rekt_n00b's blog

Mo's Algorithm on Trees [Tutorial]

By **rekt_n00b**, history, 9 months ago,

Introduction

Mo's Algorithm has become pretty popular in the past few years and is now considered as a pretty standard technique in the world of Competitive Programming. This blog will describe a method to generalize Mo's algorithm to maintain information about paths between nodes in a tree.

Prerequisites

Mo's Algorithm — If you do not know this yet, read this amazing article before continuing with this blog.

Preorder Traversal or DFS Order of the Tree.

Problem 1 — Handling Subtree Queries

Consider the following problem. You will be given a rooted Tree T of N nodes where each node is associated with a value $A[node]$. You need to handle Q queries, each comprising one integer u. In each query you must report the number of distinct values in the subtree rooted at u. In other words, if you store all the values in the subtree rooted at u in a set, what would be the size of this set?

Constraints

$1 \le N, Q \le 10^5$

$1 \le A[node] \le 10^9$

Solution(s)

Seems pretty simple, doesn't it? One easy way to solve this is to flatten the tree into an array by doing a Preorder traversal and then implement Mo's Algorithm. Maintain a lookup table which maintains the frequency of each value in the current window. By maintaining this, the answer can be updated easily. The complexity of this algorithm would be $O(Q\sqrt{N})$

Note that you can also solve this in $O(N \log^2 N)$ by maintaining a set in each node and merging the smaller set into the larger ones.

Problem 2 — Handling Path Queries

Now let's modify Problem 1 a little. Instead of computing the number of distinct values in a subtree, compute the number of distinct values in the unique path from u to v. I recommend you to pause here and try solving the problem for a while. The constraints of this problem are the same as Problem 1.

The Issue

An important reason why Problem (1) worked beautifully was because the dfs-order traversal made it possible to represent any subtree as a contiguous range in an array. Thus the problem was reduced to "finding number of distinct values in a subarray $[L, R]$ of $A[]$. Note that it is not possible to do so for path queries, as nodes which are $O(N)$ distance apart in the tree might be $O(1)$ distance apart in the flattened tree (represented by Array $A[]$). So doing a normal dfs-order would not work out.

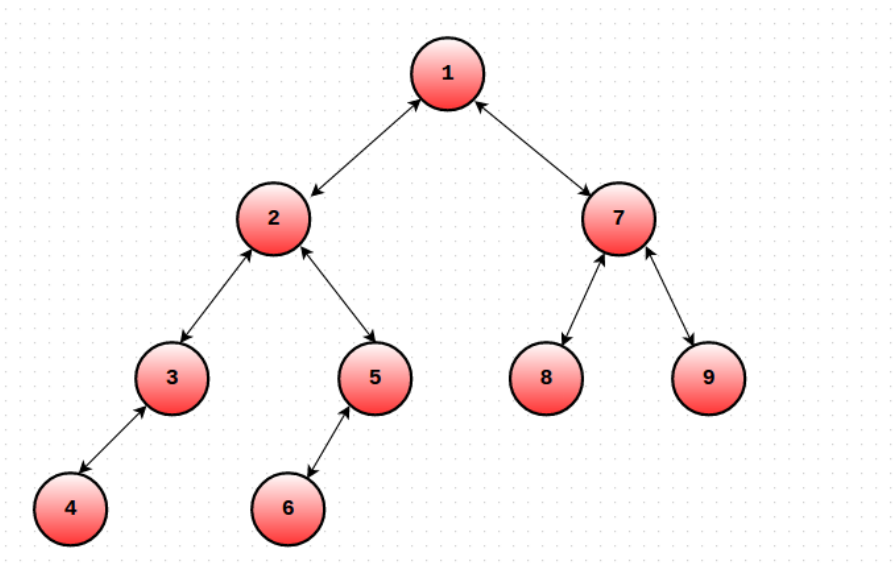
Observation(s)

Let a node u have k children. Let us number them as $v_1, v_2 \ldots v_k$. Let $S(u)$ denote the subtree rooted at u.

Let us assume that $dfs()$ will visit u's children in the order $v_1, v_2 \ldots v_k$. Let x be any node in $S(v_i)$ and y be any node in $S(v_j)$ and let $i < j$. Notice that $dfs(y)$ will be called only after $dfs(x)$ has been completed and $S(x)$ has been explored. Thus, before we call $dfs(y)$, we would have entered and exited $S(x)$. We will exploit this seemingly obvious property of $dfs()$ to modify our existing algorithm and try to represent each query as a contiguous range in a flattened array.

Modified DFS-Order

Let us modify the dfs order as follows. For each node u, maintain the Start and End time of $S(u)$. Let's call them $ST(u)$ and $EN(u)$. The only change you need to make is that you must increment the global timekeeping variable even when you finish traversing some subtree ($EN(u) = ++cur$). In short, we will maintain 2 values for each node u. One will denote the time when you entered $S(u)$ and the other would denote the time when you exited $S(u)$. Consider the tree in the picture. Given below are the $ST()$ and $EN()$ values of the nodes.



$ST(1) = 1$ $EN(1) = 18$

$ST(2) = 2$ $EN(2) = 11$

$ST(3) = 3$ $EN(3) = 6$

$ST(4) = 4$ $EN(4) = 5$

$ST(5) = 7$ $EN(5) = 10$

$ST(6) = 8$ $EN(6) = 9$

$ST(7) = 12$ $EN(7) = 17$

$ST(8) = 13$ $EN(8) = 14$

$ST(9) = 15$ $EN(9) = 16$

$A[] = \{1, 2, 3, 4, 4, 3, 5, 6, 6, 5, 2, 7, 8, 8, 9, 9, 7, 1\}$

The Algorithm

Now that we're equipped with the necessary weapons, let's understand how to process the queries.

Let a query be (u, v). We will try to map each query to a range in the flattened array. Let $ST(u) \leq ST(v)$ where $ST(u)$ denotes visit time of node u in T. Let $P = LCA(u, v)$ denote the lowest common ancestor of nodes u and v. There are 2 possible cases:

$Case\ 1: P = u$

In this case, our query range would be $[ST(u), ST(v)]$. Why will this work?

Consider any node x that does not lie in the (u, v) path.
Notice that x occurs twice or zero times in our specified query range.
Therefore, the nodes which occur exactly once in this range are precisely those that are on the (u, v) path! (Try to convince yourself of why this is true : It's all because of $dfs()$ properties.)

This forms the crux of our algorithm. While implementing Mo's, our add/remove function needs to check the number of times a particular node appears in a range. If it occurs twice (or zero times), then we don't take it's value into account! This can be easily implemented while moving the left and right pointers.

$Case\ 2: P \neq u$

In this case, our query range would be $[EN(u), ST(v)] + [ST(P), ST(P)]$.

The same logic as Case 1 applies here as well. The only difference is that we need to consider the value of P i.e the LCA separately, as it would not be counted in the query range.

This same problem is available on SPOJ.

If you aren't sure about some elements of this algorithm, take a look at this neat code.

Conclusion

We have effectively managed to reduce problem (2) to number of distinct values in a subarray by doing some careful bookkeeping. Now we can solve the problem in $O(Q\sqrt{N})$. This modified DFS order works brilliantly to handle any type path queries and works well with Mo's algo. We can use a similar approach to solve many types of path query problems.

For example, consider the question of finding number of inversions in a (u, v) path in a Tree T, where each node has a value associated with it. This can now be solved in $O(Q\sqrt{N}\log N)$ by using the above technique and maintaining a BIT or Segment Tree.

This is my first blog and I apologize for any mistakes that I may have made. I would like to thank **sidhant** for helping me understand this technique.

Sample Problems

1) Count on a Tree II
2) Frank Sinatra — Problem F
3) Vasya and Little Bear

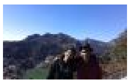
Thanks a lot for reading!

Original Post
Related Blog

+406 rekt_n00b 9 months ago 73

Comments (73)

Write comment?


meintoo

9 months ago, # | ☆ ← Rev. 2 +8

In this case, our query range would be [EN(u), ST(v)] + [ST(P), ST(P)].
Shouldn't it be **[EN(u), ST(v)] + [ST(P), ST(U)]** or am I missing something ?
→ Reply


rekt_n00b

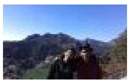
9 months ago, # ^ | ☆ ← Rev. 2 +8

Nope, it will be $[ST(P), ST(P)]$.

Consider the path from 3 to 5. In this case, $P = 2$

$EN(3) = 6$ and $ST(5) = 7$, so we consider the range $[6, 7]$ in $A[]$ corresponding to the nodes $[3, 5]$ giving us the values of nodes 3 and 5.

Our query range does not consider the value of the lca as $ST(P) < EN(u) < ST(v) < EN(P)$. Hence we must account for the value of P separately.
→ Reply


meintoo

9 months ago, # ^ | ☆ +8

Ok !!!
Thanks
Nice article anyways
→ Reply


rekt_n00b

9 months ago, # ^ | ☆ +13

Thank You :D
→ Reply


sampriti

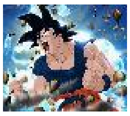
9 months ago, # | ☆ +23

Thanks! That was a really nice tutorial!
→ Reply


rekt_n00b

9 months ago, # ^ | ☆ +13

Thanks a lot :)
→ Reply


brainstorm

9 months ago, # | ☆ 0

Nice tutorial :)
can you give links to some more problems on which similar approach can be applied ?
→ Reply


sbansalcs

4 months ago, # ^ | ☆ 0

http://codeforces.com/problemset/problem/375/D
→ Reply


rekt_n00b

4 months ago, # ^ | ☆ 0

This can be done with standard Mo's Algorithm, because the queries are on subtrees and not paths.
→ Reply


sbansalcs

4 months ago, # ^ | ☆ 0

Oh sorry, I thought this guy was asking for any problems related to the algorithms described above.
→ Reply

9 months ago, # | ☆ ← Rev. 2 +5

BTW, we can find number of distinct values in a subarray [l, r] of a offline in

BTW, we can find number of distinct values in a subarray [l, r] of a online in O((q+n)\log n).

komendart

Let's sort all queries by l_i.

$d_i = 1$ if i is the first occurence of a_i in a[l...n] otherwise $d_i = 0$.

So, query (l_i, r_i) is finding sum of d[l_i...r_i].

When we move from query with l_i to query with l_i + 1 we must update only one or zero elements of d_i. It can be done in $O(\log n)$ if we precalculated for each i next occurence of a_i in array.

→ Reply

gongy

3 months ago, # ^ | ☆ +10

If you maintain the tree persistently, you can have an online solution as well.

→ Reply

bluemmb

9 months ago, # | ☆ 0

Thanks, Is known who used this idea on trees first time ?

→ Reply

belltolls

9 months ago, # ^ | ☆ +5

It must have been known from before. But I guess this is the first proper tutorial/blog for it.

→ Reply

I_love_Pro

8 months ago, # | ☆ +1

Totally went over my head! Excellent blog!

→ Reply

as_true

3 months ago, # ^ | ☆ +3

If it goes over your head, How do you realize it's an excellent ?

→ Reply

svg_af

8 months ago, # | ☆ 0

Thanks a lot you made my day !!

I've been obsessing about COT2 for almost two months without anything that comes to mind

if only i could upvote more than once

→ Reply

rekt_n00b

8 months ago, # ^ | ☆ +1

Thanks! I'm glad that you found it useful :)

→ Reply

rachitjain

8 months ago, # | ☆ 0

I implemented this algorithm on the COT2 problem on SPOJ(http://www.spoj.com/problems/COT2/). I am getting WA. Can someone help me identify the bug in my code? http://ideone.com/aLS5Yx

→ Reply

rachitjain

8 months ago, # ^ | ☆ +8

I found the mistake. Thanks for the nice tutorial.

→ Reply


SarvagyaAgarwal

7 weeks ago, # ^ | ☆ 0

What was the bug ?

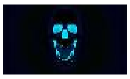
→ Reply


rachitjain

7 weeks ago, # ^ | ☆ 0

Lol. Bro that was 7 months ago.

→ Reply


SarvagyaAgarwal

7 weeks ago, # ^ | ☆ 0

Can you share your corrected code ? Because I'm getting a WA too .

→ Reply


rachitjain

7 weeks ago, # ← Rev. 2 ☆ 0

Sure. Link

→ Reply

mlv68

4 days ago, # ^ | ☆ 0

Can u please explain ur add and del functions. How are u maintaining the result after ignoring all those indexes which have occured 2 times?

→ Reply


rachitjain

4 days ago, # ^ | ☆ 0

Recently I solved one question using Mo's algorithm, and I remembered about this comment here. I overwrote the solution on the same link. Here is the solution for COT2. I think its self-explanatory how it is working.

→ Reply

NiKS001

8 months ago, # | ☆ ← Rev. 2 0

Can someone please provide the $O(N\log^2 N)$ algorithm for Problem 1?

The best I could get is (N^2)*logN [as the sum of sizes of sets of each node is O(N^2) — Worst case linear graph with all values distinct]

→ Reply


rekt_n00b

8 months ago, # ^ | ☆ ← Rev. 4 +3

Maintain a set of values for each node in the tree. Let $set(u)$ be the set of all values in the subtree rooted at u. We want $size(set(u))$ for all u.

Let a node u have k children, $v_1, v_2 ... v_k$. Every time you want to merge $set(u)$ with $set(v_i)$, pop out the elements from the smaller set and insert them into the larger one. You can think of it like implementing union find, based on size.

Consider any arbitrary node value. Every time you remove it from a certain set and insert it into some other, the size of the merged set is atleast twice the size of the original.

Say you merge sets x and y. Assume $size(x) \leq size(y)$. Therefore, by the algorithm, you will push all the elements of x into y. Let xy be the merged set. $size(xy) = size(x) + size(y)$. But $size(y) \geq size(x)$.

So $size(xy) \geq 2 * size(x)$.

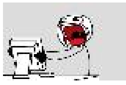
Thus, each value will not move more than $log\ n$ times. Since each

Thus, each value will not move more than $log\ n$ times. Since each move is done in $O(logn)$, the total complexity for n values amounts to $O(nlog^2\ n)$

Code
→ Reply

NiKS001
8 months ago, # ^ | ☆ +5
Awesome! Thanks for the great explanation and code!
→ Reply


safayet007
7 weeks ago, # ^ | ☆ ← Rev. 2 0
What if set x and set y isn't completely disjoint? In that case size(xy) = size(x) + size(y) statement isn't valid. Since the value on two nodes might be same.
→ Reply


arrogantIdiot
8 months ago, # | ☆ 0
Very neatly written tutorial. You make it seem amazingly easy!
→ Reply

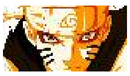

rekt_n00b
8 months ago, # ^ | ☆ 0
Thanks a lot :)
→ Reply


mbrc
8 months ago, # | ☆ +11
Superb idea! :D

Thanks! :D
→ Reply

demon_cross
8 months ago, # | ☆ 0
Nicely written!
→ Reply


rekt_n00b
8 months ago, # ^ | ☆ 0
Thanks a lot :D
→ Reply


naruto09
6 months ago, # | ☆ 0
shouldn't it be end time of u to start time of v in case 1.If we start from start time of u then u will be included 2 times one for its start time and once for end time.Correct me if i am wrong..
→ Reply


rekt_n00b
6 months ago, # ^ | ☆ 0
Case 1 implies that u is an ancestor of v.
Therefore, we won't visit u twice in the range $[ST(u), ST(v)]$ as $EN(u) > ST(v)$.
→ Reply


baobab
6 months ago, # | ☆ 0
Has anyone managed to get accepted on the SPOJ problem with a Java solution? I'm getting NZEC Runtime Error, but it looks like it's actually due to time limit exceeding.
→ Reply


rekt_n00b
6 months ago, # | ☆ +3
Update 1: Added sample problems.
→ Reply

Absolut

4 months ago, # ^ | ☆ 0

For the "Frank Sinatra" problem. How could you find the less value not present in the path?

I realize that any value greather than the size of the tree wouldn't change the answer. So, if i have at most 1E5 different values I can build a BIT. pos[i] = 1 if value i is present in the path. Then I binary search the less value k wich sum[0...k] is less than k. That would be my answer. However the complexity is O(N*sqrt(N)*log(N)*log(N)) and I think is excesive.

→ Reply

rekt_n00b

4 months ago, # ^ | ☆ 0

The complexity wouldn't be $O(N\sqrt{N}log^2N)$, it would be $O(N\sqrt{N}logN + Nlog^2N)$.

The first term is because you update your bit atmost $N\sqrt{N}$ times and the second term is because you binary search once for each query.

→ Reply

Absolut

4 months ago, # ^ | ☆ 0

Thanks, my mistake.

So, it is the best completely? Or there is another approach

→ Reply

rekt_n00b

4 months ago, # ^ | ☆ 0

You can solve the problem in $O(N\sqrt{N})$ by doing square root decomposition on the values. Each update would be done in constant time and you will take additional $\sqrt{N}$ time per query to find the $block$ which has the smallest value.

Code

→ Reply

SProf

6 months ago, # | ☆ 0

can it gives me tle,if i can't use weight compress?

→ Reply

rekt_n00b

6 months ago, # ^ | ☆ 0

If you do not compress weights, you'll need a map and that would add an additional $log(n)$ factor. However, you might be able to squeeze your solution within the TL with an `unordered_map` .

→ Reply

meshanya

6 months ago, # | ☆ +39

BTW, there is a standard solution for the first problem (see this link in Russian). For each of the colors order all the vertices of this color according to the dfs traversal, let the vertices be labelled $v_1, v_2, ..., v_k$. Add +1 to each of these verticies, and add -1 to the LCAs of the neighboring vertices $lca(v_1, v_2), lca(v_2, v_3), ..., lca(v_{k-1}, v_k)$. If you sum up the values inside a subtree, you get the number of distinct elements in it.

Since the ordering can be done in $O(n)$, and in theory you can answer lca queries for a static tree in $O(1)$ with $O(n)$ pre-processing, you have a linear solution (assuming $0 \le A[x] < N$).

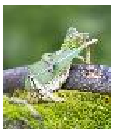
→ Reply

6 months ago, # ^ | ☆ 0

Thanks! This idea is pretty cool :)

Thanks! This idea is pretty cool :)
→ Reply

rekt_n00b



SProf

6 months ago, # | ☆ 0

in problem frank sinatra, can i quickly find minimal number,that is not present in given set.
→ Reply

howsiwei

6 months ago, # | ☆ +5

Isn't the time complexity of Mo's algorithm O(N*sqrt(Q)) instead of O(Q*sqrt(N))?
→ Reply

rekt_n00b

6 months ago, # ^ | ☆ 0

The complexity of Mo's depends on the number of times we increment/decrement the curL, curR variables. This link explains the time complexity of Mo's algorithm.
→ Reply

howsiwei

6 months ago, # ^ | ☆ ← Rev. 2 +10

If the size of each block is k, then the time complexity of moving the left pointer is O(Q*k) and the time complexity of moving the right pointer is O(N/k*N). The optimal value of k is N/sqrt(Q) which results in total time complexity O(N*sqrt(Q)).
→ Reply

rekt_n00b

6 months ago, # ^ | ☆ 0

That is true. However, in most cases upper bounds on Q and N are equal (or pretty close), so it doesn't make a difference.
→ Reply

aka.oliver

6 months ago, # | ☆ 0

I wrote the code for COT2 judge gives runtime error at 10th testcase pls help me i cant find the error thanks in advance

pls see my code

https://ideone.com/MG3XbK
→ Reply

sbansalcs

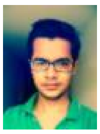
4 months ago, # | ☆ +19

Awesome Tutorial!
→ Reply

rekt_n00b

4 months ago, # ^ | ☆ +3

Thank you!
→ Reply



alphaguy4

3 months ago, # | ☆ 0

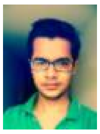
Can anyone explain how to linearize the tree .. (Not binary tree but any tree in general)

As in Problem 1..
→ Reply

rekt_n00b

3 months ago, # ^ | ☆ 0

Click
→ Reply



alphaguy4

3 months ago, # ^ | ☆ 0

Thanks alot... !

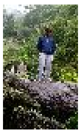
→ Reply

uttom

3 months ago, # | ☆ 0

I got Runtime Error. Here is my code There is any wrong my crealting lca tabel or anything else. Thanks in advance.

→ Reply



ankeshgupta007

3 months ago, # | ☆ 0

amazing tutorial!

→ Reply

stould

3 months ago, # | ☆ ← Rev. 3 0

If the tree store the values on the edges, you could store these values on the children (going from the root), and change the $Case\ 2$ to:

```
if(P == u || P == v) check(P);
.... asnwer the query
if(P == u || P == v) check(P);
```

→ Reply

RobertBruce

3 months ago, # | ☆ 0

Why don't you write more tutorials?

→ Reply

EbraM96

7 weeks ago, # | ☆ 0

http://www.spoj.com/problems/DQUERY/ a practice problem

→ Reply

SarvagyaAgarwal

7 weeks ago, # | ☆ +3

COT2 code link doesnt work .

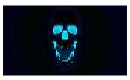
→ Reply

rekt_n00b

7 weeks ago, # ^ | ☆ +7

Updated.

→ Reply



SarvagyaAgarwal

7 weeks ago, # | ☆ 0

Why does this get WA for COT2 :/ ?

→ Reply

vibhorgoyal97

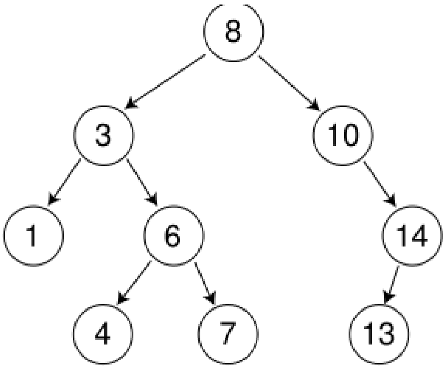
3 weeks ago, # | ☆ +5

Could you explain your idea for the problem of finding number of inversions in a (u, v) path in a Tree T.

→ Reply

2 weeks ago, # | ☆ ← Rev. 2 0

Vicennial



If I flatten the above tree, my array would be:
8 3 1 6 4 7 10 14 13
Suppose I need to use Mo's algorithm for subtrees(assume I need to find sum of values of each subtree indicated by the query)
For a given query 'Vj' how would I find its end range index in the array?
Eg if given query is node '6', the starting range would be idx 3 and ending would be idx 5.

→ Reply

SarvagyaAgarwal

2 weeks ago, # ^ | ☆ +1

Store the starting and ending times for every node during your dfs
.

→ Reply

Vicennial

2 weeks ago, # ^ | ☆ 0

Thanks, understood it after a bit of googling about discovery/begin/end times.

→ Reply

Tobby_And_Friends

5 days ago, # | ☆ 0

For a problem like this: http://lightoj.com/volume_showproblem.php?problem=1348 where I need to return sum of all the nodes in a given path & update the value of a node, how should I approach using this technique of linearizing the tree? I mean since I need to ignore nodes which have occurrence of 2 so the range becomes discontinuous for a segment tree structure.

→ Reply

memset0

5 days ago, # | ☆ 0

Nice Article ... BTW Can we solve the problem with Binary lifting ?

→ Reply