



## Arpa's blog

### [Tutorial] Sack (dsu on tree)

By [Arpa](#), [history](#), 4 years ago, , 

Changes are available in history section.

Hi!

Most of the people know about dsu but what is the "dsu on tree"?

In Iran, we call this technique "Guni" (the word means "sack" in English), instead of "dsu on tree".

I will explain it and post ends with several problems in CF that can be solved by this technique.

## What is the dsu on tree?

With dsu on tree we can answer queries of this type:

How many vertices in the subtree of vertex  has some property in  $O(n \log n)$  time (for all of the queries)?

For example:

Given a tree, every vertex has color. Query is **how many vertices in subtree of vertex  are colored with color** ?

Let's see how we can solve this problem and similar problems.

First, we have to calculate the size of the subtree of every vertice. It can be done with simple dfs:

```
int sz[maxn];
void getsz(int v, int p){
    sz[v] = 1; // every vertex has itself in its subtree
    for(auto u : g[v])
        if(u != p){
            getsz(u, v);
            sz[v] += sz[u]; // add size of child u to its parent(v)
        }
}
```

Now we have the size of the subtree of vertex  in .

The naive method for solving that problem is this code(that works in  $O(N^2)$  time)

```
int cnt[maxn];
void add(int v, int p, int x){
    cnt[ col[v] ] += x;
    for(auto u : g[v])
        if(u != p)
            add(u, v, x)
}
void dfs(int v, int p){
    add(v, p, 1);
```

#### → Pay attention

**Before contest**  
[Codeforces Round #643 \(Div. 2\)](#)  
 06:36:28  
[Register now »](#)  
 \*has extra registration

Like 312 people like this. Be the first of your friends.

#### → Top rated

#	User	Rating
1	<a href="#">MiFaFaOvO</a>	3681
2	<a href="#">Um_nik</a>	3544
3	<a href="#">maroonrk</a>	3431
4	<a href="#">tourist</a>	3409
5	<a href="#">apiadu</a>	3397
6	<a href="#">300iq</a>	3317
7	<a href="#">ecnerwala</a>	3260
7	<a href="#">Benq</a>	3260
9	<a href="#">LHiC</a>	3229
10	<a href="#">TLE</a>	3223

[Countries](#) | [Cities](#) | [Organizations](#)

[View all →](#)

#### → Top contributors

#	User	Contrib.
1	<a href="#">Errichto</a>	195
2	<a href="#">antontrygubO_o</a>	193
3	<a href="#">vovuh</a>	178
4	<a href="#">pikmike</a>	172
5	<a href="#">tourist</a>	166
6	<a href="#">Um_nik</a>	165
7	<a href="#">Radewoosh</a>	164
7	<a href="#">McDic</a>	164
7	<a href="#">ko_osaga</a>	164
10	<a href="#">300iq</a>	155


[View all →](#)


#### → Find user

Handle:

Find

#### → Recent actions

[arpunagar](#) → [Find number of unique multiples.](#) 

[DishonoredRighteous](#) → [Codeforces Round #643 \(Div. 2\).](#) 

[Pie-Lie-Die](#) → [Help in priority\\_queue with comparator.](#) 

*//now cnt[c] is the number of vertices in subtree of vertex v that has color c. You can answer the queries easily.*

```
add(v, p, -1);
for(auto u : g[v])
    if(u != p)
        dfs(u, v);
}
```

Now, how to improve it? There are several styles of coding for this technique.

## 1. easy to code but $O(n \log^2 n)$ .

```
map<int, int> *cnt[maxn];
void dfs(int v, int p){
    int mx = -1, bigChild = -1;
    for(auto u : g[v])
        if(u != p){
            dfs(u, v);
            if(sz[u] > mx)
                mx = sz[u], bigChild = u;
        }
    if(bigChild != -1)
        cnt[v] = cnt[bigChild];
    else
        cnt[v] = new map<int, int> ();
    (*cnt[v])[ col[v] ] ++;
    for(auto u : g[v])
        if(u != p && u != bigChild){
            for(auto x : *cnt[u])
                (*cnt[v])[x.first] += x.second;
        }
}
```

*//now (\*cnt[v])[c] is the number of vertices in subtree of vertex v that has color c. You can answer the queries easily.*

```
}
```

## 2. easy to code and $O(n \log n)$ .

```
vector<int> *vec[maxn];
int cnt[maxn];
void dfs(int v, int p, bool keep){
    int mx = -1, bigChild = -1;
    for(auto u : g[v])
        if(u != p && sz[u] > mx)
            mx = sz[u], bigChild = u;
    for(auto u : g[v])
        if(u != p && u != bigChild)
            dfs(u, v, 0);
    if(bigChild != -1)
        dfs(bigChild, v, 1), vec[v] = vec[bigChild];
    else
        vec[v] = new vector<int> ();
    vec[v]->push_back(v);
    cnt[ col[v] ]++;
    for(auto u : g[v])
        if(u != p && u != bigChild)
            for(auto x : *vec[u]){
                cnt[ col[x] ]++;
                vec[v] -> push_back(x);
            }
}
```

*//now (\*cnt[v])[c] is the number of vertices in subtree of vertex v that has color c. You can answer the queries easily.*

*// note that in this step \*vec[v] contains all of the subtree of vertex v.*

```
if(keep == 0)
    for(auto u : *vec[v])
```

**retrograd** → [A \(possibly simpler\) algorithm for closest pair problem](#)

**lexiyvv** → [something wrong with organizations ranks](#)

**andreyv** → [Codeforces Round #142 Problem Analysis](#)

**gs15120** → [What can I do if I am stuck for 1 year?](#)

**Semal10** → [AlgoShots, Great Initiative!](#)

**Everule** → [Proving transitivity of comparator functions](#)

**MostafaUsama** → [wrong output format Unexpected end of file - token expected \(test case 76\)](#)

**sahil110310** → [How to practice?](#)

**hmehta** → [Topcoder SRM 786](#)

**ZhNV** → [Paz6op Codeforces Round #324 \(Div. 2\)](#)

**hmehta** → [Topcoder Dynamic Programming Webinar and Practice Contest](#)

**winter\_summer** → [Regarding timing clash of Educational Codeforces Round 87 \(Rated for Div. 2\) with Google kick start Round C](#)

**Halit** → [USACO RESOURCES!](#)

**kartik8800** → [CSES Range Queries section editorial](#)

**ProgSlacking** → [Codeforces Round #641 Editorial](#)

**OsmanliTokadi** → [how to use mine problems which prepared in polygon in GYM?](#)

**beginner\_610** → [Stuck in a problem.](#)

**pikmike** → [Educational Codeforces Round 86 \[Rated for Div. 2\]](#)

**prashant.2k03** → [input:input.txt and output:output.txt](#)

**OsmanliTokadi** → [Preparing for olymiads Contest-2](#)

**vovuh** → [Codeforces Round #642 \(Div. 3\) Editorial](#)

**Halit** → [Preparing for Olympiads Contest#2](#)

[Detailed →](#)

```

        cnt[ col[u] ]--;
    }

```

### 3. heavy-light decomposition style $O(n \log n)$ .

```

int cnt[maxn];
bool big[maxn];
void add(int v, int p, int x){
    cnt[ col[v] ] += x;
    for(auto u: g[v])
        if(u != p && !big[u])
            add(u, v, x)
}
void dfs(int v, int p, bool keep){
    int mx = -1, bigChild = -1;
    for(auto u : g[v])
        if(u != p && sz[u] > mx)
            mx = sz[u], bigChild = u;
    for(auto u : g[v])
        if(u != p && u != bigChild)
            dfs(u, v, 0); // run a dfs on small childs and clear them from cnt
    if(bigChild != -1)
        dfs(bigChild, v, 1), big[bigChild] = 1; // bigChild marked as big and
not cleared from cnt
    add(v, p, 1);
    //now cnt[c] is the number of vertices in subtree of vertex v that has
color c. You can answer the queries easily.
    if(bigChild != -1)
        big[bigChild] = 0;
    if(keep == 0)
        add(v, p, -1);
}

```

### 4. My invented style $O(n \log n)$ .

This implementation for "Dsu on tree" technique is new and invented by me. This implementation is easier to code than others. Let  $st[v]$  dfs starting time of vertex  $v$ ,  $ft[v]$  be it's finishing time and  $ver[time]$  is the vertex which it's starting time is equal to  $time$ .

```

int cnt[maxn];
void dfs(int v, int p, bool keep){
    int mx = -1, bigChild = -1;
    for(auto u : g[v])
        if(u != p && sz[u] > mx)
            mx = sz[u], bigChild = u;
    for(auto u : g[v])
        if(u != p && u != bigChild)
            dfs(u, v, 0); // run a dfs on small childs and clear them from cnt
    if(bigChild != -1)
        dfs(bigChild, v, 1); // bigChild marked as big and not cleared from
cnt
    for(auto u : g[v])
        if(u != p && u != bigChild)
            for(int p = st[u]; p < ft[u]; p++)
                cnt[ col[ ver[p] ] ]++;
    cnt[ col[v] ]++;
    //now cnt[c] is the number of vertices in subtree of vertex v that has
color c. You can answer the queries easily.
    if(keep == 0)
        for(int p = st[v]; p < ft[v]; p++)
            cnt[ col[ ver[p] ] ]--;
}

```

But why it is  $O(n \log n)$ ? You know that why dsu has  $O(q \log n)$  time (for  $q$  queries); the code uses the same method. Merge smaller to greater.

If you have heard `heavy-light decomposition` you will see that function `add` will go light edges only, because of this, code works in  $O(n \log n)$  time.

Any problems of this type can be solved with same `dfs` function and just differs in `add` function.

Hmmm, this is what you want, problems that can be solved with this technique:

(List is sorted by difficulty and my code for each problem is given, my codes has `heavy-light` style)

600E - Lomsat gelral : `heavy-light decomposition` style : [Link](#), easy style : [Link](#). I think this is the easiest problem of this technique in CF and it's good to start coding with this problem.

570D - Tree Requests : 17961189 Thanks to [Soradorasora](#); this problem is also good for start coding.

Sgu507 (SGU is unavailable, read the problem statements [here](#)) This problem is also good for the start.

HackerEarth, The Grass Type This problem is also good for start (See [bhishma](#)'s comment below).

246E - Blood Cousins Return : 15409328

208E - Blood Cousins : 16897324

IOI 2011, Race (See [SaSaSaS](#)'s comment below).

291E - Tree-String Problem : See [bhargav104](#)'s comment below.

1009F - Dominant Indices : 40332812 Arpa-Style. Thanks to [baymaxx](#).

343D - Water Tree : 15063078 Note that problem is not easy and my code doesn't use this technique (dsu on tree), but [AmirAz](#)'s solution to this problem uses this technique : 14904379.

375D - Tree and Queries : 15449102 Again note that problem is not easy :)).

716E - Digit Tree : 20776957 A hard problem. Also can be solved with centroid decomposition.

741D - Arpa's letter-marked tree and Mehrdad's Dokhtar-kosh paths : 22796438 A hard problem. You must be very familiar with Dsu on tree to solve it.

For Persian users, there is another problem in Shaazzz contest round #4 (season 2016-2017) problem 3 that is a very hard problem with this technique.

If you have another problem with this tag, give me to complete the list :)).

And after all, special thanks from [amd](#) who taught me this technique.


 Announcement of EIO 18-19 Suvelaager (Kompik)

 Tutorial of Preparation for ICPC (Series 6 - Advanced Data Structures)

 dsu on tree, sack, guni

 +69 

 [Arpa](#)

 4 years ago

 144



Comments (144)

[Write comment?](#)



4 years ago, # |

 +4 

[A2OJ's DSU Section](#) has quite a few tree DSU problems.

Thank you for this post, it explains the theory well and is very easy to read

maximaxi

Thank you for this post, it explains the theory well and is very easy to read.  
→ [Reply](#).



gotosleep

4 years ago, # ^ |

▲ -6 ▼

بدك تضل تتمنيك عكل بوستات الخرا ؟

→ [Reply](#)



shubhankarsharma00

11 months ago, # ^ |

← Rev. 3 ▲ 0 ▼

unlike this comment

→ [Reply](#)



himanshujaju

4 years ago, # |

▲ 0 ▼

What does the variable "keep" denote ?

→ [Reply](#)



NibNalin

4 years ago, # ^ |

← Rev. 4 ▲ +3 ▼

The way I understand HLD here is basically if a child is the big child, we don't want to recompute answer for it to reduce computation. So we just store the answer for it in the `cnt` array already so that it's parent doesn't need to re-dfs this subtree. `keep` denotes whether or not this child is that big child. Please correct me if I'm wrong. :)

→ [Reply](#)

4 years ago, # ^ |

← Rev. 3 ▲ 0 ▼

Look at last two lines:

```
if(keep == 0)
    add(v, p, -1);
```



Arpa

It means that if `keep == false` after `dfs` clear `v`'s subtree information from `cnt`. And if `keep == true`, don't clear `v`'s subtree information from `cnt`. In other word if `keep == true` after calling `dfs`, for each `u` from subtree of vertice `v`, `col[u]` is in `cnt` ( `cnt[ col[u] ]++` ).

And NibNalin is right. `keep` is `true` if and only if `v` is biggest child of it's parent.

→ [Reply](#)

4 years ago, # ^ |

▲ 0 ▼

Hi Arpa, thanks a ton for this awesome post. I have really learnt lot from it.



ka89

I do have one question though. What is the advantage of having `keep=false`? If that part is kept as it is, without clearing, doesn't the computation become faster? Can you please help clearing this doubt?

→ [Reply](#)

4 years ago, # ^ |

▲ 0 ▼

Hi, Thanks.



Arpa

Consider vertex `v` has two children, `q` and `p`. If you call `dfs` for both of them with `keep = true`, they will mixed up their information, and queries will be incorrectly answered.

→ [Reply](#)

4 years ago, # ^ |

▲ 0 ▼

oh ok Got it now Thanks



ka89

ON. OK. GOT IT NOW. THANKS.

→ [Reply](#)



Lance\_HAOH

3 years ago, # ^ | ← Rev. 2 ▲ 0 ▼

I guess this method is only viable if DP cannot be used? (i.e. Too many states to memoize)

→ [Reply](#)

4 years ago, # ^ |

▲ +34 ▼

Observations to understand the complexity:

1. The `dfs` function visits each node exactly once.
2. The problem might seem with the add function. You might think that it is making the algorithm  $n^2$ . Note that in the `add` function, we only go down from a vertex to its children if the edge connecting the vertex to the child is a light edge.



flatline

You can think of it in this way, each vertex `v` will be visited by a call to the `add` function for any ancestor of `v` that is connected to a light edge. Since there are at most  $\log(n)$  light edges going up from any vertex to the root, each vertex will be visited at most  $\log(n)$  times.

So the algorithm is: Say you are at a vertex `v`, first you find the bigchild, then you run dfs on small childs, passing the value of keep as `0`. Why? So they are cleared from `cnt`. Then you run a dfs on bigchild, and you do not clear it from `cnt`. Now, `cnt` stores the results for all vertices in the subtree of `bigchild` (since we cleared `cnt` for small childs and didn't do so for the bigchild), so we call the `add` function to "add" the information of children of current vertex that are connected to it via a light edge. Now we are ready to compute the answer

→ [Reply](#)

3 years ago, # ^ |

← Rev. 3 ▲ +3 ▼



gogateiiiit

As you said "add" function goes down using only light edges, Don't these two lines `if(bigChild != -1)` `big[bigChild] = 0;` of heavy light decomposition implementation would affect it as if you call "add" after all dfs are done and returned to the root then we only have one heavy edge marked that of root itself others are zero so as "add" goes below it traverses whole tree. Help me here.

→ [Reply](#)



gogateiiiit

3 years ago, # ^ |

▲ 0 ▼

Got it.

→ [Reply](#)



gogateiiiit

3 years ago, # ^ | ← Rev. 2 ▲ +3 ▼

For those who had same doubt as I had: First let's understand why it is wrong to remove it, consider you are at particular node (let it be called A) in recursion, above line is not there, you have 3 children one of them is big child (3rd) while others are normal so you traversed inside 1st and came back to A then you traversed inside 2nd child if you do not have above line then while going inside this children you would have all answer for big children of 1st child which would mess your answer. Now let's understand why complexity is  $O(n \log(n))$ :

Note 1: To calculate complexity you need

Note 1: To calculate complexity you need to measure how many times add function visits the every node.

Note 2: For first add called at a node: A node will visited by add only through its ancestors which are connected by light edges so n nodes  $\log(n)$  light edges above it this gives us  $O(n \log(n))$

Note 3: For second add called at a node: Now somebody may protest that that after above mentioned line(`big[bigChild]=0`) we are unmarking heavy edge and also calling add after that which may mess up complexity as it travels every node below it which is  $O(n)$  but `keep==0` condition ensures that for each node there atmost  $\log(n)$  nodes above in ancestor which have `keep=0` function is called. which again gives  $O(n \log(n))$ .

Giving us finally  $O(n \log(n))$  complexity. Follow this link to understand heavy light decomposition's properties: <https://blog.anudeep2011.com/heavy-light-decomposition/>

→ [Reply](#)

4 years ago, # |

▲ 0 ▼

In second easy with  $O(n \lg n)$

Why if(`keep==false`) we delete only vertex from main vector

```
for(auto u : *vec[v])
    cnt[ col[u] ]--;
```



the\_art\_of\_war

but we don't delete vertex from cnt which we changed here:

```
for(auto u : g[v])
    if(u != p && u != bigChild){
        for(auto u : *vec[u])
            cnt[ col[u] ]++;
    }
```

→ [Reply](#)

4 years ago, # ^ |

← Rev. 2 ▲ 0 ▼

There was a mistake in writing. I'm sorry.

Thanks for reporting this problem.

code should be:

```
if(u != p && u != bigChild){
    for(auto x : *vec[u])
        cnt[ col[x] ]++;
}
```



Arpa

Instead of:

```
if(u != p && u != bigChild){
    for(auto u : *vec[u])
        cnt[ col[u] ]++;
}
```

I have edited that.

→ [Reply](#)

4 years ago, # |

▲ 0 ▼

can someone tell me how 208F is solved with this technique? thanks a lot



SProf

can someone tell me how 2002 is solved with this technique: thanks a lot.  
→ [Reply](#).

4 years ago, # ^ |

▲ 0 ▼

You need to compute for each pair  $(v, p)$  the  $p$ -th cousin of  $v$ . That is equivalent to finding the number of  $p$ -th descendants of the  $p$ -th ancestor of  $v - 1$ .



flatline

So for each query, replace  $(v, p)$  with  $(p\_th\_ancestor\_of\_v, p)$ . Now you need to store in  $cnt$  the number of nodes at a certain depth. In other words,  $cnt[x]$  should be equal to number of nodes at depth  $x$  in the current subtree.

Code for Reference:

<http://codeforces.com/contest/208/submission/17513471>

→ [Reply](#).

4 years ago, # ^ |

▲ 0 ▼



shavidze

can't understand why for every vertex  $v$  we  $ans[depth[v]]$  increase by 1 when we call add function, why must we do it? or why it must be  $ans[depth[v]]$  when  $depth[v]$  means distance from root to  $v$ ?

→ [Reply](#).

4 years ago, # ^ |

← Rev. 2

▲ 0 ▼

$ans[h]$  is equal to number of vertices with height  $h$ , (with distance  $h$  from root).

Let  $par$ ,  $p$ 'th ancestor of  $v$ , the answer to query is:



Arpa

Consider only subtree of vertex  $par$ , print  $ans[height[v]] - 1$ .

So with method above we can process all of the queries.

See my code for better understanding.

→ [Reply](#).



shavidze

4 years ago, # ^ |

▲ +5 ▼

thanks everyone, now i understand.

→ [Reply](#).

4 years ago, # ^ |

▲ +10 ▼



Sora233

If i haven't read this article, i wouldn't get ac on [this](#) problem. It is another problem which can be solved easily by dsu.

[here](#) is my code in HLD-style.

Thanks!

→ [Reply](#).



Arpa

4 years ago, # ^ |

▲ 0 ▼

Thanks! Added to list ;)

→ [Reply](#).



Batman

4 years ago, # ^ |

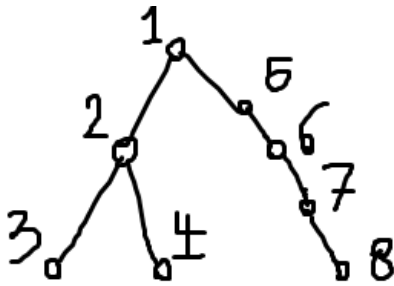
← Rev. 3

▲ +12 ▼

I can't understand why the second code is correct...

Consider this example:





We wanna calculate the cnt for Vertex 8. These are the steps:

Going to Vertex 1

Going to Vertex 2 by keep=0

Going to Vertex 3 by keep=0, Vec[3]={3}

Going to Vertex 4 by keep=1, Vec[4]={4}, Cnt[color[4]]=1

Going back to Vertex 2, Vec[2]={2,4}, Cnt[color[4]]=0, Cnt[color[3]]=1

And then when we go to Vertices 5,6,7,8 still Cnt[color[3]]=1.

Also sorry if I did the steps wrong...

**UPD** Thank you for editing the blog. My problem fixed.

→ [Reply](#)

4 years ago, # |

← Rev. 4 ▲ 0 ▼

Great post. If you explained the idea before showing the code, it would be better to understand. Also commenting the variables meaning in the code would be of great help.

It would be good to mention that most solutions will answer the queries offline, which may be a problem sometime (maybe someone didn't notice this lol).



**gabrielsimoes**

Also, it would be nice to post hints about the solutions.

Proving explicitly why it is  $n \log n$  would be good too (ie. as each node's subtree set gets merged into one set of size equal or greater, and the base set has size 1 and the last set has size  $n$ , then we take  $\log n$  steps to go from 1 to  $n$ . Summarizing, each node gets merged  $\log n$  times, so the total complexity is  $O(n \log n)$ ).

Here's my solution to 343D, maybe it will be of help to someone: [18958875](#). A lot easier to code than the one in the tutorial.

→ [Reply](#)

4 years ago, # ^ |

▲ +2 ▼

Thanks for your suggestions first !



**Arpa**

I proved that it is  $O(n \log n)$  : *You know that why dsu has  $O(q \log n)$  time (for  $q$  queries); the code uses same method. Merge smaller to greater.*

And about your code ([18958875](#)), anyone has a different opinion !

→ [Reply](#)



**gabrielsimoes**

4 years ago, # ^ |

← Rev. 2 ▲ 0 ▼

Thanks for the reply!

Yeah, you did prove. People who remember DSU's proof will most likely understand. I stated a more extensive proof would be better thinking about people who don't exactly know the proof. Don't take me wrong, but they may get a little confused reading this proof.

I mentioned my code exactly because everyone has a

I mentioned my code exactly because everyone has a different opinion,. Maybe it'll help a later reader, that's all.

→ [Reply](#)

4 years ago, # ^ |

▲ 0 ▼

Sorry this might be a stupid question to bring up, but why is the complexity of the heavy-light decomposition style one in  $O(n \log n)$ ?

In the case where each node has at most two children: Denote the root node of the tree as  $u$ , which is of size  $s$ . The child of  $u$  connected to the lighter edge is of size at most  $\frac{s}{2}$ . So the total number of nodes on which we run the "add" function would be at most  $\frac{s}{2} + \frac{s}{4} + \dots = s$ . So I don't understand where the  $\log(n)$  factor comes from.



pivorics

The online tutorial for HLD says a new chain is built when we arrive at a child node via a lighter edge, where each chain is stored as a segment tree, and so I can see there is indeed a  $O(\log n)$  factor involved.

Regardless can you perhaps elaborate a little bit more on the time complexity of the dsu structure? Thank you!

→ [Reply](#)

4 years ago, # ^ |

▲ 0 ▼

Hi !

*The online tutorial for HLD says a new chain is built when we arrive at a child node via a lighter edge, where each chain is stored as a segment tree, and so I can see there is indeed a  $O(\log n)$  factor involved.*

As you know, if you use **segment tree** in **heavy-light decomposition**, each query time complexity will be  $O(\log^2(n))$ . Because in each query you will go  $O(\log(n))$  chains and in each chain it will spend  $O(\log(n))$  time.



Arpa

Now, I will proof that "heavy-light decomposition style implementation" of "dsu on tree" is  $O(n \cdot \log(n))$ :

Consider a complete binary tree with  $n$  vertices. In dfs function you will run another dfs function in child ( $T(n/2) * 2$ ) and you will call **add** function and it will spend  $O(n/2)$  time. So,  
$$T(n) = n/2 + 2 * T(n/2) = O(n \cdot \log(n))$$

→ [Reply](#)

3 years ago, # ^ |

▲ 0 ▼

*You know that why dsu has  $O(q \log n)$  time (for  $q$  queries); the code uses same method. Merge smaller to greater.*



Emphi

Pardon me , but I don't follow. Which dsu are you talking about? The one with inverse-Ackermann function?

→ [Reply](#)



3 years ago, # ^ |

▲ 0 ▼

No. Dsu with size compare. Like this :

```
int find(int x) {
```

Arpa

```
int find(int x){
    return par[x] == x ? x : find(par[x]);
}

void merge(int v, int u){
    v = find(v), u = find(u);
    if(v == u) return ;
    if(size[v] < size[u]) swap(v, u);
    par[u] = v;
    size[v] += size[u];
}
```

→ [Reply](#)

4 years ago, # |

← Rev. 2 ▲ 0 ▼



rcg\_

In easy to code but  $O(n \log^2 2)$ , I can't understand why do we store the size of subtrees of vertices in array `sz` and use it as the criteria for choosing the big child, I think we should store in the array "`sz`" the number of distinct colors in the subtree of any node `v`, because that is what we actually iterate on when transferring the map from `v` to `u`, why is this wrong?

→ [Reply](#)

4 years ago, # ^ |

▲ 0 ▼



Arpa

Hi !

It isn't wrong! Both of your method and mine have the same worst case. But your average is better.

→ [Reply](#)



hellojarvis

14 months ago, # ^ |

▲ 0 ▼

Is child's subtree-size working as a heuristic for the number-of-distinct-colors in that child?

→ [Reply](#)

4 years ago, # |

← Rev. 2 ▲ 0 ▼



algo.experiments

Ahh, thanks gabrielsimoes, for anyone struggling to understand:  $n \cdot \log^2 n$  is about answering queries OFFLINE right during the dfs. After the dfs has finished the `cnt[v]` will no longer be a valid map for vertices that were chosen as `bigChild`.

→ [Reply](#)

4 years ago, # |

▲ +8 ▼



bhargav104

<http://codeforces.com/problemset/problem/291/E> 291E - Tree-String Problem

Arpa This problem can also be done by dsu on trees. Calculate hash value for each suffix value of target string. Then for each suffix of an edge if it is a valid prefix of the target string we would just need the frequency of the hash value of the remaining suffix of the target string in its subtree which can be maintained by this technique. The case when the entire string occurs in an edge can be dealt with separately.

→ [Reply](#)

4 years ago, # ^ |

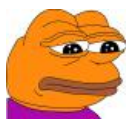
▲ +5 ▼



Arpa

Thanks added to list, but it can be solved very easier : 19827525, just use KMP.

→ [Reply](#)



Dalgerok

3 years ago, # ^ |

▲ 0 ▼

Just use hashes :)

<http://codeforces.com/contest/291/submission/29431526>

→ [Reply](#)



Mooncrater

3 months ago, # ^ |

▲ 0 ▼

Hello Arpa,

The technique which you've used in this question, for the KMP solution, in which you've created the `next` 2D array, is it called something? Is it some recognized idea or did you just

called something: is it some recognized idea or did you just came with it on the go?

Basically, you removed the inner while loop of KMP for matching strings. Instead of jumping multiple times to check whether the current characters match, you created a DP, using which we can do that in a single jump. That saves a lot of time, as we would have to do that for each string in the tree, while matching.

So, is this some already defined method?

→ [Reply](#)

3 months ago, # ^ |

▲ +1 ▼



Arpa

Hello.

Do you know [automata](#)? I'm building automata from the string.

→ [Reply](#)

3 months ago, # ^ |

▲ 0 ▼



Mooncrater

I do know automata. But damn. That's some beautiful automata you got there. GG.

→ [Reply](#)



bkt1love

2 years ago, # ^ |

▲ 0 ▼

Please send me a code of the solution with this technique)

→ [Reply](#)

3 months ago, # ^ |

← Rev. 3

▲ 0 ▼



Mooncrater

[Here is bhargav104's solution](#)

EDIT: Sorry. It doesn't use DSU on trees. Rather, it uses KMP.

→ [Reply](#)

3 years ago, # |

← Rev. 3

▲ +13 ▼



sengxian

Actually, in China, we call this method as "Heuristic Merge" which always merge the smaller to the bigger. Not hard to understand each vertex will be visited in  $O(\log n)$  times because when we visited a vertex, then the size of tree which the vertex is in doubled.

→ [Reply](#)

3 years ago, # |

▲ +1 ▼

Hey Arpa,



abhigarg1796

In your my invented style I'm unable to understand that why in third loop are you not checking for u not being parent of v. Why are you only checking for just u not being the big child.

Thanks in Advance

→ [Reply](#)

3 years ago, # ^ |

▲ 0 ▼



Arpa

Sorry, fixed. It's because I've copied my code from [741D - Arpa's letter-marked tree](#) and [Mehrdad's Dokhtar-kosh paths](#), input for this problem was a rooted tree.

→ [Reply](#)

3 years ago, # ^ |

▲ +2 ▼



abhigarg1796

Thanks a lot,

Also, I think there is one more mistake. You never added

Also, I think there is one more mistake. You never added `col[v]` to the array. Am I missing something. Thanks in advance.

→ [Reply](#)



Arpa

3 years ago, # ^ |

▲ 0 ▼

You are right, I'm very thankful to you. I was careless while coping the code from polygon.

→ [Reply](#)



bhishma

3 years ago, # |

▲ 0 ▼

In the easy to code  $O(n \log n)$  method `vec[v]` stores all the vertices in the subtree rooted at `v`. How will this fit into memory if we are not deleting the `vec` of child after merging it with the parent

→ [Reply](#)

3 years ago, # ^ |

▲ +1 ▼



Arpa

Used memory is always less than or equal to time complexity, so when time complexity is  $O(n \cdot \log n)$ , used memory is less than or equal to  $O(n \cdot \log n)$ . In this case, used memory is  $O(n \cdot \log n)$ . Although if you delete useless `vec`'s the memory become  $O(n)$ .

→ [Reply](#)



bhishma

3 years ago, # ^ |

▲ 0 ▼

Thanks for the reply. I think this problem can also be solved using your approach. ([The Grass Type HackerEarth](#))

→ [Reply](#)

3 years ago, # ^ |

← Rev. 3

▲ +1 ▼

I'll add this problem to the post if I find it related, I'm thankful anyway.



Arpa

**Edit :** Note that this is not **my** approach, but I'm the first man who publishes a tutorial about this (not sure), Sack has been used in INOI, IOI and ACM several times, so it isn't a new thing, invented by me.

**Edit :** Added.

→ [Reply](#)



SaYami

3 years ago, # ^ |

▲ 0 ▼

Can you mention problems from the IOI that are solved with sack?

→ [Reply](#)

3 years ago, Rev. 2 |

▲ +6 ▼



Arpa

I'll add one of them tonight.

**Edit :** Added.

→ [Reply](#)



SaYami

3 years ago, # ^ |

Wow, I didn't think of solving it with sack. Thx

→ [Reply](#)



Agassaa

3 years ago, # |

▲ 0 ▼

Hi **Arpa**, I can not understand, why is this approach called **dsu** on tree? This approach has a nice trick to reduce complexity by saving data about "big child". I can't see any special similarity with general dsu approach. In general dsu

problems, we merge 2 subset into 1 set by linked list approach. But in your

problems, we merge 2 subset into 1 set by linked list approach. But, in your tutorial there is no "merge" function. Am I missing something?

Also I see that, in your 600E's solution [14554536](#) you used merge function. I can't understand, could you please explain that code?

→ [Reply](#).

3 years ago, # ^ |

▲ 0 ▼



Arpa

In fact we are merging information of small children with big child. Think more.

In that code, *mrg* function merges information in *u* into *v*.

→ [Reply](#).

3 years ago, # |

▲ +1 ▼

Hi **Arpa**! Thanks for making this tutorial.



beAwesome

I just want to make sure my understanding is correct: this merging smaller maps into larger ones takes logarithmic time because when a vertex is merged, the new map it is in at least twice its size. Hence, merging can only happen  $\log(n)$  times for each of the  $n$  vertices, leading to a total runtime of  $O(n \log n)$ ?

Thanks!

→ [Reply](#).



Arpa

3 years ago, # ^ |

▲ 0 ▼

Yes, but note that if you use map, it's  $O(n \cdot \log^2 n)$ .

→ [Reply](#).



beAwesome

3 years ago, # ^ |

▲ 0 ▼

If you use unordered\_map, does it become  $O(n \cdot \log n)$ , then?

→ [Reply](#).



Arpa

3 years ago, # ^ |

▲ 0 ▼

Unordered\_map is theoretically  $O(n)$  per query. But you can suppose that it's  $O(1)$  per query in code.

→ [Reply](#).



surajghosh

3 years ago, # |

▲ 0 ▼

This [758E](#). Read [this](#) comment on how to use it.

→ [Reply](#).



HUECTRUM1

3 years ago, # |

▲ 0 ▼

Why do we need to iterate through the children of *v* after `add(v, p, -1)` in the naive approach?

→ [Reply](#).



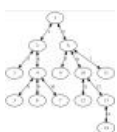
satyaki3794

3 years ago, # ^ |

▲ 0 ▼

dfs() solves the problem for all the nodes, not just one. So, after you've gotten the answer for *v*, it will calculate the answer for its children.

→ [Reply](#).



vatsal

3 years ago, # |

▲ +8 ▼

[101 Hack 47 Summing Tree](#) was solved using this technique by [satyaki3794](#) [Submission](#)

→ [Reply](#).



lukecavabarrett

3 years ago, # |

▲ +3 ▼

also [778C - Peterson Polyglot](#) is solvable with a similar technique: is that DSU on tree?

→ [Reply](#).

→ [Reply](#).



**radoslav11**

3 years ago, # ^ |

▲ +3 ▼

yes

→ [Reply](#).



**\_W\_**

3 years ago, # |

▲ +6 ▼

Can anyone give me a link to "Shaazzz contest round #4 (season 2016-2017) problem 3" or tell me where can I find it? Thanks.

→ [Reply](#).



**Arpa**

3 years ago, # ^ |

▲ +5 ▼

It's a Persian contest.

→ [Reply](#).



**\_W\_**

3 years ago, # ^ |

▲ +6 ▼

Can you tell me where can I find it? I searched for it just now but didn't get it.

→ [Reply](#).



**Arpa**

3 years ago, # ^ |

▲ +6 ▼

[Link.](#)

→ [Reply](#).



**\_W\_**

3 years ago, # ^ |

▲ +11 ▼

Thank you!

→ [Reply](#).



**tak\_fate**

3 years ago, # |

▲ 0 ▼

I can AC easily Problem 375D by the 3th way ,but WA by the 4th way.... WA on the test 4.. why..

→ [Reply](#).



**gabrielsimoes**

3 years ago, # |

▲ 0 ▼

APIO 2016 Fireworks uses this, but is a much harder problem.

→ [Reply](#).



**maxorand**

3 years ago, # |

▲ 0 ▼

**Arpa**, in the **Easy to code but  $O(n \log^2 n)$**  section code you have written a commented line that is : `//now (*cnt)[c] is the number of vertices in subtree of vertex v that has color c. You can answer the queries easily.` . But I think it would be `//now (*cnt[v])[c] is the number of vertices in subtree of vertex v that has color c. You can answer the queries easily.` . Will `(*cnt)[c]` changed with `(*cnt[v])[c]` ?

→ [Reply](#).



**Arpa**

3 years ago, # ^ |

▲ 0 ▼

Hi. Thanks. Edited.

→ [Reply](#).



**Trath**

3 years ago, # |

▲ 0 ▼

You can solve [APIO 2012 Dispatching](#) with this technique too.

→ [Reply](#).



3 years ago, # |

▲ 0 ▼

IQI 2011 — Race What is the idea of DSU on tree for this problem? I know of a

mochow

for 2011 — have what is the idea of DCC on tree for this problem? I know of a solution based on Centroid Decomposition.

→ [Reply](#).



KwanghyunOn

10 months ago, # ^ |

▲ 0 ▼

keep map<distance from root, depth> for all subtrees. By merging them from smaller to larger, you can get  $O(n \log^2 n)$  solution.

→ [Reply](#).



Zeus726

2 years ago, # |

← Rev. 2

▲ +8 ▼

914E - Palindromes in a Tree can solve with sack too, Arpa :)

ps: for this problem centroid decompose works too... :)

→ [Reply](#).



pk842

2 years ago, # |

▲ 0 ▼

can anyone please explain how to solve 716E using this technique?

→ [Reply](#).

2 years ago, # |

▲ +5 ▼

In the contest 600, no one can view other's submissions except his own.

That's why no can see your submissions for 600E - Lomsat gelral except you.

So, please give alternating link of your solutions for the first problem in the list, 600E - Lomsat gelral

→ [Reply](#).



shahidul\_brur

2 years ago, # ^ |

▲ 0 ▼

Hi.

Thanks for your feedback. Here it is: [Link](#).

→ [Reply](#).



Arpa



shahidul\_brur

2 years ago, # ^ |

▲ 0 ▼

Thank you !

→ [Reply](#).

2 years ago, # |

← Rev. 2

▲ 0 ▼

Another problem which can be solved by this technique: [Coloring Tree](#)  
Its easier than 600E - Lomsat gelral.

One more : 932F - Escape Through Leaf

→ [Reply](#).



Vicennial

2 years ago, # |

← Rev. 2

▲ 0 ▼

Hi, I would like to ask you about the code in heavy-light decomposition style. Why the bigChild is cleared before clearing the subtree at the end of the code? From my perspective, in add function bigChild will be visited and the array "big" doesn't make sense. Can you explain it in detail? Thanks a lot.

→ [Reply](#).



IzoI\_hhn

2 years ago, # |

▲ -7 ▼

No good explanation! Only code! Worst tutorial.

→ [Reply](#).



kirito\_



satvik007

2 years ago, # ^ |

▲ 0 ▼

If you have a doubt why not ask in the comments rather than whining about how bad the tutorial is.

→ [Reply](#).



2 years ago, # |

▲ +5 ▼

EXPIRED

Nice tutorial!

and also happy Nowruz:)

→ [Reply](#)

M\_H\_H\_7



Ehsan\_sShuvo

2 years ago, # ^ |

← Rev. 2 ▲ 0 ▼

Could you please explain 2no. style ? [Upd : Got it ]

→ [Reply](#)



vivace\_jr

23 months ago, # ^ |

← Rev. 3 ▲ 0 ▼

In 2nd style Arpa can you please explain?..when we do  $\text{cnt}[u] = \text{cnt}[\text{heavy child}]$  does this happen in  $O(1)$ ?

→ [Reply](#)



vivace\_jr

23 months ago, # ^ |

← Rev. 2 ▲ 0 ▼

if yes -> how? else what the benefit of doing this?  
upd[got it]

→ [Reply](#)



Ehsan\_sShuvo

2 years ago, # |

▲ 0 ▼

If i do my code using 2nd approach , shouldn't my problem be static ? And for query operation , we will do offline query , won't that ?

→ [Reply](#)



Azurey

2 years ago, # |

▲ +8 ▼

For those who still confused about the time complexity, I found [this](#) explanation by [radoslav11](#) helps a lot.

→ [Reply](#)



andiso

2 years ago, # |

▲ 0 ▼

What is  $\text{col}[v]$ ?

→ [Reply](#)



Azurey

2 years ago, # ^ |

▲ 0 ▼

Color of the v-th vertex

→ [Reply](#)

2 years ago, # |

▲ 0 ▼

I loved the way you used the Euler tour sequence to iterate through the subtrees. My 2 cents is a way to use C++ to implement this nicely. The main idea is to use a structure, that keeps pointers instead of  $\text{st}[v]$  and  $\text{ft}[v]$  , and to give it a `begin()` and `end()` member function, so that you can iterate through it with

```
for(int u : T[i])  
    cout << u << " is in the subtree of " << i << "\n";
```

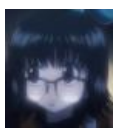
To actually solve the problem, maybe in a Heavy-Light style, I kept also for each vertices a pointer to the leaf of its heavy-path, so that I could only change the answer in the leaf.

My full implementation of lomsat gelral can be found [here](#)

→ [Reply](#)



\_Na2Th



yaksha

23 months ago, # |

▲ 0 ▼

Hi , Arpa, I used this technique to solve [600E - Lomsat gelral](#), its a very neat technique.. but wont this get a MLE? I am getting a MLE with the  $O(n \log^2 n)$

method I saw your solution and I see its the same as mine but mine gets

method, ... I saw your solution and I see its the same as mine, but mine gets MLE.. My solution :- [39325497](#)

→ [Reply](#)

23 months ago, # ^ |

← Rev. 3 ▲ +5 ▼



dmkozyrev

Solutions for Educational Codeforces Round 2 is private, nobody can't see your code. Perhaps you do not use pointers, dynamic memory allocation, and memory cleanup. In this technique it is forbidden to copy huge containers.

I solved this problem with Euler tour on tree and Mo algorithm in  $O(n \cdot \sqrt{n})$  time and  $O(n)$  memory. [Code](#).

→ [Reply](#)



hackerwizard

23 months ago, # |

▲ 0 ▼

Great blog but I am not able to understand the logic behind the  $O(n \log n)$  solution it would be a great help if anyone can explain it.

→ [Reply](#)

22 months ago, # |

▲ 0 ▼

I have a question for style 3 i.e HLD style.

I'm not so sure what's happening in there, I have 2 assumptions. Both of them are wrong, so it would great if someone could point out the mistake and explain what's right.

We are in root.

1. We first go down the light edges of root and when we finish with them, we clear the cnt array, so there is absolutely nothing left in it and then we proceed to the heavy child of the root and then we just update cnt array. Now we go back to the light edges of the root and (here's the problem) as the cnt array only contains information for the heavy child of the root, we must go through EVERY vertex in subtrees of light children of the root. If we don't go to the heavy children in the subtrees (as it proposes in tutorial?), then the answer is wrong, as we didn't count them (remember that we cleared the cnt array).
2. We first go down the light edges of the root, but this time, for every heavy child, we keep the information. But then as we proceed to the heavy child of the root, the array cnt won't be empty and the answer for heavy child will be incorrect.

→ [Reply](#)



The\_Wolfpack

22 months ago, # ^ |

▲ +9 ▼



Arpa

Consider you entered vertex  $v$ , let it's heavy child  $h$ . First, go to all child of  $v$  except  $h$  and calculate the answer for them and each time clear the cnt array. Then go to  $h$  and calculate the answer and don't clear the cnt array. Then traverse all of the subtree of  $v$  except subtree of  $h$ , and add them to cnt.

→ [Reply](#)



The\_Wolfpack

22 months ago, # ^ |

▲ +13 ▼

I got it, thanks for the quick reply!

→ [Reply](#)

22 months ago, # |

▲ +18 ▼

Another problem can be solved with this technique. [Arpa](#) please add this one in the list.

<http://codeforces.com/contest/1009/problem/F> (Dominant Indices)

→ [Reply](#)



Tanmoy\_Datta



22 months ago, # |

▲ 0 ▼

Can someone explain why this solution [40510312](#) is getting TLE on problem

deepdark-  
coldstars-light

Can someone explain why this solution [70010012](#) is getting TLE on problem 600E - Lomsat gelral. I'm using style 4.

→ [Reply](#).



Irvideckis

22 months ago, # ^ |

▲ +8 ▼

Can't view your submission

→ [Reply](#).



deepdark-  
coldstars-light

22 months ago, # ^ |

▲ +5 ▼

Firstly, thank you for reaching out to help :). I got my mistake.

→ [Reply](#).



ChandyShot

22 months ago, # |

▲ 0 ▼

I'm continuously getting TLE in test case 30 in [570 : D : Tree Request](#) (The second one given in the blog practice problem) while implementing through DSU similar to what is given here. Submission Id : 40780911 , can somebody make a look over this and provide hint to optimize it.

→ [Reply](#).



m0nk3ydluffy

16 months ago, # ^ |

▲ 0 ▼

My solution passed the time limit after replacing `endl` with `"\n"`.

→ [Reply](#).

22 months ago, # |

▲ 0 ▼

This blog was a bit code-heavy for me when I first read this. Hence I have tried to simplify the concept a bit more in a more textual fashion in my own tutorial spin-off. I have tried to provide the intuition behind small-to-large merging including small to large on trees, also known as DSU on trees.



RestingRajarshi

However I. haven't provided any code as the code given by the OP is more than enough.

[DSU-on-Tree-Intuition](#)

→ [Reply](#).

17 months ago, # |

▲ 0 ▼

Method -3

WHy here use bigchild??

and why use "keep==0 " ??

→ [Reply](#).

17 months ago, # |

▲ 0 ▼

sorry but you absolutely wrong about the complexity, i know the trick merge smaller to greater but it works when you don't `for(i = 0; i < greater.size(); i++)` again, but here in  $O(n \log n^2)$  you use `cnt[v] = cnt[bigchild]`, it equal to `"for(int i = 0; i < cnt[bigchild].size(); i++) {}"`, so it's not  $O(n \log n^2)$

→ [Reply](#).



mra2322001



Arpa

17 months ago, # ^ |

▲ 0 ▼

Note that `cnt[v]` is a pointer.

→ [Reply](#).



mra2322001

17 months ago, # ^ |

▲ 0 ▼

you mean that it take  $O(\log n)$  time ?

→ [Reply](#).



17 months ago, # ^ |

▲ 0 ▼

$O(1)$ .

→ [Reply](#).

Arpa

→ Reply



mra2322001

17 months ago, # ^ |

▲ 0 ▼

got it, but how about the vector in your  $O(n \log n)$  time ?

→ Reply



Arpa

17 months ago, # ^ |

▲ 0 ▼

It's pointer too, isn't it?

→ Reply



mra2322001

17 months ago, # ^ |

▲ 0 ▼

thanks a lot, I got it.

→ Reply



tanmay28

16 months ago, # |

▲ 0 ▼

Can someone explain, why will the complexity of the first code will be  $O(n \log^2 n)$ ?

→ Reply



Arpa

16 months ago, # ^ |

▲ +3 ▼

Every vertex will appear in  $\log n$  cnt's, every time we want to insert a vertex it costs  $\log n$ , so the whole time complexity is  $O(n \log^2 n)$ .

→ Reply



tanmay28

16 months ago, # ^ |

▲ 0 ▼

1. That means, if I use unordered\_map my complexity will reduce to  $O(n \log n)$ ?
2. How can you say that every vertex will appear in  $\log(n)$  cnt's?

→ Reply



Arpa

16 months ago, # ^ |

▲ +3 ▼

1. Yes.
2. Read the proof of that for each vertex there is at most  $\log n$  light edges in the path between this vertex to root: [Link](#).

→ Reply



tanmay28

16 months ago, # ^ |

▲ 0 ▼

Thanks :)

→ Reply



savinov

16 months ago, # |

← Rev. 2

▲ +8 ▼

This particular problem can be solved in linear time:

```
int cnt[maxn];

void dfs(int v, int p){
    // Subtract cnt[c] for query here, e.g. answer[query] -= cnt[c];

    cnt[col[v]]++;
    for(auto u : g[v])
        if(u != p)
            dfs(u, v);

    // Add cnt[c] for query here, e.g. answer[query] += cnt[c];
}
```

→ Reply

idk,  
google it.

idLe.

14 months ago, # |

▲ 0 ▼

Can somebody help me understand why my solution [51874074](#) for problem [246E - Blood Cousins Return](#) gets WA on test 50? I tried to implement the Heavy Light Decomposition style dsu

→ [Reply](#).



lil\_chonk

13 months ago, # |

▲ 0 ▼

How to answer online queries using this technique??

→ [Reply](#).



Arpa

13 months ago, # ^ |

▲ +4 ▼

I didn't find till today.

→ [Reply](#).



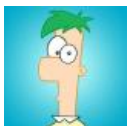
nsut\_coder

13 months ago, # |

▲ 0 ▼

why did he declare every time using pointer, can't we simply declare array of map for first approach as `map<int,int> cnt[maxx]` instead of `map<int,int>*cnt[maxx]` ?

→ [Reply](#).



abhayps

13 months ago, # ^ |

▲ +8 ▼

Due to this line, `cnt[v] = cnt[bigChild]`. Because we are using pointer, we only pass the reference. In other case, whole map will be copied(time complexity: linear in size of map).

→ [Reply](#).



Varun\_Shah

11 months ago, # ^ |

← Rev. 2

▲ 0 ▼

Then, will the information for the child be lost when we modify the parent's map which actually references the child's map... If so then how to answer queries in case when I need to access the nodes' map later?

→ [Reply](#).



Underdog\_eagle

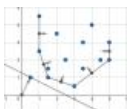
13 months ago, # |

← Rev. 2

▲ 0 ▼

Why is [my code](#) getting TLE for problem [600E — Lomsat gelral](#)? I'm using the technique no. 1 from this tutorial.

→ [Reply](#).



dragonfire100

11 months ago, # ^ |

← Rev. 2

▲ -8 ▼

your code is working in  $O(N^2)$ . You can't go through the whole ( $*cnt[u]$ ) to answer queries.

→ [Reply](#).



LODB---D

11 months ago, # ^ |

▲ 0 ▼

after 2 months!!!

you think he / she, will check this blog again looking for his answer?

→ [Reply](#).



lgkm39

11 months ago, # |

▲ +16 ▼

Love this.

→ [Reply](#).



chenyewei\_1234

9 months ago, # |

▲ +1 ▼

Thank you for the article.Now I understand it.

→ [Reply](#).



z4120

6 months ago, # |

▲ 0 ▼

Small nitpick: In solution 2, the comment

//now (\*cnt[v])[c] is the number of vertices in subtree of vertex v that has color c.

should be:

//now cnt[c] is the number of vertices in subtree of vertex v that has color c.

→ [Reply](#)



Mooncrater

4 months ago, # |

← Rev. 2

▲ +4 ▼

For everyone who needs an explanation, [here is a blog](#).

And obviously, thank you Arpa for this wonderful content!

→ [Reply](#)



Mmasker

3 months ago, # |

▲ 0 ▼

Thanks a lot really.

→ [Reply](#)



dunjen\_master

3 months ago, # |

▲ 0 ▼

For heavy-light decomposition style for the first add(v,p,1) except the big child sub-tree we are visiting the whole sub-tree of all the light children. Also for second add we are visiting again all the sub-tree if keep==0. How does the complexity becomes  $O(n \log n)$ . Having a hard time to understand, please can anyone tell me how is the complexity calculated.

→ [Reply](#)



Arpa

3 months ago, # ^ |

▲ +1 ▼

For each vertex like  $v$ , let  $l(v)$  number of light edges in the path from the root to  $v$ . Add function called on  $v$ ,  $l(v)$  times.  $l(v) \in \mathcal{O}(\log n)$ , so overall complexity is  $\mathcal{O}(n \log n)$ .

→ [Reply](#)



dunjen\_master

3 months ago, # ^ |

← Rev. 2

▲ 0 ▼

For second add function we have made big[bigChild]=0 for all bigChilds in subtree. Now when we call add(v,p,x) the whole subtree is visited. How does this make complexity  $O(n \log n)$

→ [Reply](#)

3 months ago, # ^ |

▲ 0 ▼



Mooncrater

We're deleting all the children only when the current node is not a bigChild, so that it doesn't mess up the global data structure. When the current node is a bigChild, then `keep=1`, and no data regarding the current subtree is deleted.

→ [Reply](#)



unrated\_grandmaster

7 weeks ago, # |

▲ 0 ▼

<https://csacademy.com/contest/round-31/task/uniform-trees/statement/>

another nice problem

→ [Reply](#)



SPyofgame

7 weeks ago, # |

▲ 0 ▼

Can we know the shortest distance from a vertex using dsu ?

→ [Reply](#)

---

[Codeforces](#) (c) Copyright 2010-2020 Mike Mirzayanov  
The only programming contests Web 2.0 platform  
Server time: May/16/2020 10:27:48<sup>UTC+5.5</sup> (h3).  
Desktop version, switch to [mobile version](#).  
[Privacy Policy](#).

Supported by



ITMO UNIVERSITY