E. Policeman and a Tree

You are given a tree (a connected non-oriented graph without cycles) with vertices numbered from 1 to n, and the length of the i-th edge is w_i. In the vertex s there is a policeman, in the vertices $x_1, x_2, ..., x_m$ ($x_j \neq s$) m criminals are located.

The policeman can walk along the edges with speed 1, the criminals can move with arbitrary large speed. If a criminal at some moment is at the same point as the policeman, he instantly gets caught by the policeman. Determine the time needed for the policeman to catch all criminals, assuming everybody behaves optimally (i.e. the criminals maximize that time, the policeman minimizes that time). Everybody knows positions of everybody else at any moment of time.

Input

The first line contains single integer n ($1 \leq n \leq 50$) — the number of vertices in the tree. The next $n - 1$ lines contain three integers each: u_i, v_i, w_i ($1 \leq u_i, v_i \leq n$, $1 \leq w_i \leq 50$) denoting edges and their lengths. It is guaranteed that the given graph is a tree.

The next line contains single integer s ($1 \leq s \leq n$) — the number of vertex where the policeman starts.

The next line contains single integer m ($1 \leq m \leq 50$) — the number of criminals. The next line contains m integers $x_1, x_2, ..., x_m$ ($1 \leq x_j \leq n$, $x_j \neq s$) — the number of vertices where the criminals are located. x_j are not necessarily distinct.

Output

If the policeman can't catch criminals, print single line "`Terrorists win`" (without quotes).

Otherwise, print single integer — the time needed to catch all criminals.

Examples

input

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

output

```
8
```

input

```
6
1 2 3
2 3 5
3 4 1
3 5 4
2 6 3
2
3
1 3 5
```

output

```
21
```

Note

In the first example one of the optimal scenarios is the following. The criminal number 2 moves to vertex 3, the criminal 4 — to vertex 4. The policeman goes to vertex 4 and catches two criminals. After that the criminal number 1 moves to the vertex 2. The policeman goes to vertex 3 and catches criminal 2, then goes to the vertex 2 and catches the remaining criminal.