USACO MAR11 Problem 'meetplace' Analysis

by Fatih Gelgi

A simple idea is to use the distances from nodes to their ancestors. Note that in the question paths are one way from nodes to their ancestors. After calculating the distances from nodes to their ancestor, the meeting place of a and b is j where distance(a,j) + distance(b,j) is minimum.

This algorithm requires O(N^2) time to calculate distances and O(MN) time to answer questions. Hence the complexity is O(N(N+M)) which is fast enough for the problem.

Below is an example solution:

```
#include <fstream>
#define MAXN 1001

using namespace std;

int n,m,dist[MAXN][MAXN],parent[MAXN];

int main()
{
	ifstream fin("meetplace.in");
	ofstream fout("meetplace.out");

	fin >> n >> m;
	for (int i=2; i <= n; i++)
		fin >> parent[i];

	// initialize distances with a large number
	for (int i=1; i <= n; i++)
		for (int j=1; j <= n; j++)
			dist[i][j]=MAXN;

	// calculate distances from nodes to their ancestors
	for (int i=1; i <= n; i++)
		for (int j=i,k=0; j; j=parent[j])
			dist[i][j]=k++;

	// answer the queries
	for (int i=0; i < m; i++)	{
		int a,b,meet,d=MAXN*2;
		fin >> a >> b;

		// find meeting place j with minimum distance
		for (int j=1; j <= n; j++)
			if (d > dist[a][j]+dist[b][j])
				d=dist[a][j]+dist[b][j],meet=j;

		fout << meet << "\n";
	}

	fin.close();
```

```
        fout.close();
}
```

Another idea is to calculate heights of nodes then find the meeting place by syncronizing the heights while iterating on parents. Algorithm requires O(N) time for calculating the heights and O(MN) time to answer all queries which is O(MN) in total.

Here is Damon's solution:

```
#include <stdio.h>

const int MAXN = 5005;
FILE *fin = fopen("meetplace.in", "r"), *fout = fopen("meetplace.out", "w");
int N, M, P[MAXN], height[MAXN];

int calc_height(int node) {
        if (height[node] != -1)
                return height[node];

        return height[node] = calc_height(P[node]) + 1;
}

int find_meeting(int a, int b) {
        while (a != b) {
                if (height[a] < height[b])
                        b = P[b];
                else
                        a = P[a];
        }
        return a;
}

int main() {
        int a, b;
        fscanf(fin, "%d%d", &N, &M);

        height[1] = 0;
        for (int i = 2; i <= N; i++) {
                fscanf(fin, "%d", &P[i]);
                height[i] = -1;
        }

        for (int i = 2; i <= N; i++)
                calc_height(i);

        for (int i = 0; i < M; i++) {
                fscanf(fin, "%d%d", &a, &b);
                fprintf(fout, "%d\n", find_meeting(a, b));
        }

        fclose(fin); fclose(fout);
        return 0;
}
```