Arpa's blog

[Tutorial] Sack (dsu on tree)

By Arpa, history, 3 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 `v` has some property in $O(nlogn)$ time (for all of the queries)?

For example:

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

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 `v` in `sz[v]` .

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])
```

→ Pay attention

Before contest
Mail.Ru Cup 2018 Round 2
45:54:43

Like 48 people like this. Be the first of your friends.

→ Top rated

#	User	Rating
1	tourist	3401
2	mnbvmar	3343
3	Radewoosh	3296
4	OO0OOO00O0OOO0O0...O	3264
5	scott_wu	3217
6	Benq	3187
7	Um_nik	3179
8	Petr	3140
9	fjzzq2002	3101
10	Swistakk	3089

Countries | Cities | Organizations View all →

→ Top contributors

#	User	Contrib.
1	Radewoosh	192
2	Errichto	172
3	rng_58	160
4	neal	156
5	Um_nik	155
6	tourist	154
7	Ashishgup	151
7	Petr	151
7	300iq	151
10	PikMike	150

View all →

→ Find user

Handle:

Find

→ Recent actions

MikeMirzayanov → Polygon: examples formatting update

```
        if(u != p)
            add(u, v, x)
}
void dfs(int v, int p){
    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.
    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] ]++;
```

soohotiam → Help Me All Code Lover!

icode_8990 → Explain the combinatorics used in soldier and cards

shoya → Problems which taught you a lot

Rollo → Polygon Generator

ParadiseSoul → SEERC 2018

patience → dsu

skywalkert → 2018-2019 ACM-ICPC, Asia Nanjing Regional Contest (Online Mirror on Gym)

A00000 → Number of integral solutions to a linear equation

Rtn_e → Segment Tree Problems

Ashishgup → Codeforces Data Scraping Websites

0day20hour → Hill Climb vs Golden Section Search

Le_Savage → An Interesting Problem?

snarknews → ICPC Uzbekistan Subregional 2018 — Teams and Sites Registration

sirknightingfail → A blog on the Sprague-Grundy Theorem

sirius_lyra → Why am I getting MLE ?

asifthegreat → Is it possible to set IOI style problems in cf polygon ?

brdy → Euler Tour Magic

lebedkin → Is there any way to use modern versions of JavaScript? (like ES6)

acmicpc2018dhaka → Teams going to ACM ICPC Dhaka Regional 2018

voidmax → Editorial Tinkoff Internship Warmup Round 2018 and Codeforces Round #475 (Div. 1 + Div. 2)

EonHino → A maximum matching problem with unknown edge

tourist55 → How fast is PyPy2 or PyPy3 compared to C++14 OR Java on Codeforces?

watchdogs132 → SPOJ FACTCG2 - Medium Factorization(Giving Wrong Answer)

CantbecomDiv1too_unlucky → Is there any kind of drugs to keep you focused during the round?

Detailed →

```
    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])
            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 **Sorasorasora**; 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.

dsu on tree, **sack**, **guni**

+68 Arpa 3 years ago 102

Comments (102)

Write comment?

maximaxi

3 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.

→ Reply

gotosleep

2 years ago, # ^ | -6

بدك تضل تتمنيك عكل بوستات الخرا ؟

→ Reply

dumbass

3 years ago, # | 0

What does the variable "keep" denote ?

→ Reply

NibNalin

3 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

Arpa

3 years ago, # ^ | ← Rev. 3 0

Look at last two lines:

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

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

2 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, doesnt the computation become faster? Can you please help clearing this doubt?

→ Reply

Arpa

2 years ago, # ^ | 0

Hi, Thanks.

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

ka89

2 years ago, # ^ | 0

oh..ok. Got it now. Thanks.

→ Reply

Lance_HAOH

16 months ago, # ^ ← Rev. 2 0

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

→ Reply

bk2dcradle

3 years ago, # ^ | +34

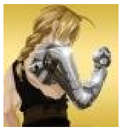
Observations to understand the complexity:

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.

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

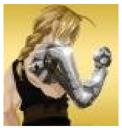


gogateiiit

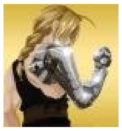
16 months ago, # ^ | ← Rev. 3 +3

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



16 months ago, # ^ | 0



gogateiiit

16 months ago, # ^ Rev. 2 +3

For those who had same doubt as I had: First lets 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 to measure how many times add function visits the ever y 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

the_art_of_war

3 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] ]--;
```

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

Arpa

2 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] ]++;
}
```

Instead of:

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

I have edited that.

→ Reply

SProf

2 years ago, # | 0

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

→ Reply

bk2dcradle

2 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.

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

shavidze

2 years ago, # ^ | 0

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[deth[v]] when depth[v] means distance from root to v?

→ Reply

Arpa

2 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:

Consider only subtree of vertice 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

2 years ago, # ^ | +5

thanks everyone , now i understand.
→ Reply

Sora233

2 years ago, # | +10

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

2 years ago, # ^ | 0

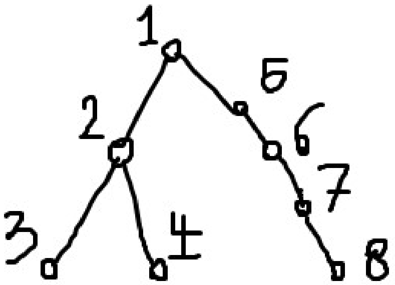
Thanks! Added to list ;)
→ Reply

Batman

2 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

2 years ago, # | ← Rev. 4 0

gabrielsimoes

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).

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

Proving explicitly why it is $nlogn$ 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 $logn$ steps to go from 1 to n. Summarizing, each node gets merged $logn$ times, so the total complexity is $O(nlogn)$).

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

2 years ago, # ^ | +2

Arpa

Thanks for your suggestions first !

I proved that it is O(nlogn) : *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

2 years ago, # ^ | ← Rev. 2 0

gabrielsimoes

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 different opinion,. Maybe it'll help a later reader, that's all.

→ Reply

2 years ago, # ^ | 0

pivorics

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

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 $\frac{s}{2} + \frac{s}{4} + \cdots = s$. So I don't understand where the $log(n)$ factor comes from.

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 $O(logn)$ factor involved.

Regardless can you perhaps elaborate a little bit

more on the time complexity of the dsu structure?
Thank you!
→ Reply

Arpa

2 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(logn) 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.

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

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, $T(n) = n / 2 + 2 * T(n / 2) = O(n.log(n))$
→ Reply

Emphi

22 months 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.*

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

Arpa

22 months ago, # ^ | 0

No. Dsu with size compare. Like this :

```
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

rcg_

2 years ago, # | ← Rev. 2 0

In easy to code but O(nlog^ 2), I cant'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

Arpa

2 years ago, # ^ | 0

Hi !

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

algo.experiments

2 years ago, # | ← Rev. 2 0

Ahh, thanks gabrielsimoes, for anyone struggling to understand: n*log^2n 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

bhargav104

2 years ago, # | +8

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

Arpa

2 years ago, # ^ | +5

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

Dalgerok

15 months ago, # ^ | 0

Just use hashes :) http://codeforces.com/contest/291/submission/29431526
→ Reply

bktl1love

7 months ago, # ^ | 0

Please send me a code of the solution with this techinque)
→ Reply

sengxian

23 months ago, # | ← Rev. 3 +13

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

abhigarg1796

23 months ago, # | +1

Hey Arpa,

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

23 months ago, # ^ | 0

23 months ago, # ^ | +2

Thanks a lot,

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

abhigarg1796

→ Reply

23 months ago, # ^ | 0

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

Arpa

→ Reply

23 months ago, # | 0

In the easy to code $O(nlogn)$ method `vec[v]` stores all the vertices in the 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

bhishma

→ Reply

23 months ago, # ^ | +1

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

Arpa

→ Reply

23 months ago, # ^ | 0

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

bhishma

→ Reply

23 months ago, # ^ | ← Rev. 3 +1

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

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 be me.

Arpa

Edit : Added.

→ Reply

23 months ago, # ^ | 0

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

SaYami

→ Reply

23 months ago, # ^ | ← Rev. 2 +6

I'll add one of them tonight.

Edit : Added.
→ Reply

Arpa

SaYami

23 months ago, # ^ | 0

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

Agassaa

22 months 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 tutorial there is no "merge" function. Am I missing something?

Also I see that, in your 600E's solution 14554536 you used merge functon. I can't understand, could you please explain that code?
→ Reply

Arpa

22 months ago, # ^ | 0

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

beAwesome

22 months ago, # | +1

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

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 $O(nlogn)$?

Thanks!
→ Reply

Arpa

22 months ago, # ^ | 0

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

beAwesome

22 months ago, # ^ | 0

If you use unordered_map, does it become $O(n \cdot logn)$, then?
→ Reply

Arpa

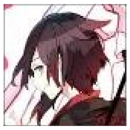
22 months ago, # ^ | 0

Unordered_map is theoretically $O(n)$ per query. But you can suppose that it's O(1) per query in code.
→ Reply

surajghosh

21 month(s) ago, # | 0

This 758E.Read this comment on how to use it.
→ Reply


HUECTRUM1

20 months ago, # | 0

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

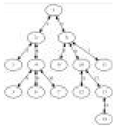
→ Reply

satyaki3794

20 months 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


vatsalsharma376

20 months ago, # | +8

101 Hack 47 Summing Tree was solved using this technique by satyaki3794 Submission

→ Reply

lukecavabarrett

20 months ago, # | +3

also 778C - Peterson Polyglot is solvable with a similar tecnique: is that DSU on tree?

→ Reply

radoslav11

20 months ago, # ^ | +3

yes

→ Reply

__W__

17 months 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

17 months ago, # ^ | +5

It's a Persian contest.

→ Reply

__W__

17 months ago, # ^ | +6

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

→ Reply

Arpa

17 months ago, # ^ | +6

Link.

→ Reply

__W__

17 months ago, # ^ | +11

Thank you!

→ Reply

tak_fate

17 months 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

16 months ago, # | 0

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

→ Reply

maxorand

15 months ago, # | 0

Arpa, in the **Easy to code but $O(nlog^2n)$** 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

15 months ago, # ^ | 0

Hi. Thanks. Edited.

→ Reply

Trath

14 months ago, # | 0

You can solve APIO 2012 Dispatching with this technique too.

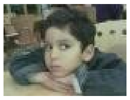
→ Reply

mochow

12 months ago, # | 0

IOI 2011 — Race What is the idea of DSU on tree for this problem? I know of a solution based on Centroid Decomposition.

→ Reply



CyberSword

10 months ago, # | ← Rev. 2 +8

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

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

→ Reply

pk845

9 months ago, # | 0

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

→ Reply

shahidul_brur

9 months 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

Arpa

9 months ago, # ^ | 0

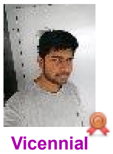
Hi.

Thanks for your feedback. Here it is: Link.

→ Reply

9 months ago, # ^ | 0

Thank you !



Vicennial

8 months 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

Izoi_hhn

8 months 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

kirito_

8 months ago, # | -7

No good explanation! Only code! Worst tutorial.

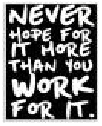
→ Reply

satvik007

8 months ago, # ^ | 0

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

→ Reply



M_H_H_7

8 months ago, # | +5

Nice tutorial!

and also happy Nowruz:)

→ Reply

Ehsan_sShuvo

7 months ago, # ^ | ← Rev. 2 0

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

→ Reply

vivace_jr

5 months ago, # ^ | ← Rev. 3 0

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

→ Reply

vivace_jr

5 months ago, # ^ | ← Rev. 2 0

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

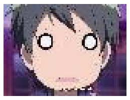
→ Reply

Ehsan_sShuvo

7 months 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

6 months ago, # | +8

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

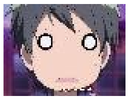
→ Reply

a.kleber.d

6 months ago, # | 0

What is col[v]?

→ Reply



Azurey

6 months ago, # ^ | 0

Color of the v-th vertex

→ Reply

_Na2Th

5 months 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 `st[v]` and `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

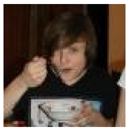
→ Reply

yaksha

5 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(nlog^2n0 method ,.. I saw your solution and I see its the same as mine, but mine gets MLE.. My solution :- 39325497

→ Reply



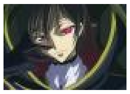
dmkozyrev

5 months ago, # ^ | ← Rev. 3 +5

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

4 months ago, # | 0

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

→ Reply

The_Wolfpack

4 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

4 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

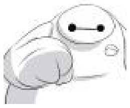
The_Wolfpack

4 months ago, # ^ | +13

I got it, thanks for the quick reply!

→ Reply

4 months ago, # | +18

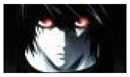

baymaxx

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

4 months ago, # | 0


prayas0709

Can someone explain why this solution 40510312 is getting TLE on problem 600E - Lomsat gelral. I'm using style 4.

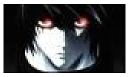
→ Reply

4 months ago, # ^ | +8

Irvideckis

Can't view your submission

→ Reply


prayas0709

4 months ago, # ^ | +5

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

→ Reply

3 months ago, # | 0

ChandyShot

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

rajarshi_basu

3 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 knows as DSU on trees.

However I. haven't provided any code as the code given by the OP is more than enough.
DSU-on-Tree-Intuition
→ Reply