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 .... 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

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 -

```
const int maxn = 1e5+10;
vector<int> tree[4*maxn];
int n, m, arr[maxn];
#define all(v) v.begin(), v.end()
void build(int node, int l, int r) {
   if(l == r) {
      tree[node].push_back(arr[l]);
      return;
   }
   int mid = l + r >> 1,
   left = node << 1, right = left|1;

   build(left, l, mid);
   build(right, mid+1, r);

   merge(all(tree[left]), all(tree[right]),
           back_inserter(tree[node]));
}
```

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.

Query: Query part is quite easy, here is a simple implementation-

```
int query(int node, int l, int r, int i, int j, int k) {
   if(i > r || l > j) return 0;
   if(i <= l && r <= j) {
      return lower_bound(all(tree[node]), k)
                  - tree[node].begin();
   }
   int mid = l + r >> 1,
      left = node << 1, right = left|1;
   return query(left, l, mid, i, j, k) +
         query(right, mid+1, r, i, j, k);
}
```

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 n) and updating will be O(log^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

You can simply use this is the elements of the array are distinct! But when the are not

Tutorial Segment tree Merge sort Data structure

☐ MARK AS COMPLETED

★★★★★ Leave a Review

< Previous
[FENTREE] Fenwick Trees

Next >
Persistent segment trees: Explained with SPOJ problems

7 Reply

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:

Rezwan Arefin
Love Competitive Programming and Mathematics
100%

DISCUSSION

HINT

Rezwan Arefin
Love Competitive Programming and Mathematics · 1y

HINT AUTHOR INFORMATIVE

Hint 1:

Hidden content: Tap to show!

Hidden content: Tap to show!

Have a question? Ask here... Post

Read more... (203 words)

3 • • Reply

SOLUTION

Shubham Chandra
1y

SOLUTION DEBUGGING HELP

Here is my solution for http://www.spoj.com/problems/KQUERY/

I am using mergesort tree and fast IO, I have no idea why this is giving TLE. Please check my binary search code which I suspect can be a reason for TLE .. PLZ help.

```
#include <bits/stdc++.h>
#define maxn 30000
using namespace std;

vector <int> tree[4*maxn];
int a[maxn + 5];

```

Read more... (41 words)

• • Reply

Rezwan Arefin
Love Competitive Programming and Mathematics · 1y

POSITIVE AUTHOR DEBUGGING HELP INFORMATIVE

Well, KQUERY hhas VERY tight Time limit. You can't get AC with Merge Sort Tree here. Try solving KQUERYO, Online version of KQUERY :) It will get AC :)

KQUERY has very tight TL because the problem is meant to be solved in O(log n) with some other DS

1 • • Reply

Have a question? Ask here... Post

A 16 year old competitive programmer who aims to represent India at the IOI. One time IOITC cr . :D · 1y

QUESTION INFORMATIVE

Problem KQUERY

This problem can be solved by merge-sort tree but for some reason I'm getting TLE.. :/

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <stdio.h>

#define mp make_pair
#define pb push_back
#define x first
```

Read more... (18 words)

1 • Reply

Rezwan Arefin
Love Competitive Programming and Mathematics · 1y

POSITIVE CRITIQUE AUTHOR INFORMATIVE

Actually not!

KQUERY has time limit 0.184 sec!

PrinceOfPersia i.e AMD (from codeforces) uploaded a version named KQUERYO (Kquery-Online) That has TL 0.2 second :) Submit your code there :) it will get AC I hope :) [with simple modification of course]

Here it is - Problem KQUERYO

Trees :(

Read more... (67 words)

1 • Reply

Have a question? Ask here... Post

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 :)

1 Reply

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 · 1y

Have a question? Ask here... Post

you learnt segment trees you were given these rather complex functions.. You 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 be the test cases

Read more... (117 words)

3 • • 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... Post

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...

Post

Submit

I have an account. Log in instead

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

Back to top

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

How to Contribute

Upgrade to Pro

Help and FAQ

Commonlounge Meta

Team

Privacy, Terms & Refunds

GET IN TOUCH

hello@commonlounge.com

Facebook

Twitter

Have a question? Ask here... **Post**