PROBLEM BASED LEARNING EXPERIMENT 1:

Poisonous Plant Problem:

There are a number of plants in a garden. Each of the plants has been treated with some amount of pesticide. After each day, if any plant has more pesticide than the plant on its left, being weaker than the left one, it dies.

You are given the initial values of the pesticide in each of the plants. Determine the number of days after which no plant dies, i.e. the time after which there is no plant with more pesticide content than the plant to its left.

Example

// pesticide levels

Use a -indexed array. On day , plants and die leaving . On day , plant in dies leaving . There is no plant with a higher concentration of pesticide than the one to its left, so plants stop dying after day .

Function Description

Complete the function poisonousPlants in the editor below.

poisonousPlants has the following parameter(s):

int p[n]: the pesticide levels in each plant

Returns

- int: the number of days until plants no longer die from pesticide

Input Format

The first line contains an integer , the size of the array .

The next line contains space-separated integers .

Constraints

Sample Input

```
7
6 5 8 4 7 10 9
```

Sample Output

```
2
```

Explanation

Initially all plants are alive.

Plants = {(6,1), (5,2), (8,3), (4,4), (7,5), (10,6), (9,7)}

Plants[k] = (i,j) => jth plant has pesticide amount = i.

After the 1st day, 4 plants remain as plants 3, 5, and 6 die.

Plants = {(6,1), (5,2), (4,4), (9,7)}

After the 2nd day, 3 plants survive as plant 7 dies.

Plants = {(6,1), (5,2), (4,4)}

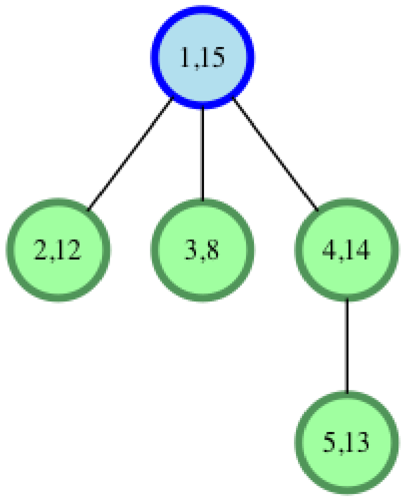
Plants stop dying after the 2nd day.

PROBLEM BASED LEARNING EXPERIMENT 2:

Balanced Forest:

Greg has a tree of nodes containing integer data. He wants to insert a node with some non-zero integer value somewhere into the tree. His goal is to be able to cut two edges and have the values of each of the three new trees sum to the same amount. This is called a *balanced forest*. Being frugal, the data value he inserts should be minimal. Determine the minimal amount that a new node can have to allow creation of a balanced forest. If it's not possible to create a balanced forest, return -1.

For example, you are given node values and . It is the following tree:



The blue node is root, the first number in a node is node number and the second is its value. Cuts can be made between nodes and and nodes and to have three trees with sums , and . Adding a new node of to the third tree completes the solution.

Function Description

Complete the *balancedForest* function in the editor below. It must return an integer representing the minimum value of that can be added to allow creation of a balanced forest, or if it is not possible.

balancedForest has the following parameter(s):

- *c*: an array of integers, the data values for each node
- *edges*: an array of 2 element arrays, the node pairs per edge

Input Format

The first line contains a single integer, , the number of queries.

Each of the following sets of lines is as follows:

- The first line contains an integer, , the number of nodes in the tree.
- The second line contains space-separated integers describing the respective values of , where each denotes the value at node .
- Each of the following lines contains two space-separated integers, and , describing edge connecting nodes and .

Constraints

-
-
-
- Each query forms a valid undirected tree.

Subtasks

For of the maximum score:

-
-

For of the maximum score:

-
-

Output Format

For each query, return the minimum value of the integer . If no such value exists,

return instead.

Sample Input

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

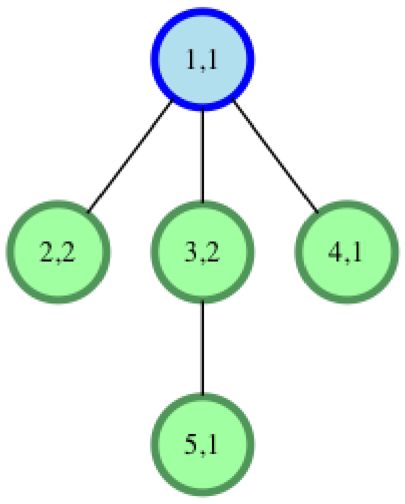
Sample Output

```
2
-1
```

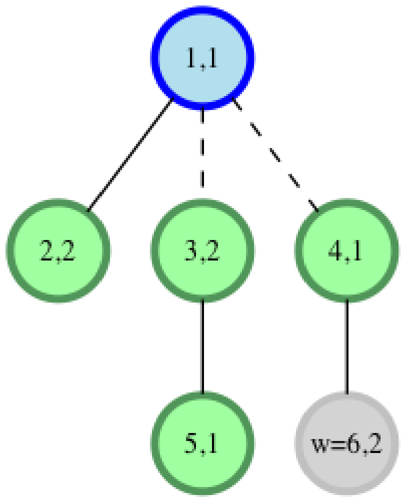
Explanation

We perform the following two queries:

1. The tree initially looks like this:



Greg can add a new node with and create a new edge connecting nodes and . Then he cuts the edge connecting nodes and and the edge connecting nodes and . We now have a three-tree balanced forest where each tree has a sum of .



2. In the second query, it's impossible to add a node in such a way that we can split the tree into a three-tree balanced forest so we return .