USACO NOV10 Problem 'vacation' Analysis

by Neal Wu

Note that from the information given in the problem, the cows form a graph that is a tree. We can use dynamic programming on subtrees to solve this problem. Specifically, we use DP to determine the following:

- the maximum number of cows in a subtree that can be visited if the root is visited.
- the maximum number of cows in a subtree that can be visited if the root is not visited.

To obtain a method for recursing on a node's children, note that if the node is visited, then all its children must not be visited, and if the node is not visited, its children may be either visited or not. We may take the maximum of the two. Overall, this gives an O(N) solution. Sample code is below:

```
#include <cstdio>
#include <vector>
#include <algorithm>
using namespace std;

FILE *fin = fopen ("vacation.in", "r"), *fout = fopen ("vacation.out", "w");

const int MAXN = 50005;

int N, vis[MAXN], novis[MAXN];
vector <int> adj[MAXN];

void solve (int ind, int parent) {
    vis[ind] = 1;
    novis[ind] = 0;

    for (int i = 0; i < (int) adj[ind].size (); i++) {
        int next = adj[ind][i];
        if (next == parent) continue;

        /* recurse on the children of the node */
        solve (next, ind);

        /* compute as explained above */
        vis[ind] += novis[next];
        novis[ind] += max (vis[next], novis[next]);
    }
}

int main () {
    fscanf (fin, "%d", &N);
    for (int a, b, i = 0; i < N - 1; i++) {
        fscanf (fin, "%d %d", &a, &b);
        a--; b--;
        adj[a].push_back (b);
        adj[b].push_back (a);
    }

    solve (0, -1);
    fprintf (fout, "%d\n", max (vis[0], novis[0]));
}
```