##### <https://codeforces.com/blog/entry/101271>

## The diameter

Given an unweighted tree, let's define dist(a,b)=dist(𝑎,𝑏)= the number of edges in the simple path a→b𝑎→𝑏.

A diameter of the tree a→b𝑎→𝑏 is the longest path, i.e., the one that maximizes dist(a,b)dist(𝑎,𝑏) over all pairs of nodes. If there are multiple diameters, let's pick any of them.

The same definition is valid for a weighted tree with nonnegative weights (with dist(a,b)=dist(𝑎,𝑏)= the sum of the weights of the edges in the simple path a→b𝑎→𝑏).

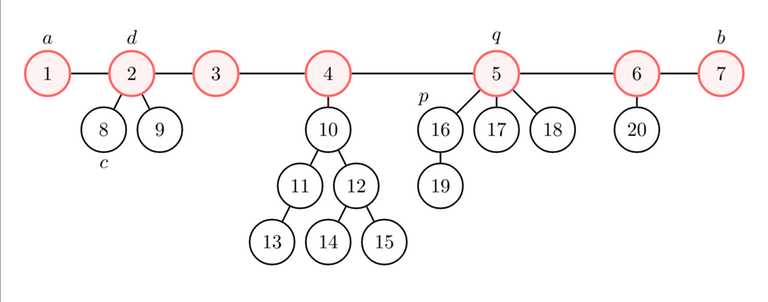
Finding a diameter

Given a tree with n𝑛 nodes are multiple ways to find a diameter. Here is one of the simplest ways:

*Run a DFS from any node*p𝑝*. Let*a𝑎*be a node whose distance from node*p𝑝*is maximized. Run another DFS from node*a𝑎*. Let*b𝑏*be a node whose distance from node*a𝑎*is maximized.*a→b𝑎→𝑏*is a diameter.*

Tree = edges of a diameter + forest

Before proving the previous algorithm, let's analyze the structure of the tree (we will mention the diameter, but we will not use the fact that a→b𝑎→𝑏 is actually a diameter before proving it).



We started a DFS from node p=16𝑝=16, and we got that node a=1𝑎=1 is the farthest from p𝑝, and node b=7𝑏=7 is the farthest from a𝑎.

Let's represent the diameter on a line. If you remove the edges of the diameter, you get a forest (i.e., several trees). Let's root each tree at the node in the diameter. What's the height (i.e., the maximum distance from the root to any node) of each component?

Let q𝑞 be the root of the component of p𝑝. Let's consider any component whose root d𝑑 is between a𝑎 (included) and q𝑞 (excluded), and one of its nodes c𝑐.

We get

dist(p,a)≥dist(p,c)⟹dist(p,a)−dist(p,d)≥dist(p,c)−dist(p,d)⟹dist(a,d)≥dist(c,d)dist(𝑝,𝑎)≥dist(𝑝,𝑐)⟹dist(𝑝,𝑎)−dist(𝑝,𝑑)≥dist(𝑝,𝑐)−dist(𝑝,𝑑)⟹dist(𝑎,𝑑)≥dist(𝑐,𝑑).

In other words, the height of each component with root in the left half of the diameter (i.e., dist(a,d)<dist(d,b)dist(𝑎,𝑑)<dist(𝑑,𝑏)) is at most the distance of the root of the component from the left end of the diameter.

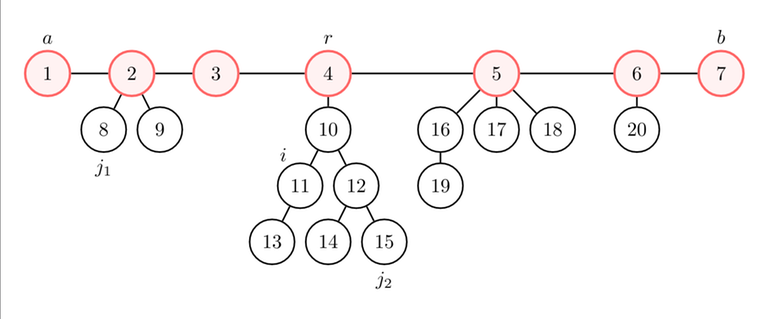
You can prove the same statement for the right half of the diameter (i.e., dist(a,d)≥dist(d,b)dist(𝑎,𝑑)≥dist(𝑑,𝑏)), using that b𝑏 is the farthest node from a𝑎.

Farthest node for each node

For each node i𝑖, let's find a node j𝑗 such that dist(i,j)dist(𝑖,𝑗) is maximum.

*Claim:*j=a𝑗=𝑎*or*j=b𝑗=𝑏*always works.*

Proof:



* If j=j1𝑗=𝑗1 works (j1𝑗1 is not in the same component of i𝑖; let's assume without loss of generality that j1𝑗1 is closer to a𝑎 than to b𝑏), dist(i,j1)=dist(i,r)+dist(r,j1)≤dist(i,r)+dist(r,a)=dist(i,a)dist(𝑖,𝑗1)=dist(𝑖,𝑟)+dist(𝑟,𝑗1)≤dist(𝑖,𝑟)+dist(𝑟,𝑎)=dist(𝑖,𝑎). Then, j=a𝑗=𝑎 also works.
* If j=j2𝑗=𝑗2 works (j2𝑗2 is in the same component of i𝑖), dist(i,j2)≤dist(i,r)+dist(r,j2)≤dist(i,r)+dist(r,a)=dist(i,a)dist(𝑖,𝑗2)≤dist(𝑖,𝑟)+dist(𝑟,𝑗2)≤dist(𝑖,𝑟)+dist(𝑟,𝑎)=dist(𝑖,𝑎). Then, j=a𝑗=𝑎 also works.

Proof that a→b𝑎→𝑏 is a diameter

Now we can finish the proof.

Suppose that u→v𝑢→𝑣 is a diameter. We have either dist(u,a)≥dist(u,v)dist(𝑢,𝑎)≥dist(𝑢,𝑣) or dist(u,b)≥dist(u,v)dist(𝑢,𝑏)≥dist(𝑢,𝑣) (see "Farthest node for each node").

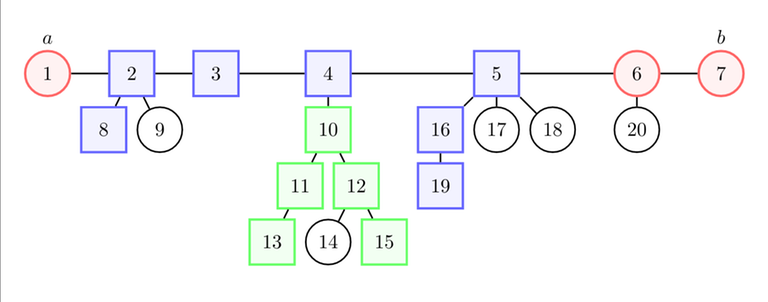
Let's assume without loss of generality that dist(u,b)≥dist(u,v)dist(𝑢,𝑏)≥dist(𝑢,𝑣). We get dist(a,b)≥dist(u,b)≥dist(u,v)dist(𝑎,𝑏)≥dist(𝑢,𝑏)≥dist(𝑢,𝑣), so a→b𝑎→𝑏 is a diameter.

Observations

The algorithm also works in a weighted tree with positive edges (we've never used that the weights are 11).

However, it doesn't work on general graphs ([discussion](https://codeforces.com/blog/entry/4116)).

How to use the diameter



Most of the times, spamming "the farthest node from each node is one end of the diameter" and "the height of each component is smaller than the distance to the closest end of the diameter" is enough to reduce the problem to something simpler.

Find a diameter a→b𝑎→𝑏 (from now, a→b𝑎→𝑏 will always be a diameter, unless otherwise stated). Now, you may need to consider any path of the tree. There are two cases: the path intersects (blue) or doesn't intersect (green) the diameter.

Then, you may wonder how to make the path longer / "more optimal" / etc. according to the statement. For example, you may need to use dist(7,5)≥dist(5,19)dist(7,5)≥dist(5,19) to show that 8→78→7 is "more optimal" than 8→198→19.