January's Color

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Given a rooted tree with n vertices, where the root is at vertex 1. It is guaranteed that no vertex has exactly one child in the tree. In other words, each vertex is either a leaf or has **at least two** children. You own some of the vertices in the tree.

You may have some vertices in your hand. You can obtain new vertices in the following two ways:

- Directly purchase a vertex i from the bank at a cost of c_i.
- Select **two different** vertices from your hand that share the same parent, discard them, and obtain their parent for free.

Now, you have m queries. In each query, you initially have exactly one vertex x, and you want to end up with exactly one vertex y with no other vertices left. Find the minimum cost needed to achieve this, or report no solution.

Input

The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 10^5$), the number of test cases. For each test case:

- The first line contains two integers n and m ($3 \le n \le 3 \cdot 10^5, 1 \le m \le 3 \cdot 10^5$), where n is the number of vertices in the tree and m is the number of queries.
- The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$), which are the costs to obtain each vertex from the bank.
- The next $n-1$ lines each contain two integers u and v ($1 \le u, v \le n, u \ne v$), representing an edge in the tree.
- The next m lines each contain two integers x and y ($1 \le x, y \le n$, $x \ne y$), representing a query.

It is guaranteed that the sum of n and the sum of m over all test cases do not exceed $3 \cdot 10^5$.

Output

For each query, output a single integer representing the minimum additional cost for that query. If there is no way to end up with a single vertex y, output -1.

Example

standard input	standard output
3 5 5 1 2 3 4 5 1 2 1 3 2 4 2 5 3 1 2 1 4 1 5 1 5 2 5 5 1 5 1 1 1 1 2 1 3 2 4 2 5 3 1 2 1 4 1 5 1 2 5 6 5 9 9 8 2 4 4 1 2 1 3 1 4 1 5 1 6 2 1 3 1 4 1 5 1 6 1	2 3 8 7 4 2 1 2 2 -1 2 2 4 2 2

Note

For the first query in the first test case, you initially have vertex 3 and wish to obtain vertex 1. The cheapest solution is to directly purchase vertex 2 at a cost of $c_2 = 2$, then discard vertices 2 and 3 to obtain their common parent 1 for free. It can be proven that this is the minimum-cost solution.

For the first query in the second test case, you again start with vertex 3 and want to obtain vertex 1. Instead of purchasing vertex 2, you choose to purchase vertices 4 and 5 at a total cost of $c_4 + c_5 = 2$, then discard vertices 4 and 5 to obtain their parent vertex 2 for free. Next, discard vertices 2 and 3 to obtain their parent vertex 1 for free. It can be proven that this is the minimum-cost solution.

Note that you can not purchase another vertex 3 and then discard it with the original vertex 3 to obtain vertex 1, since these two vertices 3 do not satisfy the requirement of being two different vertices.