Problem List

Dynamic Layout 0

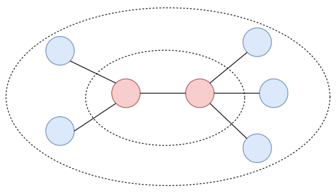
Description Editorial Solutions (1.1K) Subn *i* C++ Auto

centroids in a tree-alike graph.

Algorithm

> Given the above intuition, the problem is now reduced down to looking for all the ***centroid*** nodes in a tree-alike graph, which in addition are no more than two.

The idea is that we *trim* out the leaf nodes layer by layer, until we reach the *core* of the graph, which are the centroids nodes.



Once we trim out the first layer of the leaf nodes (nodes that have only one connection), some of the non-leaf nodes would become leaf nodes.

The trimming process continues until there are only two nodes left in the graph, which are the *centroids* that we are looking for.

The above algorithm resembles the *topological sorting* algorithm which generates the order of objects based on their dependencies.
For instance, in the scenario of course scheduling, the courses that have the least dependency would appear first in the order.

In our case, we trim out the leaf nodes first, which are the **farther** away from the centroids.
At each step, the nodes we trim out are

```
class Solution {
public:
    vector<int> findMinHeightTrees(int
        if(n == 1)
            return {0};

        // since we know there're a
        // we want to design a topo
        // nodes left

        // build graph
        vector<vector<int>> g(n);
        vector<int> degree(n,0);
        for(auto &e : edges) {
            g[e[0]].push_back(e[1]);
            g[e[1]].push_back(e[0]);
            degree[e[0]]++;
            degree[e[1]]++;
        }

        // BFS based topo sort
        vector<int> curLeaves;
        vector<int> nextLeaves;
        for(int node=0; node<n; ++node)
            if(degree[node] == 1)
                curLeaves.push_back(nod
        while(n > 2) {  // remaining no
            for(int node : curLeaves) {
                --degree[node];
                for(int next : g[node])
                    if(--degree[next] =
                        nextLeaves.push
            }
            n -= curLeaves.size();
```

Saved to cloud

Console Run Subr