Q

About

Go Pro

How to Con

Limited-time discount for early adopters | Pro Membership is only \$3 when paid yearly. 70% OFF



Commonlounge has courses with up-to-date, bite-sized lessons that deliver the most value for the time you invest. **Subscribe** below to get started, track your progress, and get the next article in your inbox every week.

Dismiss

Tutorial • Competitiv... • Last updated 9 months ago

•••

■ Part of course: Competitive Programming: From Beginner to Expert

Merge Sort Trees [Tutorial] [Edit]

Prerequisites: Segment Trees and Merge-Sort.

Motivation: Given an array of N integers. You have to answer some queries in form (l, r, k). To answer this query you have to print the number of integers less than k in the sub-array array $[l \dots r]$.

You should be able to solve the problem in O(log^2 n) per query at least. How? You can use Merge Sort Trees.

Now what is a Merge Sort Tree? Merge Sort Tree is actually a Segment Tree but each node contains a vector. If the range of the node is [l,r] then the vector will contain the elements of array[l...r] but in sorted order.

To solve the above problem we can make a Merge Sort Tree first and go into relevant segments and binary search on the vector that was stored in those nodes to count how many numbers are less than k. This will give O(log^2 n)

Building the Merge Sort Tree

Now, how do we implement Merge Sort Tree? First try yourself:)

One way is- you go into each node of the segment tree, push the elements that was supposed

Have a question? Ask here...

Q About Go Pro How to Con

Limited-time discount for early adopters | Pro Membership is only \$3 when paid yearly. 70% OFF

For a parent node you can just merge the left and right child's vector. The merging step can be done in O(n) using two pointer. Same as the merge step of merge sort.

So only in the leaf nodes push the array element and the then recursively build the whole merge sort tree. :)

This will give you O(n log n) complexity to build the merge sort tree :)

Query in Merge Sort Tree

Now the query part. It is also easy. You just need to go to each relevant segment and binary search on them to find how many elements are less than k in this range! Then add them recursively:) DONE!!! It will have complexity O(log^2 n)!

Implementation Details:

Build Function: Build Function will look something like this -

```
1 | const int maxn = 1e5+10;
 2 vector<int> tree[4*maxn];
 3 int n, m, arr[maxn];
 4 #define all(v) v.begin(), v.end()
 5 void build(int node, int 1, int r) {
 6
       if(l == r) {
 7
          tree[node].push_back(arr[l]);
 8
          return;
 9
       }
10
       int mid = l + r \gg 1,
       left = node << 1, right = left|1;</pre>
11
12
13
       build(left, l, mid);
14
       build(right, mid+1, r);
15
16
       merge(all(tree[left]), all(tree[right]),
17
               back_inserter(tree[node]));
18 | }
```

When l=r then it is a leaf node, so we just push it into that node's vector. For other nodes, we recursively build it's left and right child. Then we merge them. The last line merges left and right child's vector in O(n) time. You can do this with two pointer too.

Have a question? Ask here...

Q About Go Pro How to Con

Limited-time discount for early adopters | Pro Membership is only \$3 when paid yearly. **70% OFF Query:** Query part is quite easy, nere is a simple implementation-

```
1 | int query(int node, int l, int r, int i, int j, int k) {
2
       if(i > r \mid l \mid l > j) return 0;
3
       if(i <= 1 && r <= j) {
4
           return lower_bound(all(tree[node]), k)
5
                        - tree[node].begin();
6
7
       int mid = l + r \gg 1,
          left = node << 1, right = left|1;</pre>
9
       return query(left, l, mid, i, j, k) +
10
              query(right, mid+1, r, i, j, k);
11 | }
```

I am going to a relevant node and searching in that nodes range, how many numbers are less than k.

Here is a practice problem -

Problem MKTHNUM □

I think after these explanations it will be very easy to solve :) First try for at least 30 minutes. It you stuck then Hints are in replies :)

If anyone have some more problems that can be solved using this technique please leave a reply:)

Update in Merge Sort Trees!

We can do point updates in Merge Sort Trees! We will need STL:policy based data structure. We can go to each relevant node, erase element of that index and insert another!

We will still need lower_bound() or upper_bound(), so we can't use set here! That's why we need policy based DS.

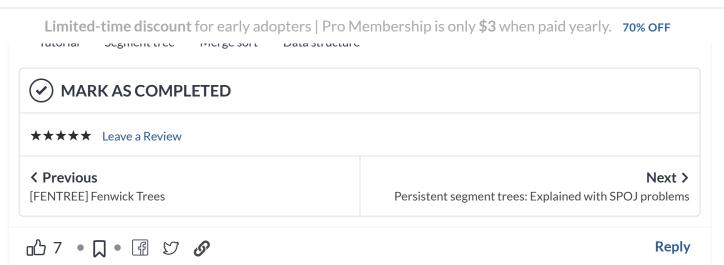
Then build complexity will be $O(n \log^2 2 n)$ and updating will be $O(\log^2 2 n)$. As there can be $O(\log n)$ relevant nodes and updating each of then will need $O(\log n)$.

If you don't know policy based ds you can read here - C++ STL: Policy based data structures □

Vous can cimply use this is the elements of the arrow are distinct! Dut when the are not

Have a question? Ask here...

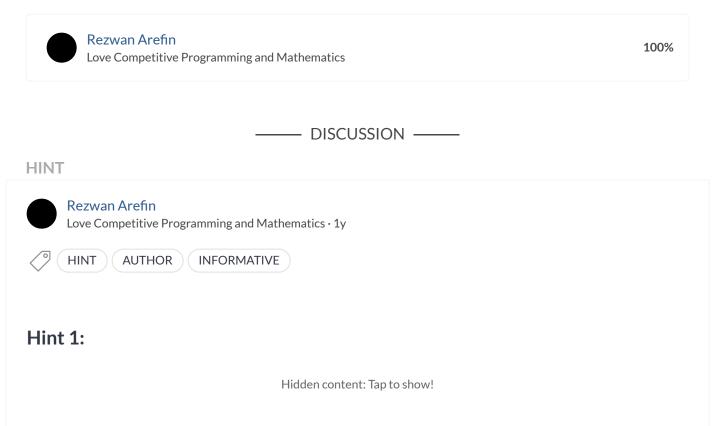




PART OF COURSES:

- Competitive Programming: From Beginner to Expert
- International Olympiad in Informatics Training: Path to Gold
- Advanced Algorithms and Data Structures

ABOUT THE CONTRIBUTOR:



Hidden content: Tan to show!

https://www.commonlounge.com/discussion/d871499b49e443259cfbea9b16f9b958

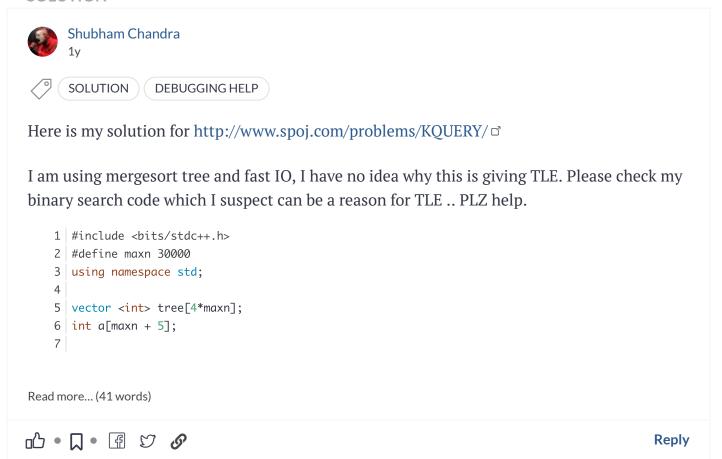
Have a question? Ask here...

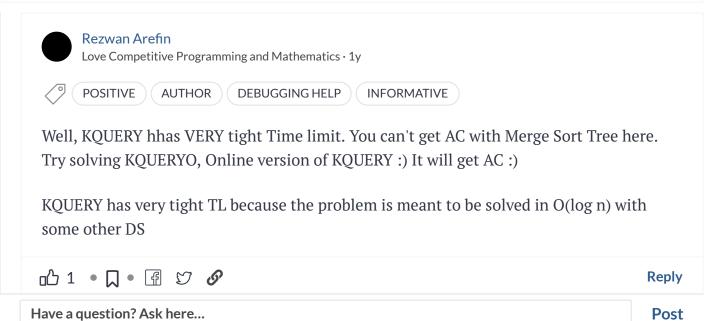


Limited-time discount for early adopters | Pro Membership is only \$3 when paid yearly. 70% OFF

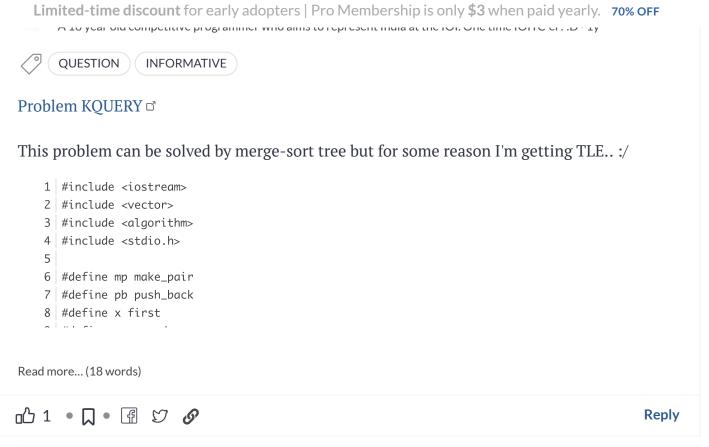


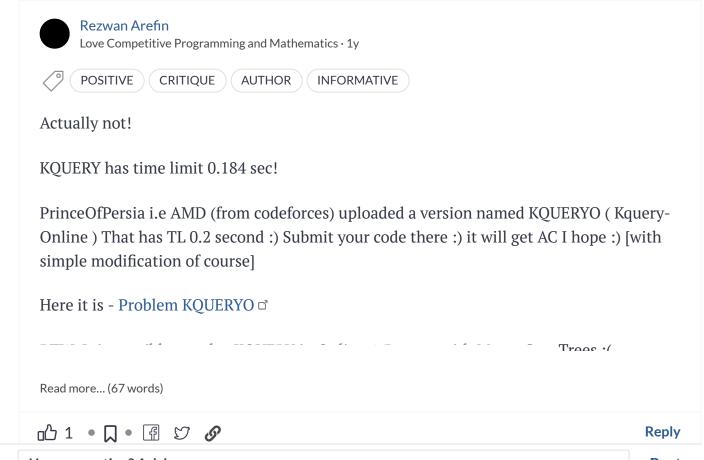
SOLUTION



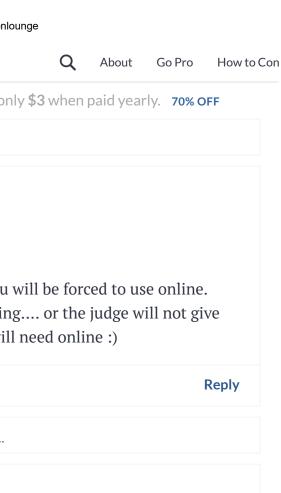








Have a question? Ask here...



Limited-time discount for early adopters | Pro Membership is only \$3 when paid yearly. 70% OFF Udit Sanghi • 1y • The first question can also be done using offline ... Rezwan Arefin Love Competitive Programming and Mathematics • 1y **POSITIVE AUTHOR INFORMATIVE** But you need to know the online solution. Sometimes you will be forced to use online. Like the next query = answer of previous query ^ something.... or the judge will not give you another query unless you flush stdout :P Then you will need online :) Udit Sanghi • 1y • Yes I know that it is important to know different ... Rezwan Arefin Love Competitive Programming and Mathematics • 1y **POSITIVE AUTHOR** QUESTION **INFORMATIVE** Btw. Do you know how to solve this by Persistent Segment Tree? I guess that will decrease time complexity:) current complexity is O(log^2 n) right? I guess using persistent segment tree will be O(log n) ^^^I think it is better to use BIT instead of Segment Tree for this specific problem :) BIT works fast :) At first I solve the problem using Offline BIT too :) Reply Udit Sanghi • 1y • I think BIT and seg tree will be equally fast. And... Tanavya Dimri • 1y • BIT and SegTree won't be equally fast as SegTrees ... Udit Sanghi • 1y • Actually I just wanted to ask this that y do v use... Tanavya Dimri A 16 year old competitive programmer who aims to represent India at the IOI. One time IOITC-er! :D \cdot 1y

Have a question? Ask here...

Q About Go Pro How to Con

Limited-time discount for early adopters | Pro Membership is only \$3 when paid yearly. 70% OFF

you learnt segment trees you were given these rather complex functions.. rou wouldn't have found it as simple and obvious as you see segment trees now.

- 2. Moreover, with practice, coding time for either will be about equally fast, so there's very little point in spending so much time learning this.
- 3. Recursive solution is more flexible for different sort of problems.
- 4. The speed gain from using iterative solution won't be huge, and a recursive

Read more... (117 words)



Reply

SHOW 3 MORE REPLIES

SHOW 3 MORE REPLIES

See All Replies →

RELATED COURSES

Learn Algorithms and Data Structures

26 tutorials

Competitive Programming: From Beginner to Expert

88 tutorials

RELATED ARTICLES

Have a question? Ask here...

Q

About

Go Pro

How to Con

Limited-time discount for early adopters | Pro Membership is only \$3 when paid yearly. 70% OFF

Suffix Arrays

21 minute read

MO's Algorithm (Query square root decomposition)

12 minute read

Heavy Light Decomposition

22 minute read

Ready to join our community?











Sign up below to automatically get notified of new courses, get **reminders** to finish ones you subscribe to, and **bookmark** lessons to read later.

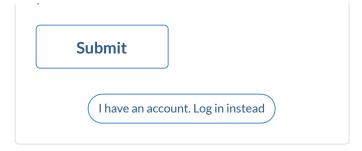
Continue with Facebook

- OR -

Have a question? Ask here...



Limited-time discount for early adopters | Pro Membership is only \$3 when paid yearly. 70% OFF



By signing up, you agree to our Terms and our Privacy Policy.



POPULAR COURSES

Machine Learning

Deep Learning

Natural Language Processing

Big Data

Data Science

Bioinformatics

Algorithms

Cryptography

NEW COURSES

Web Development

UX & UI Design

Startups

Product Management

Cryptocurrencies

Finance

College Admissions

Have a question? Ask here...

Q About Go Pro How to Con

Limited-time discount for early adopters | Pro Membership is only \$3 when paid yearly. 70% OFF

How to Contribute

Upgrade to Pro

Help and FAQ

Commonlounge Meta

Team

Privacy, Terms & Refunds

GET IN TOUCH

hello@commonlounge.com

Facebook

Twitter

Copyright 2016-18, Compose Labs Inc. All rights reserved.

Have a question? Ask here...