Balanced Forest

Greg received an n-node tree as a graduation gift, where each node i contains c_i coins. He wants to insert exactly one new node, w, into the tree using the following process:

- Select a node, v, where $1 \leq v \leq n$.
- Create a new edge connecting node v to a new node, w.
- Add c_w coins to node w (this can be any non-negative integer).

Now that Greg's tree has $n + 1$ nodes, he wants to cut two of its edges to create a forest of 3 trees *where each tree contains an equal number of coins*. If such a configuration is possible, he calls it a *balanced forest*.

For each tree Greg receives as a gift, determine the minimum value of c_w such that the tree can be split into a *balanced forest* (meaning that each of the forest's three trees have the same number of coins); if no c_w exists that enables Greg to create a balanced forest, print -1 instead.

Input Format

The first line contains a single integer, q, denoting the number of trees gifted to Greg. The subsequent lines describe each query in the following format:

- The first line contains an integer, n, denoting the number of nodes in the tree.
- The second line contains n space-separated integers describing the respective values of $c_1, c_2, \ldots, c_n$, where each c_i denotes the number of coins at node i.
- Each line j of the $n - 1$ subsequent lines contains two space-separated integers, x_j and y_j (where $1 \leq x_j, y_j \leq n$), describing edge j connecting nodes x_j and y_j.

Note: It is guaranteed that each query forms a valid undirected tree.

Constraints

- $1 \leq q \leq 5$
- $1 \leq n \leq 5 \times 10^4$
- $1 \leq c_i \leq 10^9$

Subtasks

For 30% of the maximum score:

- $1 \leq n \leq 100$
- $1 \leq c_i \leq 100$

For 50% of the maximum score:

- $1 \leq n \leq 2000$
- $1 \leq c_i \leq 10^9$

Output Format

For each query, print the minimum value of c_w on a new line; if no such value exists, print -1 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
```

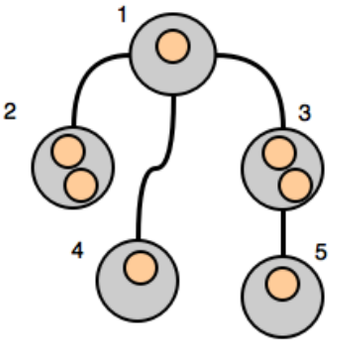
Sample Output

```
2
-1
```

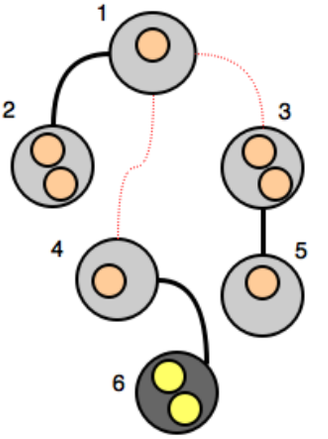
Explanation

We perform the following two queries:

1. The tree initially looks like this:



   Greg can add a new node $w = 6$ with $c_w = 2$ coins and create a new edge connecting nodes 4 and 6. Then he cuts the edge connecting nodes 1 and 4 and the edge connecting nodes 1 and 3. We now have a three-tree forest, where each tree has 3 coins.



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 forest where each tree has an equal number of coins, so we print -1.