Pondo LCA II

time limit per test: 2 seconds (5 seconds for Python)
memory limit per test: 500 megabytes
input: standard input
output: standard output
points: 100 · problem code: pondolcaii

You are given a tree with n vertices, rooted at vertex 1, and q queries. Each query gives two vertices u and v. Output the lowest common ancestor of u and v.

The lowest common ancestor of u and v is the deepest vertex that lies on both the path from the root to u and the path from the root to v. In particular, $\text{lca}(u, u) = u$, and if u is an ancestor of v then $\text{lca}(u, v) = u$.

Solve the queries with an Euler tour, the same way as in Pondo LCA I: record a vertex every time you visit it, then the lowest common ancestor of u and v is the vertex of minimum depth on the tour between their first visits.

The bounds are larger here, so scanning that range per query is too slow. Use the sparse table below to find the index of the minimum value in a range in $O(1)$ time after $O(n \log n)$ preprocessing. `query(L, R)` is inclusive.

```
class SparseTable:
    def __init__(self, arr):
        self.arr = arr
        self.n = len(arr)
        self.log = [0] * (self.n + 1)
        self._compute_logs()
        self.st = self._build_sparse_table()

    def _compute_logs(self):
        for i in range(2, self.n + 1):
            self.log[i] = self.log[i // 2] + 1

    def _build_sparse_table(self):
        k = self.log[self.n] + 1
        st = [[(0, 0)] * k for _ in range(self.n)]
        for i in range(self.n):
            st[i][0] = (self.arr[i], i)
        j = 1
        while (1 << j) <= self.n:
            i = 0
            while i + (1 << j) - 1 < self.n:
                if st[i][j - 1][0] < st[i + (1 << (j - 1))][j - 1][0]:
                    st[i][j] = st[i][j - 1]
                else:
                    st[i][j] = st[i + (1 << (j - 1))][j - 1]
                i += 1
            j += 1
        return st

    def query(self, L, R):
        j = self.log[R - L + 1]
        left = self.st[L][j]
        right = self.st[R - (1 << j) + 1][j]
        if left[0] < right[0]:
            return left
        return right
```

Input

The first line contains two integers n and q.

Each of the next $n - 1$ lines contains two integers u and v, denoting an undirected edge between u and v.

Each of the next q lines contains two integers u and v, the vertices of one query.

Vertices are numbered 1 through n. The edges form a tree.

Output

Print q lines. The i-th line should contain the lowest common ancestor of the i-th query.

Constraints

- $1 \leq n, q \leq 10^5$
- $1 \leq u, v \leq n$

Example 1

Input

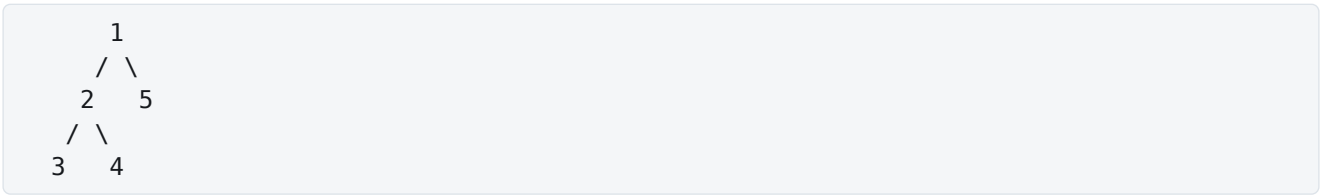
```
5 5
1 2
2 3
2 4
1 5
2 5
5 2
3 4
2 3
4 4
```

Output

```
1
1
2
2
4
```

Explanation

The tree, rooted at 1, looks like this:



- $\text{lca}(2, 5) = 1$
- $\text{lca}(5, 2) = 1$
- $\text{lca}(3, 4) = 2$
- $\text{lca}(2, 3) = 2$
- $\text{lca}(4, 4) = 4$

Example 2

Input

```
10 10
4 7
7 8
2 4
6 7
8 10
5 8
2 9
3 7
1 9
2 9
6 8
8 9
2 8
8 9
3 8
4 10
5 7
5 6
3 9
```

Output

```
9
7
9
2
9
7
4
7
7
9
```