
# *Greetings From Globussoft*

- Given below are 5 Programming questions, you have to solve any 3 out of 5 questions.

- These 5 questions you can attempt in any technology like C/C++, java, .Net, PHP

- To solve these 3 questions you've max. 3 hours.

- While Solving these questions you are not allowed to use any Search Engine like Google, Yahoo, Bing ...

All the best for your test

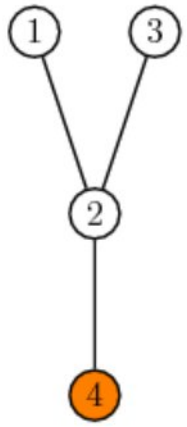
Globussoft

# QUESTION - 1

Hey, ACRush and Jelly are playing a game ! Let take a look at its rule:

You are given a tree. Two players take turns cutting edges on a tree. Some nodes is on the "ground". When a player cuts an edge, all the edges that are no longer connected to the ground disappear. The player who can not take a move loses.

ACRush plays first. Both of them are very good players. If you know state of the tree they are playing with, can you guess who will win?



Node 4 is on the ground.

## Input

Input consists of multiple test-cases. The first line contains one integer *t* - number of cases (0 < *t* <= 20). For each case, the input format is following. The first line contains one integer *N* (1 <= *N* <= 100000). The next line *N* integers s[i] (1 or 0). If s[i] is 1, the *i*-th node is on the ground. If s[i] is 0, the *i*-th node is not on the ground. Each line of the following *N* - 1 lines contains two integers u, v. They denote there is an edge between node u and node v (1 <= u,v <= N).
There is no blank line after each case.

## Output

For each case, output who will win the game. If ACRush wins, output *1*; otherwise, output *0* (Jelly wins).
There is no blank line after each case.

## Example

**Input:**

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

```
2 3
2 4
```

**Output:**

```
1
```

# QUESTION – 2

After a lot of exams at school, Amber and his friends Ahyangyi, Dragon have a short vacation in Hong Kong Disneyland. Many interesting places they want to visit there: resort, castle,... . In each place and between them, there are special bidirectional rails, so that the visitors can drive a small special car, go around and have a sightseeing tour. This rail system is quite optimal it has tree shape ! Each time, you start a new route (or you can call it "path") with a car, you must purchase a new ticket.

Amber and his friends surely want to visit all places, and each place exactly once, so bored to visit one place many times. But the trouble is they don't carry much money. So Amber thinks about a good way to purchase as small number of tickets as possible (i.e. minimal number of routes). We don't care how they can switch cars during their trip.

Now you're given maps of the Disneyland, please help them to find an optimal solution.

Take a look at the figure below:

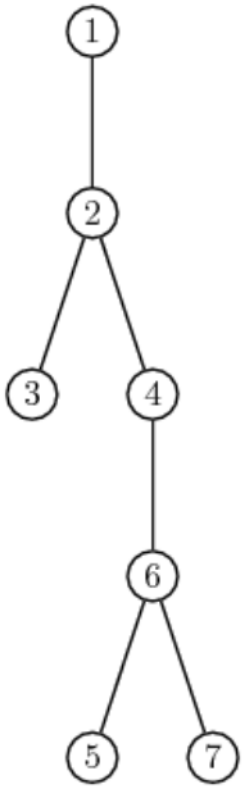



Figure 1: An example map of Disneyland

There are many optimal solutions here and Amber must purchase at least 3 tickets, for 3 disjoint routes. Two possible solutions are:
**Solution 1:**
1-st route: they visit 1 2 3
2-nd route: they visit 4
3-rd route: they visit 5 6 7
**Solution 2:**
1-st route: they visit 1 2 4 6 5
2-nd route: they visit 3
3-rd route: they visit 7

## Input

There may be many maps in one input file. The first line of file is number of maps *T* (0 < T <= 10). The following line is blank. Then, there are the descriptions of *T* maps.

For each map, the first line contains one integer *N* --- number of places in the Disneyland (0 < N <= 10000). We number places from 1 to N. Next N-1 lines contain N-1 rails between places --- Each line contains a pair (u, v) means there is a rail between place u and place v (1 <= u,v <= N). There is a blank line after each description.

## Output

For each map, the first line, write minimal number of routes *K*. Next *K* lines, show out the routes in your solution, each has form u[1] u[2]...u[m], means the route starts at place u[1] then visits place u[2],..., ends at place u[m]. So between 2 consecutive places in each route must have a rail. If there are many solutions, any of them will be accepted.
There is no blank line after each case.

## Example

**Input:**

```
1

7
1 2
2 3
2 4
4 6
5 6
6 7
```

**Output:**

```
3
1 2 3
4
5 6 7
```

# QUESTION – 3

You are given a node-labeled rooted tree with *n* nodes.

Define the query (*x*, *k*): Find the node whose label is *k*-th largest in the subtree of the node *x*. Assume no two nodes have the same labels.

## Input

The first line contains one integer *n* ($1 <= n <= 10^5$). The next line contains *n* integers $l_i$ ($0 <= l_i <= 10^9$) which denotes the label of the *i*-th node.

Each line of the following *n* - 1 lines contains two integers *u*, *v*. They denote there is an edge between node *u* and node *v*. Node 1 is the root of the tree.

The next line contains one integer *m* ($1 <= m <= 10^4$) which denotes the number of the queries. Each line of the next *m* contains two integers *x*, *k*. (*k* <= the total node number in the subtree of *x*)

### Output

For each query ($x$, $k$), output the index of the node whose label is the $k$-th largest in the subtree of the node $x$.

### Example

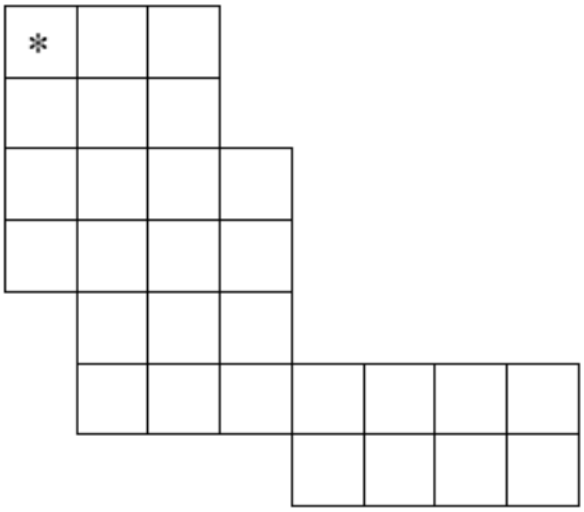
```
Input:
5
1 3 5 2 7
1 2
2 3
1 4
3 5
4
2 3
4 1
3 2
3 2

Output:
5
4
5
5
```

# QUESTION – 4

A knight is a piece used in the game of chess. The chessboard itself is square array of cells. Each time a knight moves, its resulting position is two rows and one column, or two columns and one row away from its starting position. Thus a knight starting on row r, column c – which we'll denote as (r,c) – can move to any of the squares (r-2,c-1), (r-2,c+1), (r-1,c-2), (r-1,c+2), (r+1,c-2), (r+1,c+2), (r+2,c-1), or (r+2,c+1). Of course, the knight may not move to any square that is not on the board.

Suppose the chessboard is not square, but instead has rows with variable numbers of columns, and with each row offset zero or more columns to the right of the row above it. The figure to the left illustrates one possible configuration. How many of the squares in such a modified chessboard can a knight, starting in the upper left square (marked with an asterisk), not reach in any number of moves without resting in any square more than once?



If necessary, the knight is permitted to pass over regions that are outside the borders of the modified chessboard, but as usual, it can only move to squares that are within the borders of the board.

## Input

There will be multiple cases to consider. The input for each case begins with an integer n, between 1 and 10, that specifies the number of rows in the modified chessboard. Following n there will be n pairs of integers, with the ith pair corresponding to the ith row of the chessboard. The first integer of each pair indicates the number of squares skipped at the beginning of the row. The second integer indicates the number of squares in the row (which will always be at least 1).The last case will be followed by the integer 0.

For example, input for the case illustrated by the chessboard shown above would be:

7 0 3 0 3 0 4 0 4 1 3 1 7 4 4

The maximum dimensions of the board will be 10 rows and 10 columns. That is, any modified chessboard specified by the input will fit completely on a 10 row, 10 column board.

## Output

For each input case, display the case number (1, 2, …), and the number of squares that the knight can not reach. Display the results in the format shown in the examples below.

### Example

```
Input:
7 0 3 0 3 0 4 0 4 1 3 1 7 4 4
3 0 3 0 3 0 3
2 0 1 2 1
0

Output:
Case 1, 4 squares can not be reached.
Case 2, 1 square can not be reached.
Case 3, 0 squares can not be reached.
```

# QUESTION – 5

The Bogus Corporation distributes salary to its employees in a weird manner. The salary is distributed every **K** days, and instead of same salary for each day, the salary for the $i^{th}$ day is $a_i$. An ambitious young manager, fresh from *Institute of Mismanagement*, observes that people usually prefer to take leave towards the end of this period of K days, when the workload is higher. Instead of revising each of the $a_i$'s, the manager comes up with a quick fix solution - he redefines the new salary on the $i^{th}$ day as $b_i=a_i+2a_{i-1}+2^2a_{i-2}+2^3a_{i-3}+.....+2^{i-1}a_1$ . Baba, one of the employees, is in a dire financial crisis, and must accumulate at least **N** rupees at the end of the forthcoming period. Being a lazy worker that he is, he is interested in finding out if attending particular days would guarantee him *exactly* **N** rupees at the end of the period. Can you help Baba?

### Input

First line contains a single integer integer **T**, the number of test cases ( 1<=T<=100). Each test case is described on two lines. First line contains two integers, **N** and **K** ( $1<=N<=2^{63}-1$, 1<=K<=50) , the second line contains a space separated list of K integers, the $\mathbf{a_i}$**'s** ( $1<=a_i<=1000$).

### Output

For each test case, output on a single line 1-based indices of the days (separated by a single space) he should attend to ensure a salary of exactly N rupees at the end of the period. The indices should be printed in the sorted order. In case of multiple answers, output any one of them. If there is no answer, print -1.

### Example

```
Input:
2
```

9 3
1 1 2
10 2
2 3
**Output:**
1 3
-1