Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2014

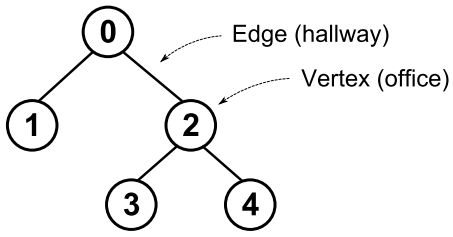
Lengthy Traveling Salesman

`lengthy.c`, `lengthy.cpp`, `Lengthy.java`

Karel wants to register his robot Karel for a robot contest. To do so, he needs to obtain a certificate of safety of his robot, fill in various waivers, etc. He needs to start at the entrance of the building of the contest organizers, visit all prescribed offices in a particular order, and return back to the entrance. Fortunately, the building is easy to navigate, as its hallways do not form any cycles, and thus there is only one path between any two offices. However, Karel would like to know in advance how long it will take to visit all the offices.

Given a building layout and an ordering of all offices v_1, v_2, ..., v_n, the task is to determine the length of the walk going from v_1 to v_2, then from v_2 to v_3, ..., and finally from v_n back to v_1.

This sounds like a nice contest problem, doesn't it? We want to give you an idea what it is like to organize a programming contest. If you want to organize one (please contact us after this contest if you *really* do), writing the textual problem statement and sample solutions is not enough. We also need to prepare the test data that will be used to verify the correctness of submissions. Beside others, we want to eliminate inefficient solutions, so we want to generate a permutation of offices that results in the maximal possible walk. Your task is to find it.



For the purpose of this problem, the building is described as a tree with one office in each of its vertices. For those who are not familiar with graph terms, a short informal definition follows: A *tree* consist of a set of *vertices* connected by *edges* (each edge connects two vertices) in such a way that there is exactly one possible path between any two vertices v_i and v_j. The *length* of the path is the number of edges that must be traversed when traveling from v_i to v_j and it is denoted $dist(v_i, v_j)$.

Input Specification

The input contains several test cases. The first line of each test case contains an integer N, giving the number of the vertices in a tree ($2 \leq N \leq 10\,000$). The vertices are numbered from 0 to $N-1$. The i-th of the following $N-1$ lines contains an integer f_i ($0 \leq f_i < i$), indicating that the tree contains an edge between vertices i and f_i.

There is one empty line after each test case.

Output Specification

For each test case, print a single line containing N integers $v_1, \ldots, v_n$ separated by spaces. Each of the numbers 0, 1, ..., $N-1$ must appear exactly once. The sum

$$\sum_{i=1}^{n-1} dist(v_i, v_{i+1}) \quad + \quad dist(v_n, v_1)$$

should be the maximum possible. If there are multiple solutions (as they usually are), you may print any of them.

Sample Input

```
5
0
0
2
2

2
0
```

Output for Sample Input

```
1 2 0 4 3
0 1
```