    {
        bit[index] += value;
    }
}

int value(int index, int *bit)
{
    int count = 0;
    for (; index > 0; index -= (index & (-index)))
    {
```

```
        count += bit[index];
    }
    return count;
}

int main()
{

    int n;
    cin >> n;

    int *arr = new int[n + 1];
    for (int i = 1; i <= n; i++)
    {
        cin >> arr[i];
    }

    int q;
    cin >> q;

    event *query = new event[q];
    for (int i = 0; i < q; i++)
    {
        cin >> query[i].first >> query[i].second;
        query[i].index = i;
    }

    sort(query, query + q, compare);

    int *bit = new int[n + 1];
    int *ans = new int[q];

    int total = 0;
    int k = 0;
    int *last = new int[1000001];
    for (int i = 0; i <= 1000000; i++)
    {
        last[i] = -1;
    }

    for (int i = 1; i <= n; i++)
    {
        if (last[arr[i]] != -1)
        {
```

```
            update(last[arr[i]], -1, n, bit);
        }
        else
        {
            total++;
        }

        update(i, 1, n, bit);
        last[arr[i]] = i;

        while (k < q && query[k].second == i)
        {
            ans[query[k].index] = total - value(query[k].first - 1, bit);
            k++;
        }
    }

    for (int i = 0; i < q; i++)
    {
        cout << ans[i] << endl;
    }
    return 0;
}
```

3-Tut : OrderSet - Problem

Send Feedback

In this problem, you have to maintain a dynamic set of numbers which support the two fundamental operations

INSERT(S,x): if x is not in S, insert x into S
DELETE(S,x): if x is in S, delete x from S
and the two type of queries

K-TH(S) : return the k-th smallest element of S
COUNT(S,x): return the number of elements of S smaller than x

Input Format:

Line 1: Q , the number of operations
In the next Q lines, the first token of each line is a character I, D, K or C meaning that the corresponding operation is INSERT, DELETE, K-TH or COUNT, respectively, following by a whitespace and an integer which is the parameter for that operation.

Constraints:

1 <= Q <= 2*10^5
-10^9 <= x <= 10^9

Output Format:

For each query, print the corresponding result in a single line. In particular, for the queries K-TH, if k is larger than the number of elements in S, print the word 'invalid'.

Sample Input 1:

```
8
I -1
I -1
I 2
C 0
K 2
D -1
K 1
K 2
```

Sample Output 1:

```
1
2
2

invalid
```

```
#include <bits/stdc++.h>

using namespace std;

#define MAX 200005
#define ll long long int

pair<int, int> M[MAX], T[MAX];
int BIT[MAX], A[MAX], n;
char C[MAX];
bool b;

int get_count(int idx)
{
    if (idx == 0)
        return 0;
    ll sum = 0;
    while (idx > 0)
    {
        sum += BIT[idx];
        idx -= idx & (-idx);
    }
    return sum;
}
void updateBIT(int idx, int val)
{
```

```
27.     while (idx < MAX)
28.     {
29.
30.         BIT[idx] += val;
31.         idx += idx & (-idx);
32.     }
33. }
34. int b_search(int x)
35. {
36.     int lo = 0, hi = n - 1, mid, ans = -1;
37.     while (lo <= hi)
38.     {
39.         mid = (lo + hi) >> 1;
40.
41.         if (M[mid].first == x)
42.         {
43.             b = 1;
44.             return mid;
45.         }
46.         else if (M[mid].first > x)
47.             hi = mid - 1;
48.         else
49.         {
50.             ans = mid;
51.             lo = mid + 1;
52.         }
53.     }
54.     b = 0;
55.     return ans;
56. }
57. int main()
58. {
59.     int i, Q, x, k = 0, lo, hi, mid;
60.     cin >> Q;
61.     for (i = 0; i < Q; i++)
62.     {
63.         C[i] = getchar();
64.         while (C[i] < 65 || C[i] >= 91)
65.             C[i] = getchar();
66.         cin >> A[i];
67.         if (C[i] == 'I')
68.         {
69.             T[k].first = A[i];
70.             T[k++].second = 0;
```

```
71.         }
72.     }
73.     sort(T, T + k);
74.     M[0] = T[0];
75.     n = 1;
76.     for (i = 1; i < k; i++)
77.     {
78.         if (T[i].first != T[i - 1].first)
79.         {
80.             M[n++] = T[i];
81.         }
82.     }
83.     for (i = 0; i < Q; i++)
84.     {
85.         if (C[i] == 'I')
86.         {
87.             x = b_search(A[i]);
88.
89.             if (M[x].second == 0)
90.             {
91.                 M[x].second = 1;
92.                 updateBIT(x + 1, 1);
93.             }
94.         }
95.         else if (C[i] == 'D')
96.         {
97.             x = b_search(A[i]);
98.             if (x != -1 && M[x].second == 1 && b)
99.             {
100.                 updateBIT(x + 1, -1);
101.                 M[x].second = 0;
102.             }
103.          }
104.          else if (C[i] == 'C')
105.          {
106.              x = b_search(A[i]);
107.
108.              if (b)
109.                  cout << get_count(x + 1 - 1) << endl;
110.              else if (x != -1)
111.                  cout << get_count(x + 1) << endl;
112.              else
113.                  cout << 0 << endl;
114.          }
```

```
        else if (C[i] == 'K')
        {
            k = A[i];
            x = -1;
            bool mno = 0;
            lo = 1;
            hi = MAX - 1;
            while (lo <= hi)
            {
                mid = (lo + hi) >> 1;
                if (get_count(mid) == k && get_count(mid - 1) != k)
                {
                    mno = 1;
                    x = mid;
                    break;
                }
                else if (get_count(mid) < k)
                {
                    x = mid;
                    lo = mid + 1;
                }
                else
                    hi = mid - 1;
            }

            if (!mno)
                cout << "invalid" << endl;
            else
                cout << M[x - 1].first << endl;
        }
    }
    return 0;
}
```

4-Ass : KQUERY

Send Feedback

Given a sequence of n numbers a1, a2, ..., an 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 ai, ai+1, ..., aj.

Input Format

Line 1: Contains an integer N denoting the number of elements in the array
Line 2: N space-separated integers denoting the elements of the array.
Line 3: Number of queries Q

Next Q line contain two space-separated integers i, j, k descrbing the current query

Constraints:

1 <= N <= 10^5
1 <= Q <= 10^5
1 <= arr[i] <= 10^9
1 <= i <= j <= N
1 <= k <= 10^9

Output Format:

For each k-query (i, j, k), print the number of elements greater than k in the subsequence ai, ai+1, ..., aj in a single line.

Sample Input 1:

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

Sample Output 1:

```
2
0

3
```

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

vector<pair<int, int>> a;

vector<pair<pair<int, int>, pair<int, int>>> queries;

int ans[100002];
int bit[100002];
int query(int r, int n)
{
    int ans = 0;
    while (r > 0)
    {
        ans += bit[r];
        r -= (r & -r);
    }
    return ans;
}
void update(int i, int n)
{
    while (i <= n)
        bit[i]++, i += (i & -i);
```

```
24. }
25. int main()
26. {
27.     int n;
28.     cin >> n;
29.     int i;
30.     for (i = 1; i <= n; i++)
31.     {
32.         int x;
33.         cin >> x;
34.         a.push_back({x, i});
35.     }
36.     sort(a.begin(), a.end());
37.     int q;
38.     cin >> q;
39.     for (i = 1; i <= q; i++)
40.     {
41.         int l, r, k;
42.         cin >> l >> r >> k;
43.         queries.push_back({{k, i}, {l, r}});
44.     }
45.     sort(queries.begin(), queries.end(), greater<pair<pair<int, int>, pair<int, int>>>());
46.     for (pair<pair<int, int>, pair<int, int>> p : queries)
47.     {
48.         int k = p.first.first;
49.         i = p.first.second;
50.         int l = p.second.first;
51.         int r = p.second.second;
52.         int j = a.size() - 1;
53.         while (j >= 0)
54.         {
55.             if (a[j].first > k)
56.             {
57.                 update(a[j].second, n);
58.                 a.pop_back();
59.             }
60.             else
61.                 break;
62.             j--;
63.         }
64.         int temp = query(r, n) - query(l - 1, n);
65.         ans[i] = temp;
66.     }
67.     for (i = 1; i <= queries.size(); i++)
```

```
68.         cout << ans[i] << "\n";
69.     return 0;
70. }
71.
```

5-Ass : Shil and Wave Sequence

Send Feedback

Given a sequence A1 , A2 , A3 .. AN of length N. Find total number of wave subsequences having length greater than 1.

Wave subsequence of sequence A1 , A2 , A3 .. AN is defined as a set of integers i1 , i2 .. ik such that Ai1 < Ai2 > Ai3 < Ai4 .... or Ai1 > Ai2 < Ai3 > Ai4 .... and i1 < i2 < ...< ik.Two subsequences i1 , i2 .. ik and j1 , j2 .. jm are considered different if k != m or there exists some index l such that il ! = jl

Input Format:

First line of input consists of integer N denoting total length of sequence.Next line consists of N integers A1 , A2 , A3 .. AN .

Constraints:

1 ≤ N ≤ 10^5
1 ≤ Ai ≤ 10^5

Output Format:

Output total number of wave subsequences of given sequence. Since answer can be large, output its modulo with 10^9+7

Sample Input 1:

5
1 3 5 4 2

Sample Output 1:

17

Explanation:

All the possible sequences are: [1 3] , [1 5] , [1 4] , [1 2] , [1 3 2] , [1 4 2] , [1 5 2] , [1 5 4] , [3 5] , [3 4] , [3 2] , [3 5 2] , [3 4 2] , [3 5 4] , [5 4] , [5 2] , [4 2] . Note that value in the bracket are the values from the original sequence whose positions are maintained.

```
1.  #include <bits/stdc++.h>
2.  #define MOD 1000000007
3.  using namespace std;
4.
5.  int main()
6.  {
7.      long long n, x, t, sum1, sum2;
8.      cin >> n;
9.      long long dpl[100001] = {0}, dph[100001] = {0}, a[100001] = {0};
10.     for (long long i = 1; i < n + 1; i++)
11.     {
12.         cin >> x;
```

```
        t = 100000;
        sum1 = 0, sum2 = 0;
        while (t)
        {
            sum1 = (sum1 + dph[t] + a[t]) % MOD;
            t -= (t & (-t));
        }
        t = x;
        while (t)
        {
            sum1 = (sum1 - dph[t] - a[t] + MOD) % MOD;
            t -= (t & (-t));
        }
        t = x - 1;
        while (t)
        {
            sum2 = (sum2 + dpl[t] + a[t]) % MOD;
            t -= (t & (-t));
        }
        t = x;
        while (t < 100001)
        {
            dpl[t] = (dpl[t] + sum1) % MOD;
            dph[t] = (sum2 + dph[t]) % MOD;
            a[t] += 1;
            t += (t & (-t));
        }
    }
    long long ans = 0;
    sum1 = 0;
    sum2 = 0;
    t = 100000;
    while (t)
    {
        ans = (ans + dph[t]) % MOD;
        ans = (ans + dpl[t]) % MOD;
        t -= (t & (-t));
    }
    cout << ans;
}
```

6-Ass : INCSEQ

Send Feedback

Given a sequence of N integers S1, ..., SN, compute the number of increasing subsequences of S with length K and that is, the number of K-tuples i1, ..., iK such that 1 ≤ i1 < ... < iK ≤ N and Si1 < ... < SiK.

Input Format:

The first line contains the two integers N and K.
Next line contains N space-separated integers denoting the elements of the array.

Constraints:

1 <= N <= 10^4
1 <= K <= 50
1 <= arr[i] <= 10^5

Output Format:

Print a single integer representing the number of increasing subsequences of S of length K, modulo 5,000,000.

Sample Input 1:

4 3
1 2 2 10

Sample Output 1:

2

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

#define int long long
int M = 5000000;

int n, k;
int tree[52][300001];

void update(int ind, int val, int i)
{
    while (ind <= n)
    {
        tree[i][ind] = (tree[i][ind] + val) % M;
        ind += ind & (-ind);
    }
}

int query(int ind, int i)
{
    int ans = 0;
    while (ind > 0)
    {
        ans = (tree[i][ind] + ans) % M;
        ind -= ind & (-ind);
```

```
    }
    return ans;
}

void compress(vector<int> &a)
{
    vector<int> b = a;
    sort(b.begin(), b.end());
    unordered_map<int, int> m;
    for (int i = 1, c = 1; i <= n; i++)
    {
        if (m.find(b[i]) == m.end())
            m[b[i]] = c++;
    }
    for (int i = 1; i <= n; i++)
    {
        a[i] = m[a[i]];
    }
}

signed main()
{
    cin >> n >> k;
    if (k == 1)
    {
        cout << n;
        return 0;
    }

    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++)
    {
        cin >> a[i];
    }
    compress(a);

    int ans = 0;
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= k; j++)
        {
            int p = (j == 1 ? 1 : query(a[i] - 1, j - 1));
            update(a[i], p, j);
            if (j == k)
```

```
70.                 ans = (ans + p) % M;
71.         }
72.     }
73.     cout << ans;
74.
75.     return 0;
76. }
```