CodeChef Discussion

Merge Sort Tree - Tutorial

35

20

Prerequisites : Segment Trees

Target Problem : Given an array of N elements and you have to answer Q queries of the form L R K , To Count the numbers smaller than K in range L to R.

Key Idea : The key idea is to build a Segment Tree with a vector at every node and the vector contains all the elements of the subrange in a sorted order . And if you observe this segment tree structure this is some what similar to the tree formed during the merge sort algorithm (Yes , thats why they are called Merge Sort Trees) .

Building the tree :

```
vector<int>tree[5*N];
int A[N];
Void build_tree( int cur , int l , int r )
{
     if( l==r )
     {
            tree[cur].push_back( a[ l ] );
            return ;
     }
     int mid = l+(r-l)/2;
     build_tree(2*cur+1 , l , mid ); // Build left tree
     build_tree(2*cur+2 , mid+1 , r ); // Build right tree
     tree[cur] = merge( tree[2*cur+1] , tree[2*cur+2] ); //Merging the two sorted arrays
}
```

Querying the tree :

```
int query( int cur, int l, int r, int x, int y, int k)
{
       if( r < x || l > y )
      {
               return 0; //out of range
      }
      if( x<=l && r<=y )
      {
              //Binary search over the current sorted vector to find elements smaller than
K

              Return upper_bound(tree[cur].begin(),tree[cur].end(),K)-tree[cur].begin();
      }
      int mid=l+(r-l)/2;
     return query(2*cur+1,l,mid,x,y,k)+query(2*cur+2,mid+1,r,x,y,k);
}
```

Build function Analysis : Build a merge sort tree takes O(NlogN) time which is same as Merge Sort Algorithm . It will take O(NlogN) memory because each number Ai will be present in at most LogN vectors (Height of the tree) .

Query function Analysis : A range L to R can divided into at most Log(N) parts, where we will perform binary search on each part . So this gives us complexity of O(LogN * LogN) per query .

Handling Point Updates : The only reason that we cant handle updates on MST in this code is because its merge function takes too much time, so even if theres a point update it will lead to O(N). So the major issue is of vectors and rearranging them on updations, but why do we need vectors ? Just to find the elements smaller than K in that complete vector, right ? Lets forget about vectors and keep policy based data structure at each node which handles three queries (insertion , deletion and elements smaller than K in the set) in O(LogN) time . So now no need to rearrange vectors and we can use insertion - deletion to handle point queries . This is just an idea , we can discuss this in comments again if anyone has a doubt .

Bonus : How to use this to solve the query of type Kth smallest number in range L to R ? So we can binary search over the solution and find the value which has exactly K numbers smaller than it in the given range . Complexity : O(LogN * LogN * LogAi) per query .

Why to use MST: Apart from the code simplicity, they answer queries online . We could have used some offline algorithms like MOs or even Segment tree but come on, Online Querying is great because it can be used in Dp Optimisations and stuff like that . Peace Out .

edited 27 Mar '17, 20:22

asked 27 Mar '17, 20:07

segment-tree mergesort

Question tags:

segment-tree **×1,470**
mergesort **×133**
merge-sort-tree **×50**

question asked: **27 Mar '17, 20:0**
question was seen: **10,215 times**
last updated: **07 Jan, 18:33**

Related questions

- Confusion in complexity of buidling merge sort tree
- PSHTRG - Editorial
- Why some test cases are showing WA?
- Suggest a Suitable DS
- Flip Coins . Getting Wrong Answer Ple Help
- SPOJ RRANGE
- Heavy-Light Decomposition
- getting WA in 'the problem set' quest from ACMKAN14!
- tutorial for segment tree
- String Manipulation

6★ arunnsit
[1.0k]●1●5●11
accept rate: 27%

Thank you Sir. This is of great help!!

vishesh_345 (18 Jul '17, 17:19)

---

7 Answers: oldest answers newest answers popular answers

9

You can do Kth smallest number query in range L to R, in $O(n * \log^2 n)$ by building the merge sort tree on indices instead of the values. This solution doesn't depend on the values of $A[i]$, however large they may be.

For implementation details, one may refer to this code for MKTHNUM problem on Spoj.

link | award points

answered **28 Mar '17, 00:33**

6★ likecs
[3.4k]●10●50
accept rate: 9%

@likecs I have written the exactly same solution as you did, but in java. I am getting TLE. Can you please help me? Note : I am using the fast I/O also. Link to my code : http://ideone.com/pHN7Z2

5★ mayank12559 (14 Jun '17, 22:30)

@mayank12559, I tried to optimise your solution by making the binary serach iterative as well, but no sucess. May be the time limits of the problem are strict for java for passing with O(log^2n) approach. Also, only 4 solutions in java for this problem till date. May be you can try with C++.

6★ likecs (15 Jun '17, 00:39)

Thanks for the reply. I will try to write the same in c++. Java solution should also get accepted though, as c++ solutions are getting accepted with same complexity. There should not be any discrimination among languages.

5★ mayank12559 (15 Jun '17, 08:22)

@likecs I have written the code in c++. On submission I am getting Wrong Answer now. Can you please help me once again? Link to my code : http://ideone.com/qC2tVy

5★ mayank12559 (15 Jun '17, 16:54)

3

I'm confused.. Why would the complexity have a factor of logAi in the "Bonus" part. I can only think of a O(q*log^3 n) solution. Binary search over the array and find the value that has k-1 smaller numbers in O(log^2 n).

Also, would appreciate if you could upvote me. Need karma points :P Thanks!

link | award points edited **28 Apr '17, 08:38**

answered **28 Apr '17, 08:35**

1★ sinbycos
[33]●2
accept rate: 0%

2 yes, you are correct too. thats one way to see it. if you binary search over the array it will come out to be O(q*log^3 n). but if you binary search over the a[i] value then it comes out to be O(*q*log^2 n*log ai).

6★ arunnsit (07 May '17, 03:56)

But that's only better if ai < N, right?

1★ sinbycos (09 Jun '17, 08:25)

1

how can you handle updates here..?

link | award points

answered **28 Apr '17, 18:13**

5★ rajarshi_basu
[467]●6
accept rate: 10%

i have already mentioned about point updates. is there anything that you didnt understand?

6★ arunnsit (07 May '17, 03:58)

0

For the bonus part, consider a *query(L,R,X)*, and *X=R-L+1*, which means that you need to find out the maximum number from the range *L* to *R*. According to your solution, we can binary search on *Ai*, and find out which number gives *number of numbers <= Ai* to be X. But how will you keep a track of *Ai*s which are present in the range *[L,R]* of the original array *A*? This is needed because for the case where you need to find the maximum in a range *[L,R]*, any number *Y* lying between the maximum number from *[L,R]* and the next maximum in the array *A* will also yield *X* as the answer to the *query(L,R,Y)*.

link | award points

answered **31 May '17, 19:54**

2★ s1_73
[1]
accept rate: 0%

i think you aren't clear about binary search, we will binary search for the smallest number that satisfies the above condition i.e. for query(L,R,X), and X=R-L+1.find out the smallest number which gives number of numbers <= Ai to be X. Theres no need to track its presence in array.

6★ arunnsit (03 Jun '17, 14:07)

0

For elements of size 10^5 ill have to define vector<int> tree[log2(10^5) * 10^5] right?

It's giving me runtime error for such big chunk of memory in c++.

Please clarify! Thanks!

link | award points

answered **21 Jun '17, 08:20**

3★ manjrekarom29
[21]●1●3
accept rate: 0%

1 Why do you think log2(10^5) * 10^5 vectors are required? vectors should be equal to number N+N/2+N/4+N/8....(i.e number of nodes in segtree) Which is 2*N but we require a number which is multiple of 2 and greater than* 2N. So to be on safe side we usually take 4*N nodes in segtree.

6★ arunnsit (22 Jun '17, 18:15)

Oh yes! I got confused. Thanks

3★ manjrekarom29 (23 Jun '17, 08:24)

0 We can achieve much better. First, we compress the array in order to contain only values in range [0,N). We keep a table to associate a compressed value to the original one in O(1) and associate a normal value to his lower_bound in compressed range in O(logN). This take us so far NlogN. Then we build a Persistent Segment Tree, where for every time i we have stored the frequencies of each value in the array with index smaller or equal to i, and every node store the sum of the two sons. Overall is NlogN preprocessing. Every time we want to know how many integers there are in range [l,r] smaller than k, we do as follows:

- first we replace k with the compressed value of the greater original value smaller than k. //logN
- We query the Persistent Segmentent Tree in the interval [0,k], assuming that the value written in each node is the one of the tree(r) - tree(l-1).

In the end we use NlogN space and time complexity for preprocessing, and each query take logN. But we can do a lot of amazing other stuff, for example, being asked about the k-th smallest value in range [l,r], and answer in logN as well.

link | award points

answered **02 Jul '17, 17:17**

3★ lukecavabarret
[1]
accept rate: 0%

1 yeah, Persistent Segment Tree can solve such problems but do you think its as easy to explain and understand as this?

6★ arunnsit (06 Jul '17, 13:13)

maybe. After all, the only important things -after understand segment tree in its pointer implementation- are: -In each update, at most logN node change -In each update, create a new version of all nodes changed

3★ lukecavabarret (19 Jul '17, 22:20)

0 Can anyone help me in the spoj problem MKTHNUM and tell me why I am getting wrong answer? MY SOLUTION

link | award points

answered **07 Jan, 18:33**

3★ parth_patel15
[3]●2
accept rate: 0%

Your answer

[hide preview] community wiki:

Preview

reCAPTCHA V1 IS SHUTDOWN
Direct site owners to g.co/recaptcha/upgrade

Type the text

Privacy & Terms

Answer the question