9. LCA 2

문제

N(2 ≤ N ≤ 100,000)개의 정점으로 이루어진 트리가 주어진다.
트리의 각 정점은 1번부터 N번까지 번호가 매겨져 있으며, 루트는 1번이다.

두 노드의 쌍 M(1 ≤ M ≤ 100,000)개가 주어졌을 때,
두 노드의 가장 가까운 공통 조상이 몇 번인지 출력한다.

입력

첫째 줄에 노드의 개수 N이 주어지고, 다음 N-1개 줄에는 트리 상에서 연결된 두 정점이 주어진다. 그 다음 줄에는 가장 가까운 공통 조상을 알고싶은 쌍의 개수 M이 주어지고, 다음 M개 줄에는 정점 쌍이 주어진다.

출력

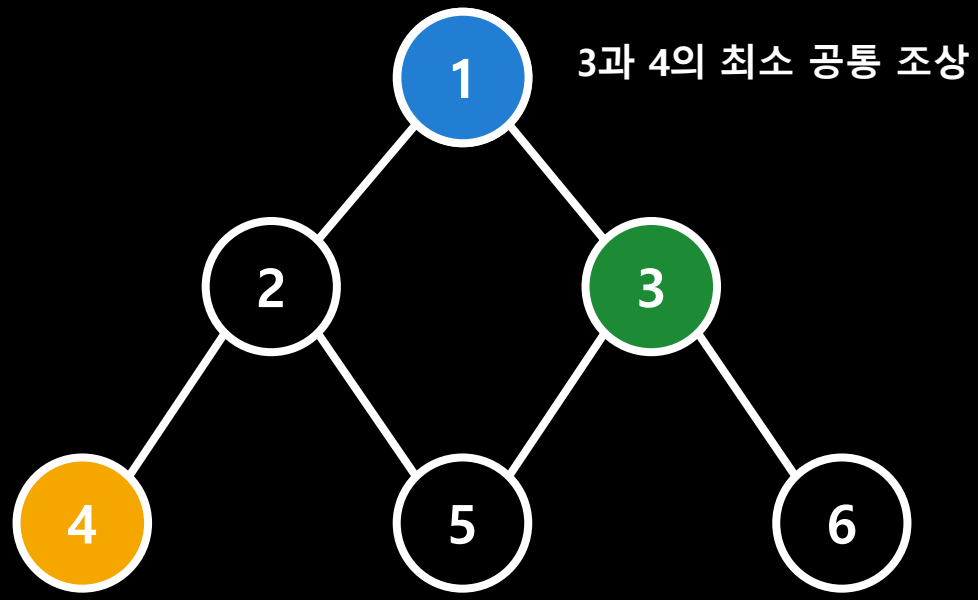
M개의 줄에 차례대로 입력받은 두 정점의 가장 가까운 공통 조상을 출력한다.

[예제 입력 1]
```
15
1 2
1 3
2 4
3 7
6 2
3 8
4 9
2 5
5 11
7 13
10 4
11 15
12 5
14 7
6
6 11
10 9
2 6
7 6
8 13
8 15
```

[예제 출력 1]
```
2
4
2
1
3
1
```

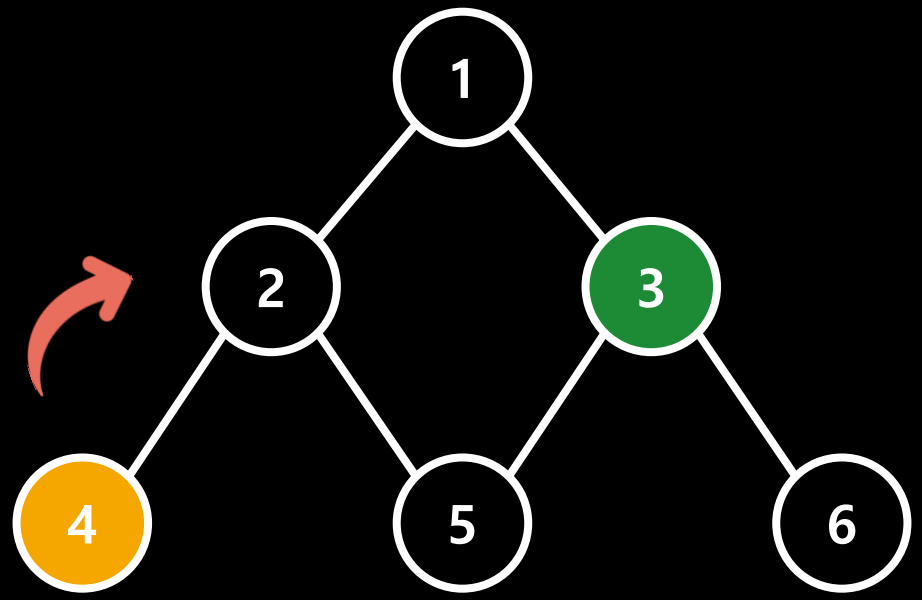
9. LCA 2

LCA = Lowest Common Ancestor(최소 공통 조상)



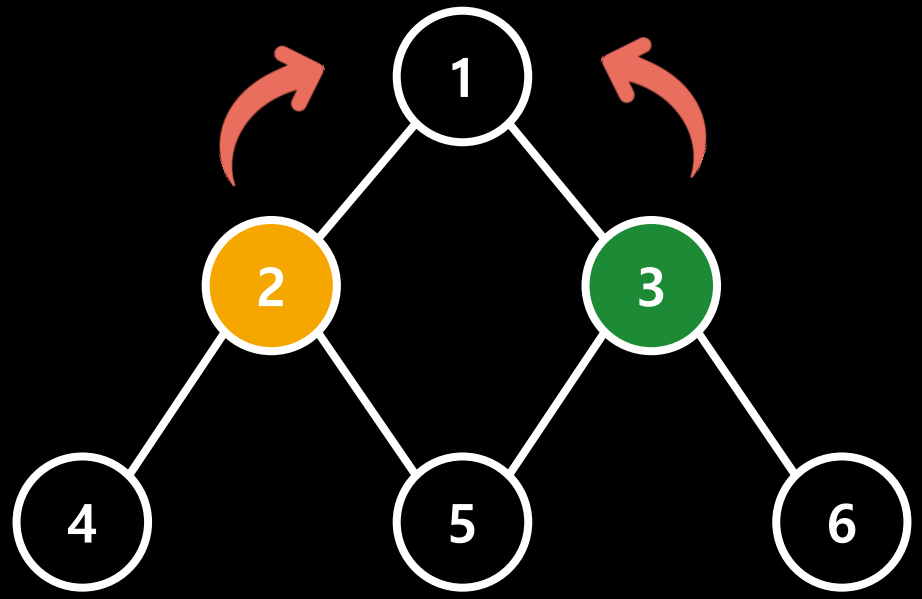
9. LCA 2

1. 두 노드의 레벨(깊이) 맞추기



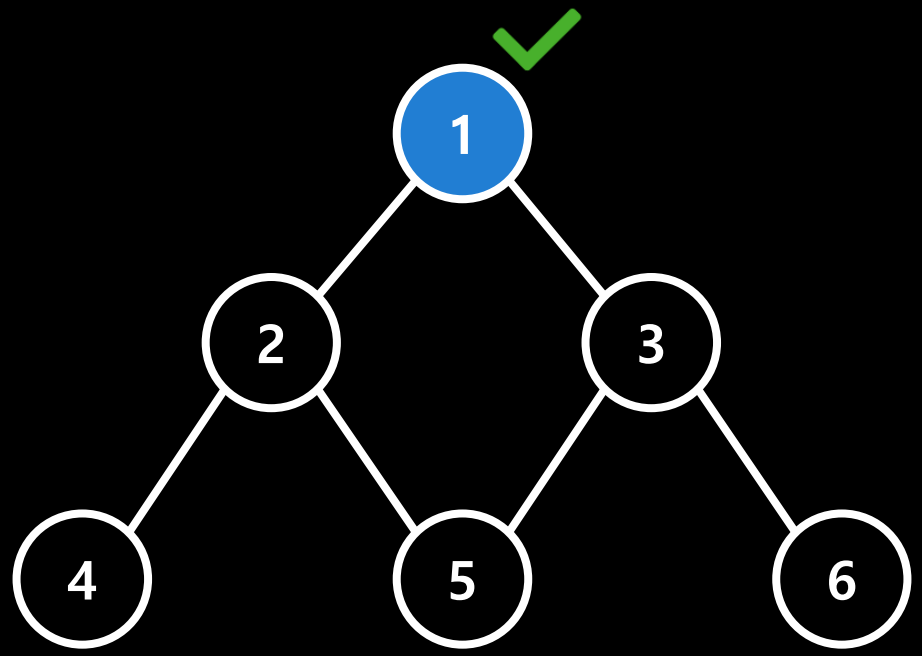
9. LCA 2

2. 레벨을 맞춘 후, 동시에 한 단계씩 트리 구조를 따라 거슬러 올라가기



9. LCA 2

3. 같은 노드를 가리킬 경우, 최소 공통 조상을 찾은 것(값 리턴 후 종료)



9. LCA 2

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>

using namespace std;
typedef pair<int, int> p;

int n, m;
vector<int> adj[100001];
int parent[100001][18]; //parent[i][j] : i번 노드의 2^j번째 조상
int level[100001], maxlevel;

void set_tree(int node, int pnode, int lv) {
    level[node] = lv;
    parent[node][0] = pnode;

    for (int i = 1; i <= maxlevel; i++) {
        parent[node][i] = parent[parent[node][i - 1]][i - 1];
    }

    for (int i = 0; i < adj[node].size(); i++) {
        int childnode = adj[node][i];
        if (childnode == pnode) continue;
        set_tree(childnode, node, lv + 1);
    }
}
```

9. LCA 2

```cpp
int LCA(int a, int b) {
    // a, b의 LCA를 찾아 반환
    if (a == 1 || b == 1) return 1;

    // a, b중 level이 더 높은 노드에 대해, 2^k번째 조상 노드를 찾아 트리의 높이를 올라감
    int target = a, compare = b;

    // a < b인 경우 target = compare, compare = target으로 값 전환
    if (level[a] < level[b]) swap(target, compare);

    // 두 노드의 level이 같아지도록 조정
    if (level[target] != level[compare]) {
        for (int i = maxlevel; i >= 0; i--) {
            if (level[parent[target][i]] >= level[compare])
                target = parent[target][i];
        }
    }

    // 동일 level로 맞춘 후 공통 조상을 찾는다
    int ret = target;
    if (target != compare) {
        for (int i = maxlevel; i >= 0; i--) {
            if (parent[target][i] != parent[compare][i]) {
                target = parent[target][i];
                compare = parent[compare][i];
            }
            ret = parent[target][i];
        }
    }

    return ret;
}
```

```cpp
// 그래프 초기화
void init() {
    int p, c;
    cin >> n;

    for (int i = 0; i < n - 1; i++) {
        cin >> p >> c;
        adj[p].push_back(c);
        adj[c].push_back(p);
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);

    init();
    // 트리의 최대 레벨(높이) 계산(2^k)
    maxlevel = (int)floor(log2(100001));

    set_tree(1, 0, 1);

    cin >> m;
    int first = 0, second = 0;
    for (int i = 0; i < m; i++) {
        cin >> first >> second;
        printf("%d\n", LCA(first, second));
    }

    return 0;
}
```