USACO DEC10 Problem 'exercise' Analysis

by Travis Hance

First, conduct a binary search on the answer, D. To do this, just find a way to check, for any D, if it is possible to make S cuts such that each tree has diameter at most D.

We will give a greedy algorithm which will compute the smallest number of cuts needed if each subtree is to have diameter at most D.

Root the tree arbitrarily.

Suppose we are processing a vertex A with children C_1, C_2, ..., C_k, and suppose that any necessary cuts have been made to edges below these k vertices. Let the depth of a vertex be the maximum length from that vertex down to a leaf.

The diameter of the subtree rooted at A is the maximum of depth(C_i) + depth(C_j) + 2, over all distinct i and j, if the maximal diameter goes through A, or the diameter of the subtree rooted at C_i for some i. Since we are assuming all lower cuts have been made, we actually just need to ensure that depth(C_i) + depth(C_j) + 2 is at most D.

If it is at most D, then there is certainly no reason to make any more cuts within this tree. If it is greater than D, then we need to cut off some edges between A and its children. We want to cut off children of maximum depth (because given a choice between cutting off two vertices, it is always better to cut off the one which would contribute the most to a large diameter), and we cut off children until the maximum value of depth(C_i) + depth(C_j) + 2 is at most D.

Perform a depth first search through the tree, and for each node, process its children before processing itself.

Meanwhile, keep track of the total number of cuts and then check if this value is less than or equal to S. This algorithm is $O(n \log^2 n)$. One log factor is for the binary search, the n factor is for traversing the tree, and another log factor is for sorting the depths of a nodes children.

Notice how this second log factor is in fact very small, since most nodes will have a small number of children. (In fact, if you are careful enough, you can cut off this extra log factor entirely.)

Here is an $O(n \log^2 n)$ solution due to Brian Hamrick:

```
#include<cstdio>
#include<cstdlib>
#include<vector>
#include<algorithm>

#define MAXN 1000005

using namespace std;

vector adj[MAXN];
bool vis[MAXN];
```

```
pair min_cuts_min_height(int v, int maxdiam) {
    vis[v] = 1;
    pair<int,int> ans;
    ans.first = 0;
    ans.second = 0;
    vector<int> cheights;
    for(int i = 0; i<adj[v].size(); i++) {
        if (!vis[adj[v][i]]) {
            pair<int,int> p = min_cuts_min_height(adj[v][i],maxdiam);
            ans.first += p.first;
            cheights.push_back(p.second);
        }
    }
    if (cheights.size () > 0) {
        sort (cheights.begin (),cheights.end ());
        reverse (cheights.begin (),cheights.end ());
        int i;
        for(i = 0; i < cheights.size()-1 &&
                 cheights[i] + cheights[i+1] + 2 > maxdiam; i++);
        if (i < cheights.size()-1) {
            ans.first += i;
            ans.second = cheights[i]+1;
        } else if (cheights[i] + 1 > maxdiam) {
            ans.first += i+1;
            ans.second = 0;
        } else {
            ans.first += i;
            ans.second = cheights[i]+1;
        }
    }
    return ans;
}

int main () {
    FILE *fin = fopen ("exercise.in","r");
    FILE *fout = fopen ("exercise.out","w");
    int N, S;

    fscanf (fin,"%d%d",&N,&S);
    for (int i = 0; i < N-1; i++) {
        int a, b;
        fscanf (fin,"%d%d",&a,&b);
        a--; b--;
        adj[a].push_back (b);
        adj[b].push_back (a);
    }
    int hi = N-1, lo = 0;
    while (lo < hi) {
        int g = (hi + lo)/2;
        for (int i = 0; i < N; i++) vis[i] = 0;
        pair<int,int> p = min_cuts_min_height (0, g);
        if(p.first > S) lo = g+1;
        else                hi = g;
    }
    fprintf (fout,"%d\n",lo);
    fclose (fin);
    fclose (fout);
    return 0;
}
```