Task Puma

⏳3 sec. 💾 256 MB

A wild black puma has been spotted in Shumen Plateau Park near the city of Shumen. The regional government has called K zoologists to capture the animal.

For simplicity, we will consider the park to have N main places denoted (physically) with the integers from 0 to $N-1$. These places are connected by $N-1$ two-way paths so that it is possible to reach any place from any other via these paths. We will call two places directly connected by a path *adjacent*.

The government wants each place to be explored by at least one zoologist. Each zoologist will start from the park's starting point which is place number 0 and move along the paths **without** revisiting places they have already explored. They will exit the park when there are no more places to go (without returning). We will call such places (where you have to exit the park upon arrival) *exits*.

For safety reasons, only one zoologist can be in the park at a time. Due to competition between the zoologists, they have decided that the only "communication" between them will be marking the places with integers from 0 to M inclusive. Initially, all markings are set to 0.

The zoologists will enter and exit one after another, in an order not known beforehand. As someone stole the map of the park, when a zoologist is at a place, they:

- know where they came from;
- know the current place's number and marking;
- see the markings of all *adjacent* places, but **not** their numbers.

Help the zoologists devise a strategy so that every place is explored by at least one zoologist. You also know that K is exactly equal to the number of *exits* and that M is enough to have a valid strategy.

Implementation details

The strategy should be implemented in the following function:

```
void zoologist (int N, int K, int M, std::vector<int> p, int m, std::
    vector<int> a)
```

- N: the number of places in the park;
- K: the number of zoologists;
- M: the maximum marking value;

- p: vector of length N, representing the paths; for $1 \leq i \leq N-1$, there is a path between place $p[i]$ and i, and $p[0] = -1$;
- m: the current marking of the starting place 0;
- a: vector, representing the markings of all *adjacent* places of 0, **in random order**.

This function will be called K times – once for each of the zoologists. The calls will be consecutive, i.e. when the first zoologist finishes their exploration, a call for the second zoologist will be made with the current state of the markings and so on. These calls can be in **separate executions** for different zoologists, but some calls may be in the same execution. This simulates that the only communication between the zoologists will be the markings. Therefore, your program **should not** try to use global memory between different calls.

The above function can make calls to the following functions to change the marking of the current place or move to an *adjacent* place:

```
void set_marking (int m)
```

- m: the value of the new marking for the current place, which has to be an integer between 0 and M inclusive.

```
std::pair<int, std::vector<int>> movement (int i)
```

- i: is the index of the path to move along, according to the vector of markings of the *adjacent* places (in the beginning this is the vector a and after that it is the second element of the last returned value by the function `movement`);
- this function returns two values – the first is the number of the next place and the second is a vector representing the markings of all places *adjacent* to the next place (excluding the marking of the place the zoologist came from), **in random order**.

If the zoologist is at an *exit*, simply terminate the function to simulate leaving the park.

Constraints

- $1 \leq N \leq 1000$;
- K is equal to the number of *exits*;
- There is a valid strategy with the given M.

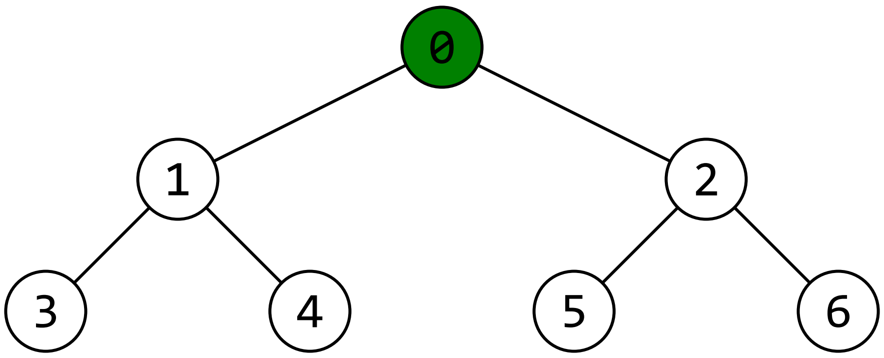
Subtasks

Subtask	Points	Required subtasks	M	Other constraints
0	0	$-$	$-$	The example.
1	6	$-$	$= 1$	Place 0 is directly connected to all other places.
2	14	$-$	$= N$	-
3	15	0	$= C^*$	$N = 2^s - 1$, place 0 has two *adjacent* places, $2^{s-1} - 2$ of the places have three *adjacent* places and the other ones have only one *adjacent* place.
4	18	$0, 3$		There are no unique values in the vector p except -1.
5	47	$0 - 4$		-

* C is the maximum number of paths a zoologist can traverse starting from place 0.

Example

Consider the following illustration of the park's places and paths:



The following tables show an example sequence of moves and markings for $M = 4$:

Your actions	Actions of the grader
	`zoologist(7,4,4,{-1, 0, 0, 1, 1, 2, 2},0,{0, 0})`
`set_marking(1)`	
`movement(0)`	`return {1, {0, 0}}`
`movement(1)`	`return {4, {}}`
`set_marking(2)`	

Your actions	Actions of the grader
	`zoologist(7,4,4,{-1, 0, 0, 1, 1, 2, 2},1,{0, 0})`
`set_marking(2)`	
`movement(0)`	`return {2, {0, 0}}`
`set_marking(1)`	
`movement(1)`	`return {6, {}}`
`set_marking(1)`	
	`zoologist(7,4,4,{-1, 0, 0, 1, 1, 2, 2},2,{1, 0})`
`set_marking(3)`	
`movement(1)`	`return {1, {2, 0}}`
`set_marking(0)`	
`movement(1)`	`return {3, {}}`
	`zoologist(7,4,4,{-1, 0, 0, 1, 1, 2, 2},3,{0, 1})`
`set_marking(4)`	
`movement(1)`	`return {2, {0, 1}}`
`movement(0)`	`return {5, {}}`
`set_marking(1)`	

Sample grader

For the sample grader, all K calls to `zoologist` are in the same execution.

The input format is the following:

- line 1: three integers – the values of N, K, and M;
- line 2: N integers – the values of vector p.

The output format is the following:

- line 1: N boolean numbers – 1 if the corresponding place is explored by at least one zoologist, 0 otherwise.

For more detailed feedback, change the value of the macro `DETAILED` from `false` to `true` in the first line of the sample grader.