

트리 2

최백준 choi@startlink.io

LCA

LCA

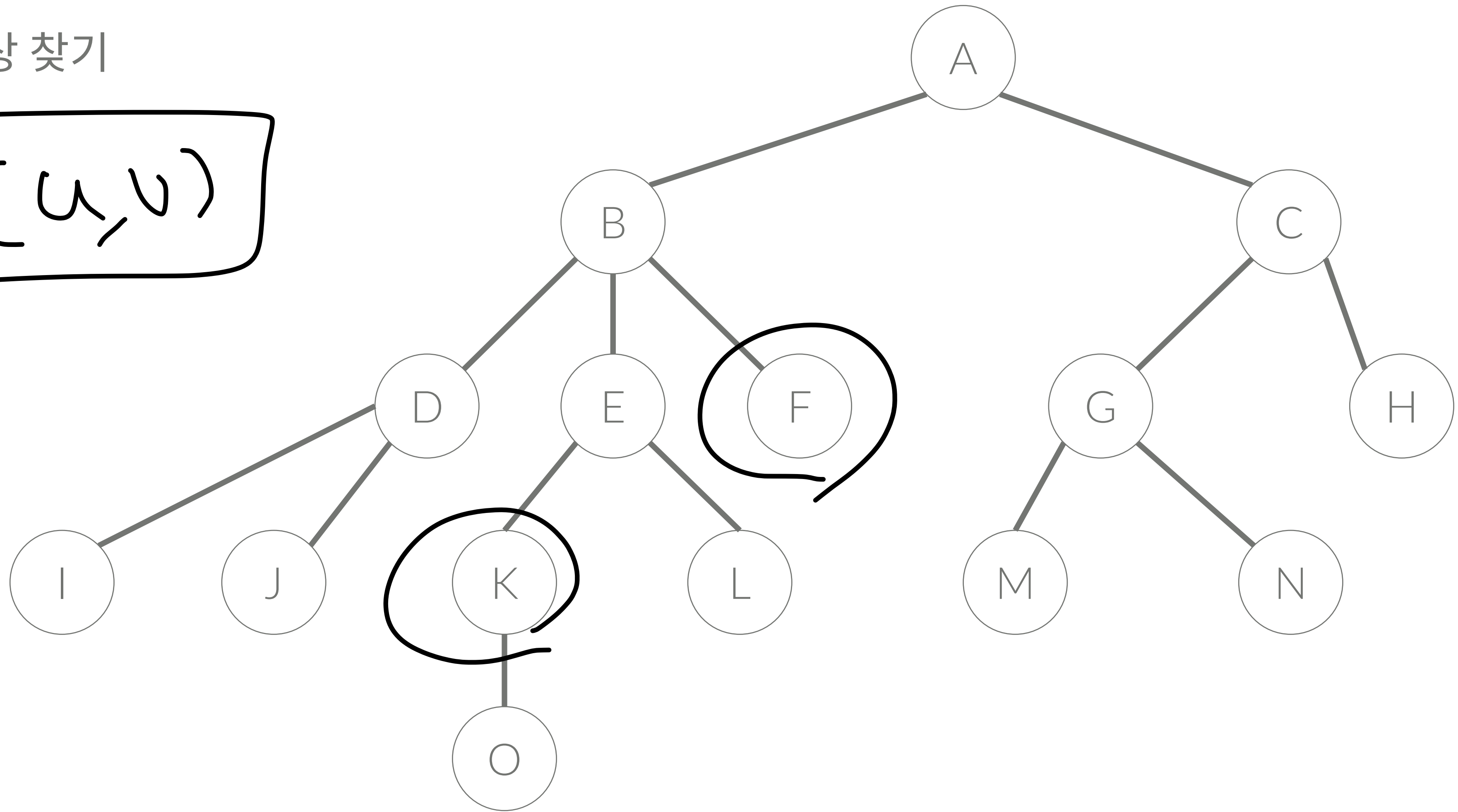
가장 가까운 공통 조상 찾기

3

LCA (Lowest Common Ancestor)

- 가장 가까운 조상 찾기

$LCA(u, v)$

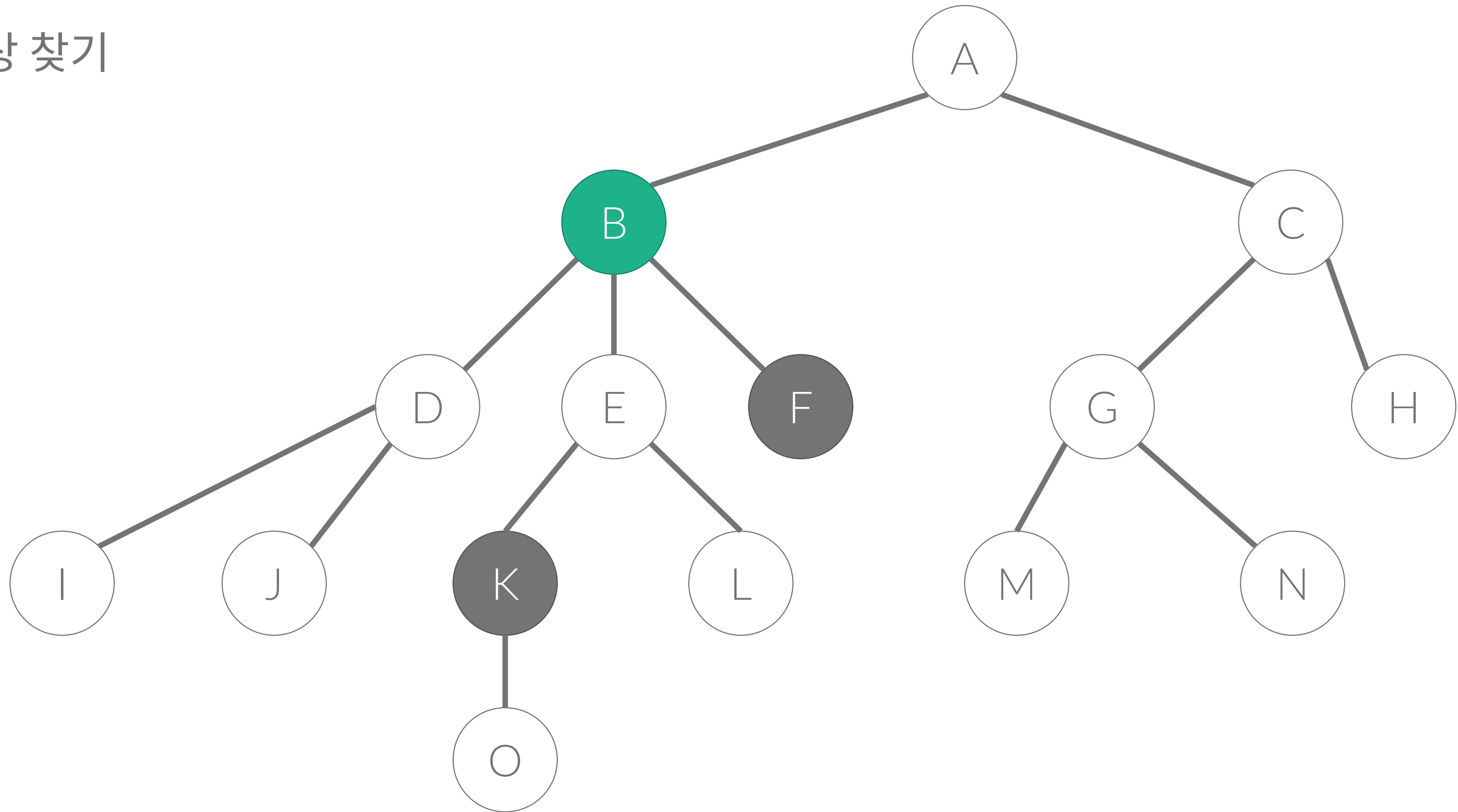


가장 가까운 공통 조상 찾기

4

LCA (Lowest Common Ancestor)

- 가장 가까운 조상 찾기

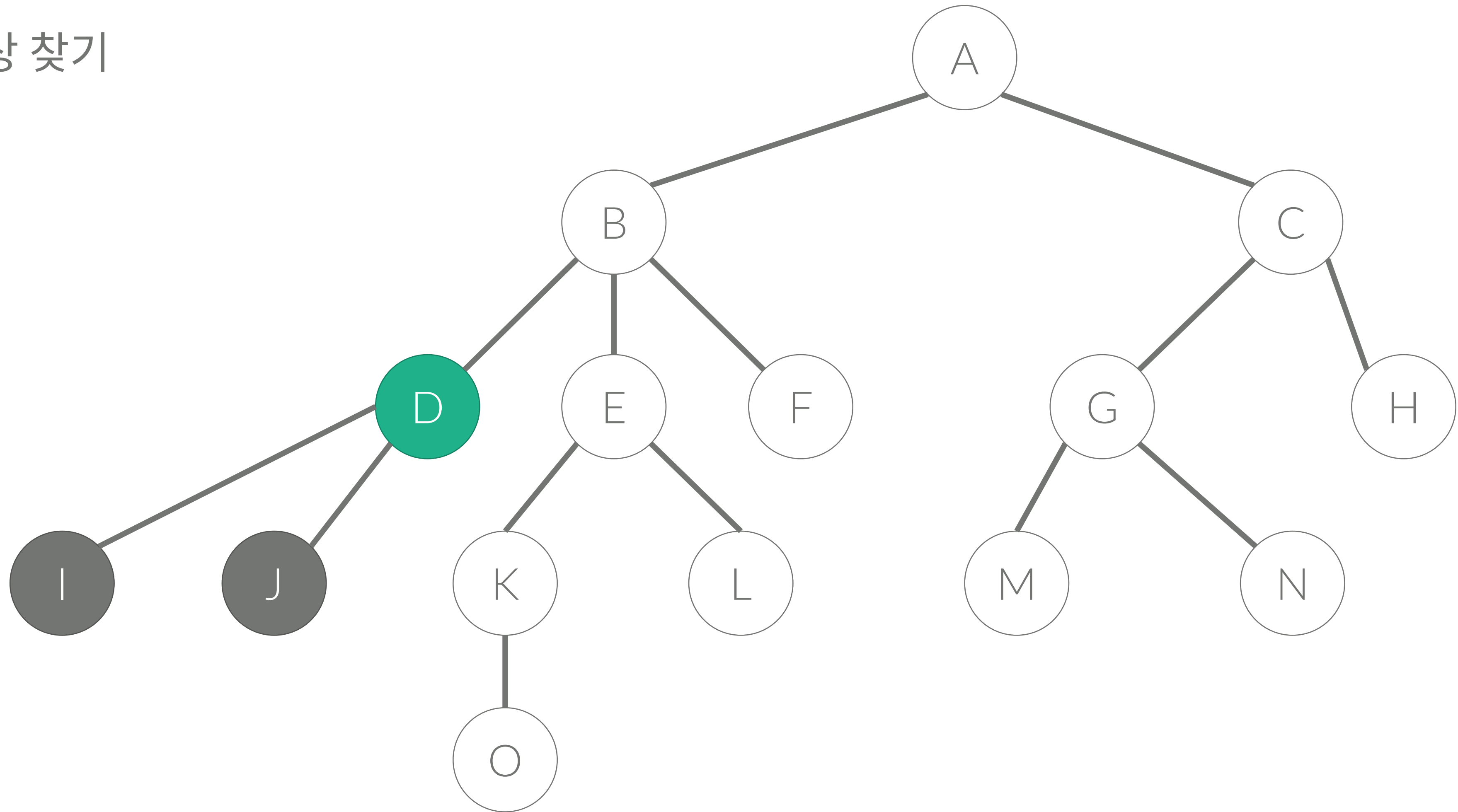


가장 가까운 공통 조상 찾기

5

LCA (Lowest Common Ancestor)

- 가장 가까운 조상 찾기

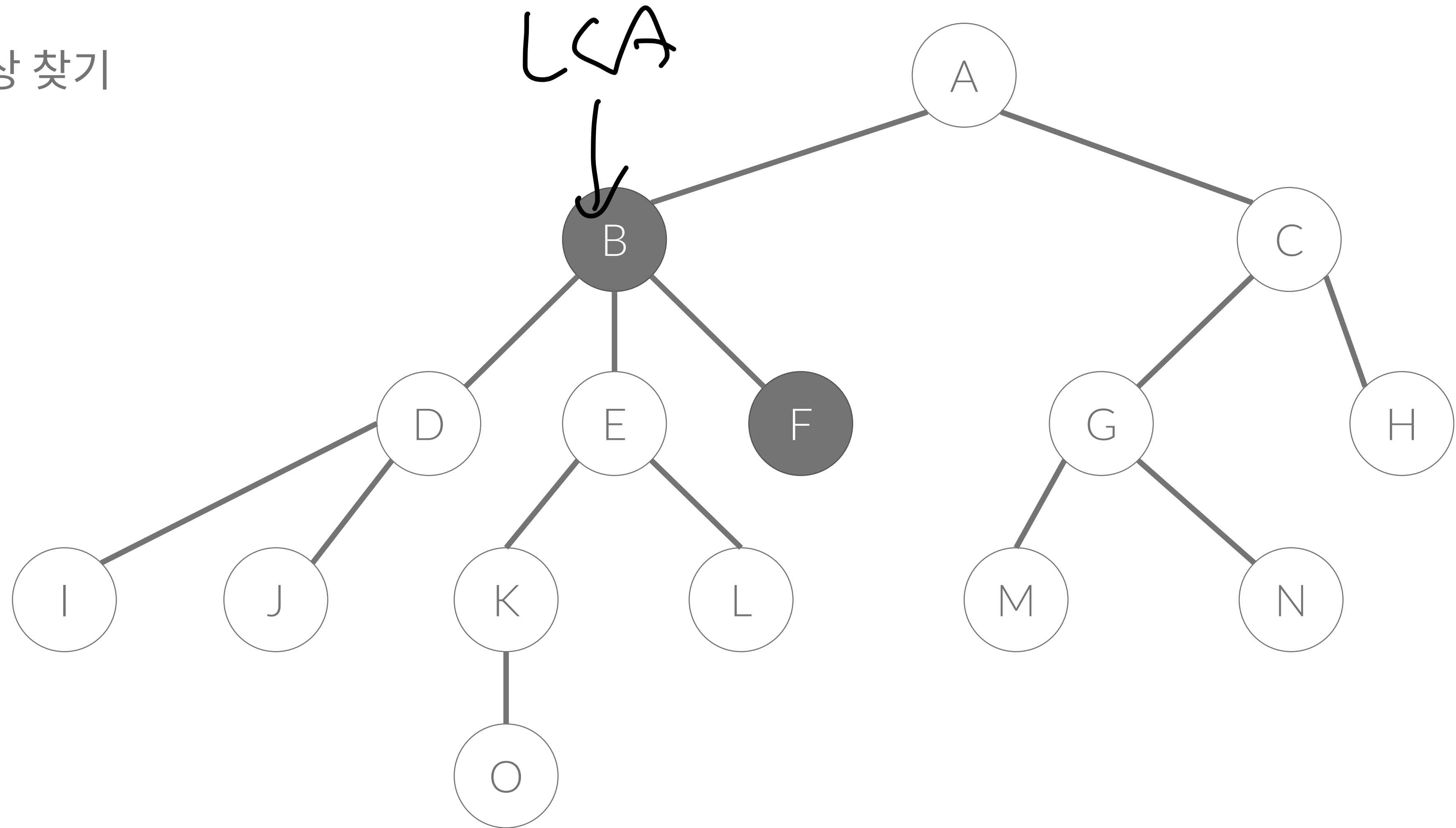


가장 가까운 공통 조상 찾기

6

LCA (Lowest Common Ancestor)

- 가장 가까운 조상 찾기

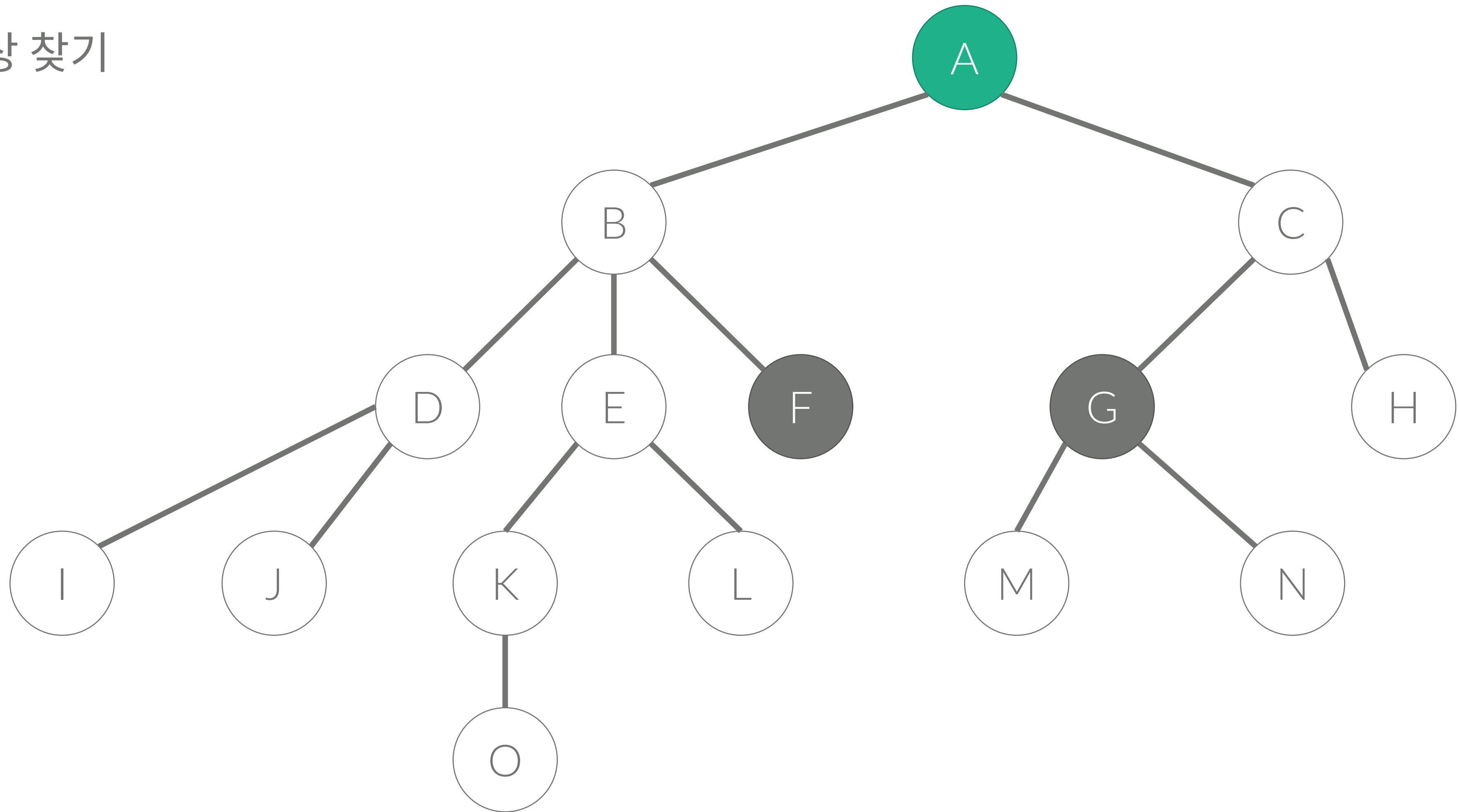


가장 가까운 공통 조상 찾기

7

LCA (Lowest Common Ancestor)

- 가장 가까운 조상 찾기

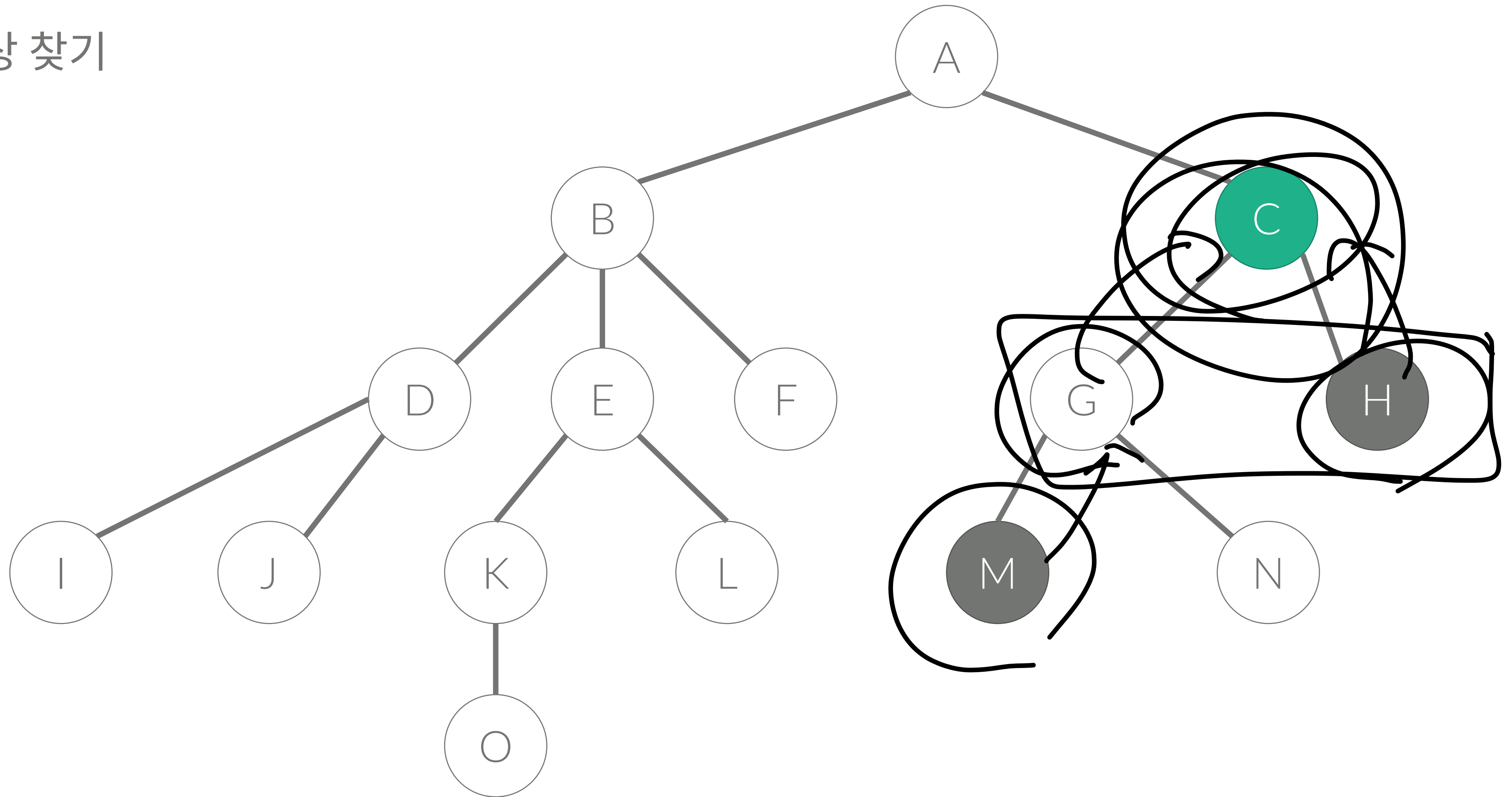


가장 가까운 공통 조상 찾기

8

LCA (Lowest Common Ancestor)

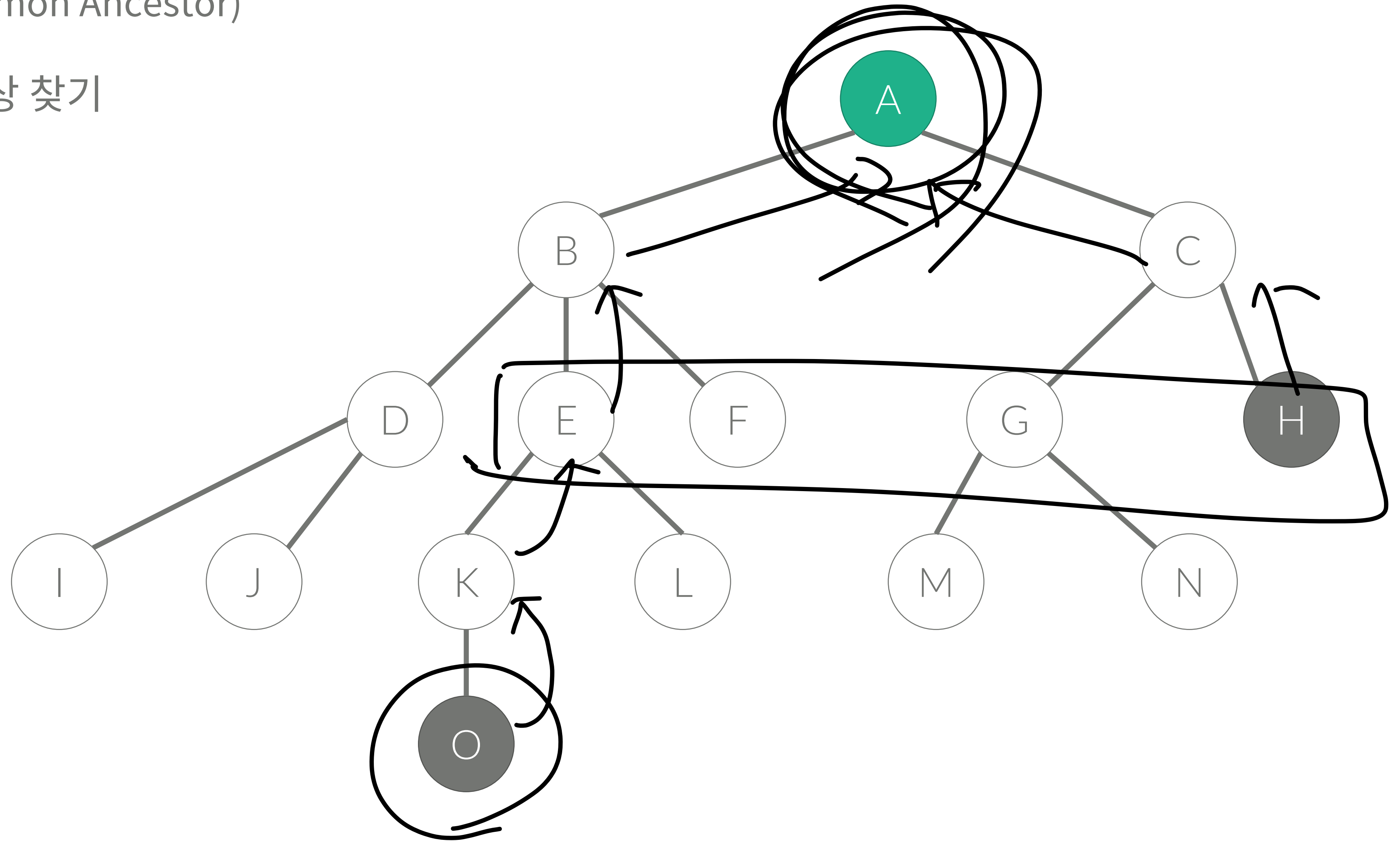
- 가장 가까운 조상 찾기



가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

- 가장 가까운 조상 찾기



가장 가까운 공통 조상 찾기

10

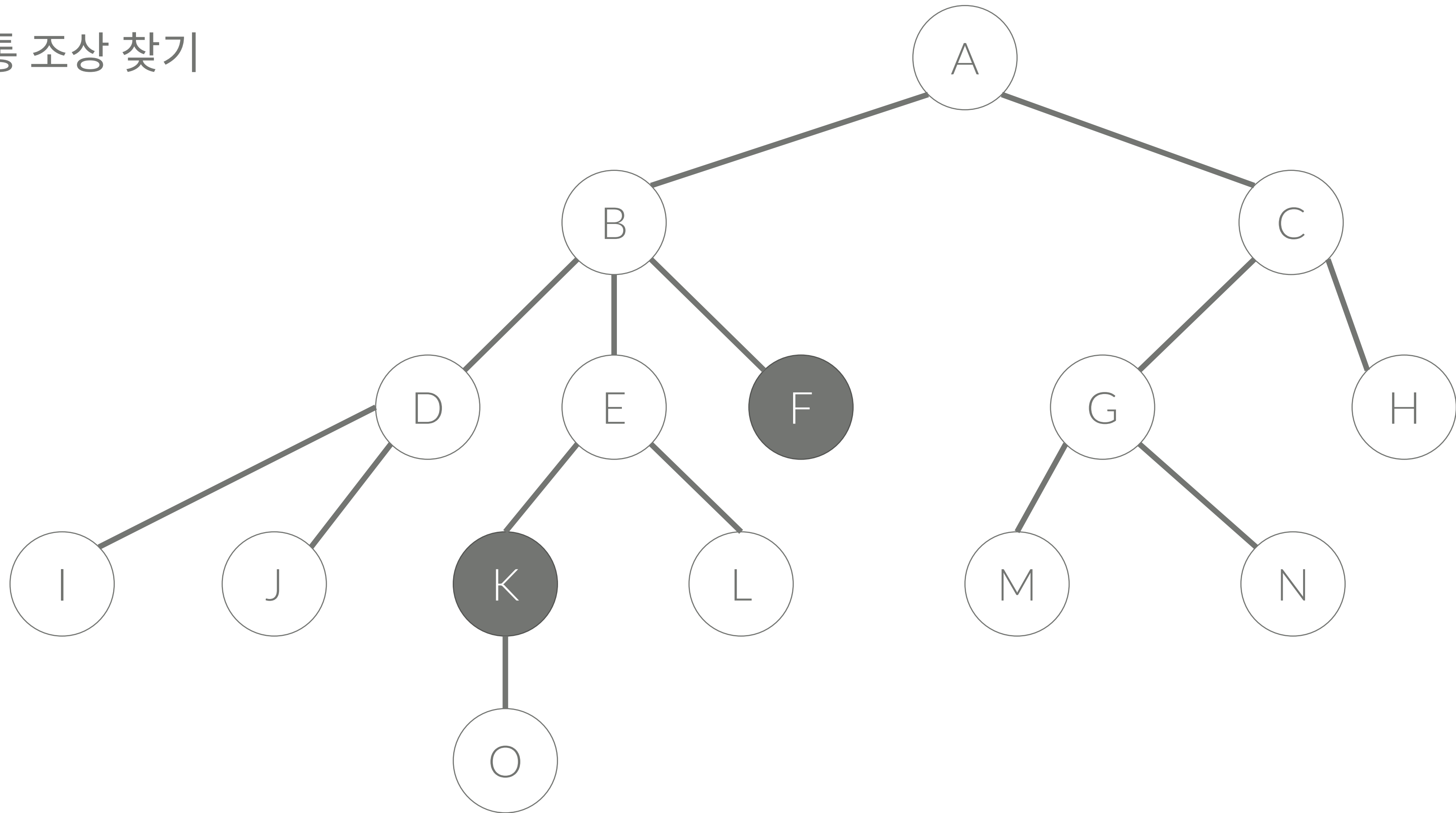
LCA (Lowest Common Ancestor)

- 두 노드 x 와 y 의 LCA 구하기
- 두 노드의 레벨이 다르면
- 레벨이 같을 때까지 레벨이 큰 것을 한 칸 씩 위로 올린다
- 두 노드의 레벨이 같아졌으면
- 같은 노드가 될 때까지 한 칸씩 위로 올린다.

가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

- 가장 가까운 공통 조상 찾기

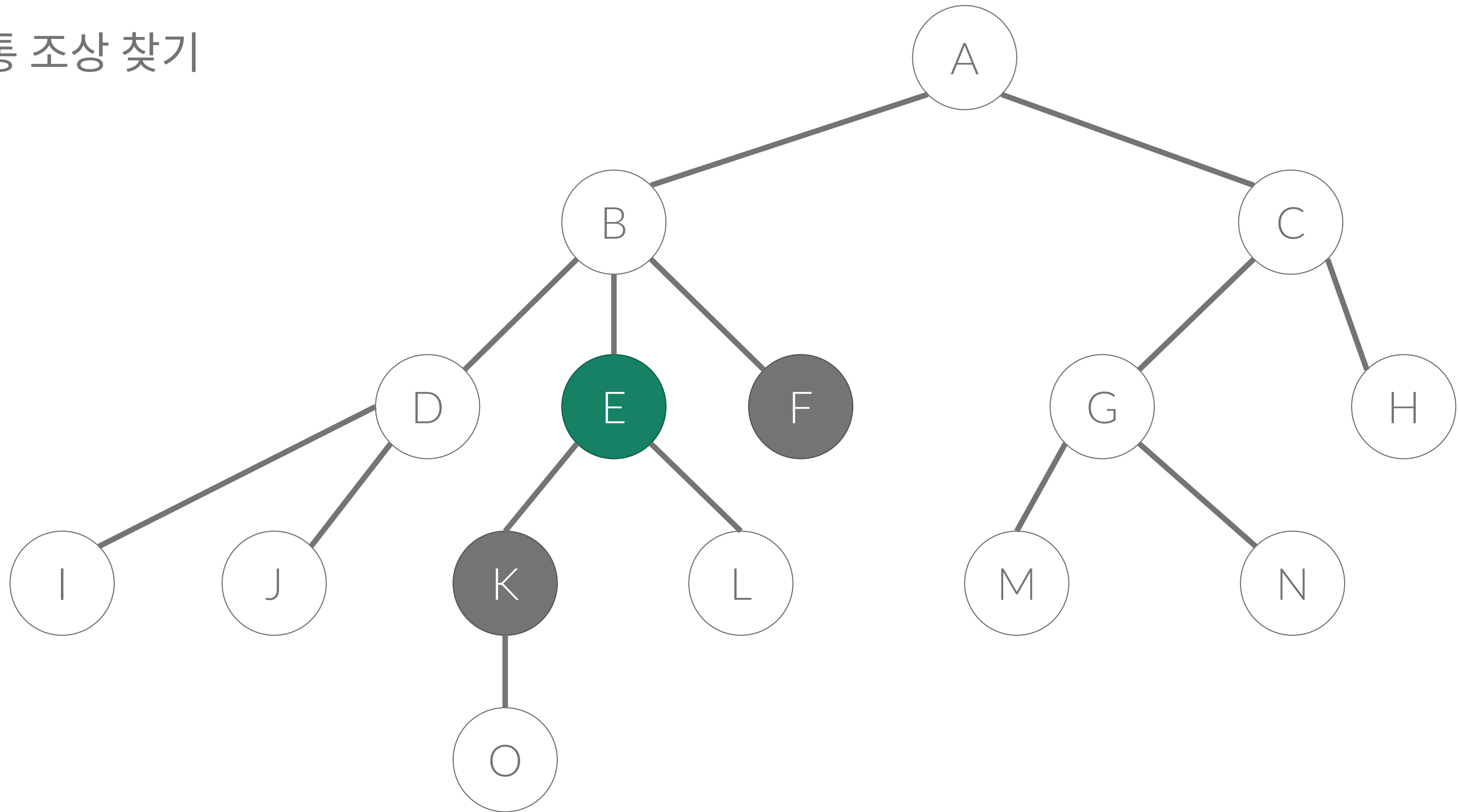


가장 가까운 공통 조상 찾기

12

LCA (Lowest Common Ancestor)

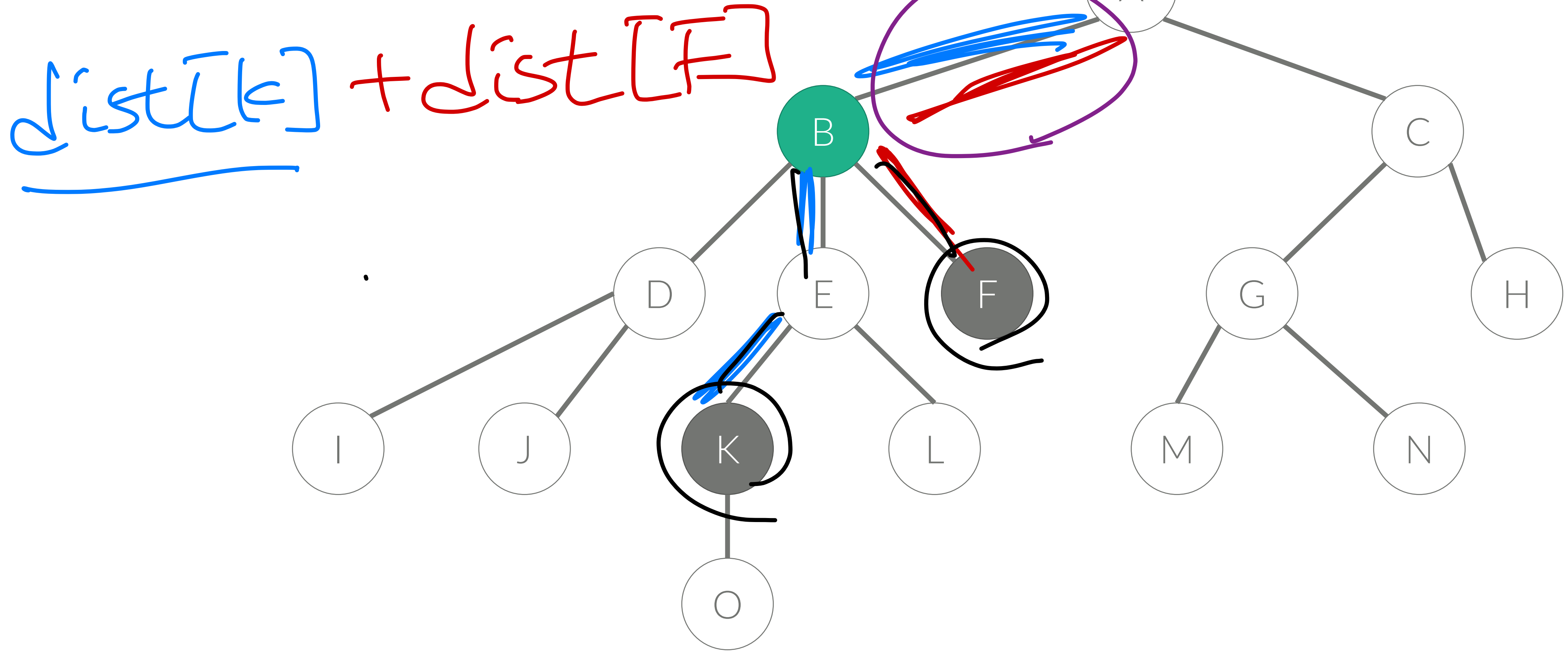
- 가장 가까운 공통 조상 찾기



가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

- ## • 가장 가까운 공통 조상 찾기

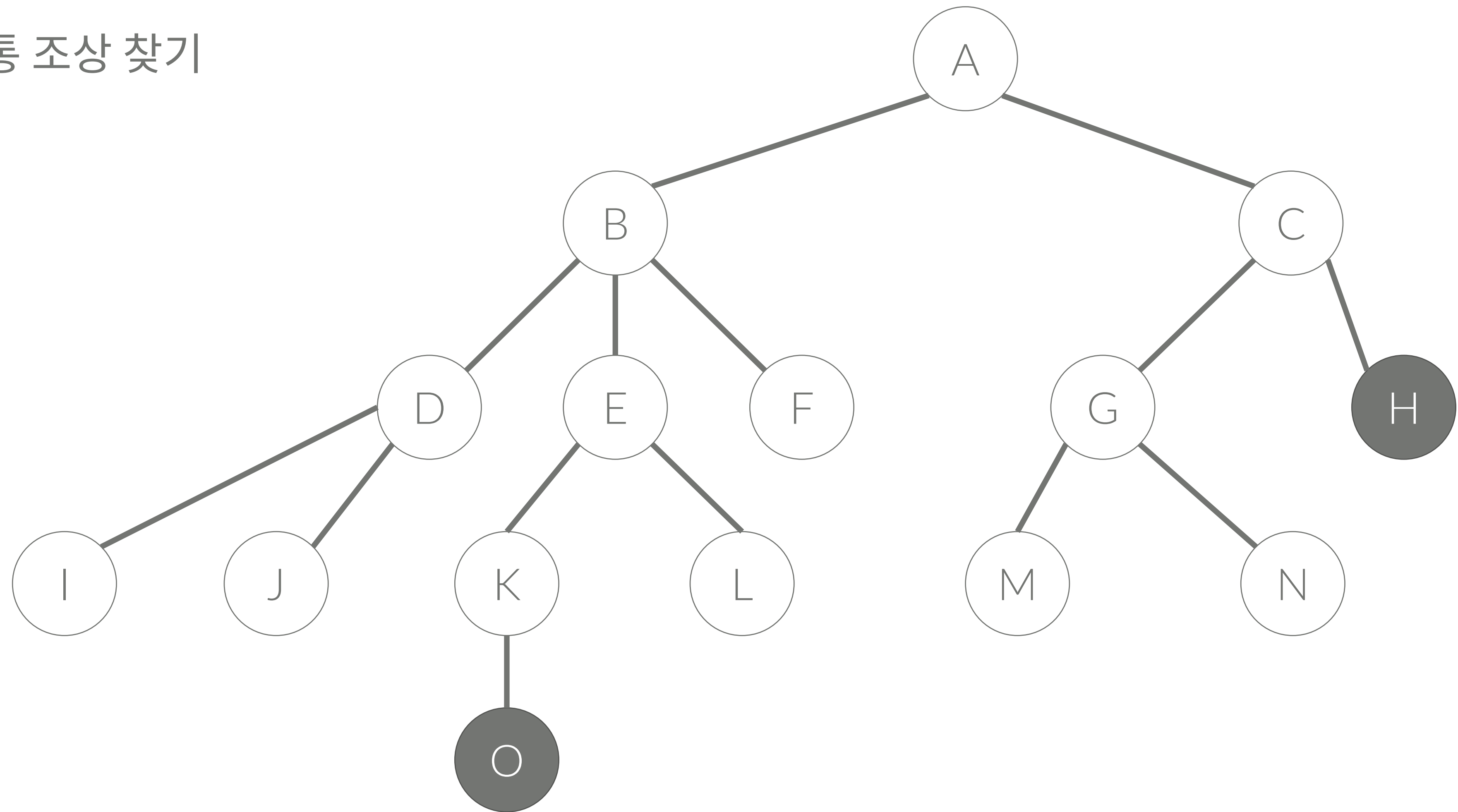


가장 가까운 공통 조상 찾기

14

LCA (Lowest Common Ancestor)

- 가장 가까운 공통 조상 찾기

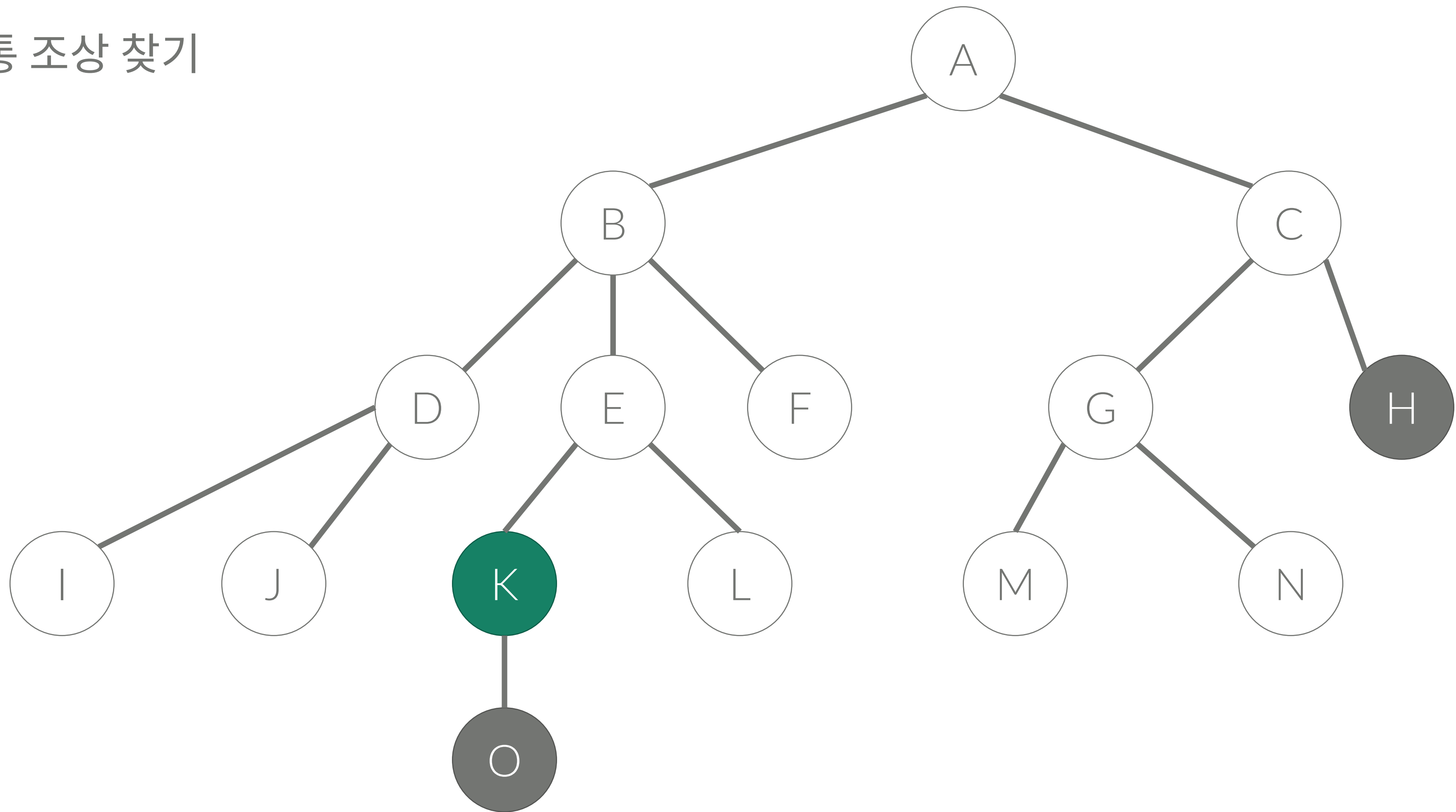


가장 가까운 공통 조상 찾기

15

LCA (Lowest Common Ancestor)

- 가장 가까운 공통 조상 찾기

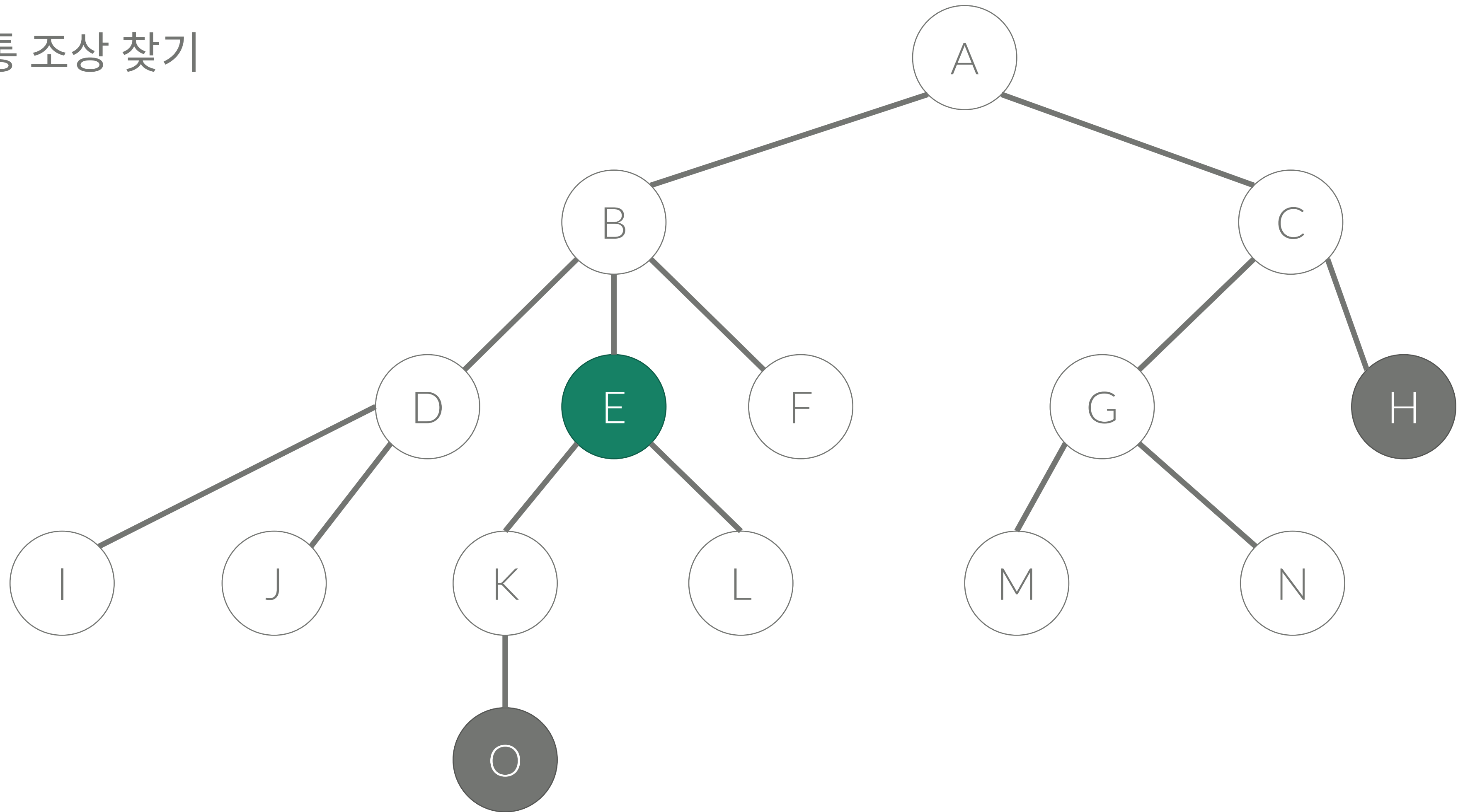


가장 가까운 공통 조상 찾기

16

LCA (Lowest Common Ancestor)

- 가장 가까운 공통 조상 찾기

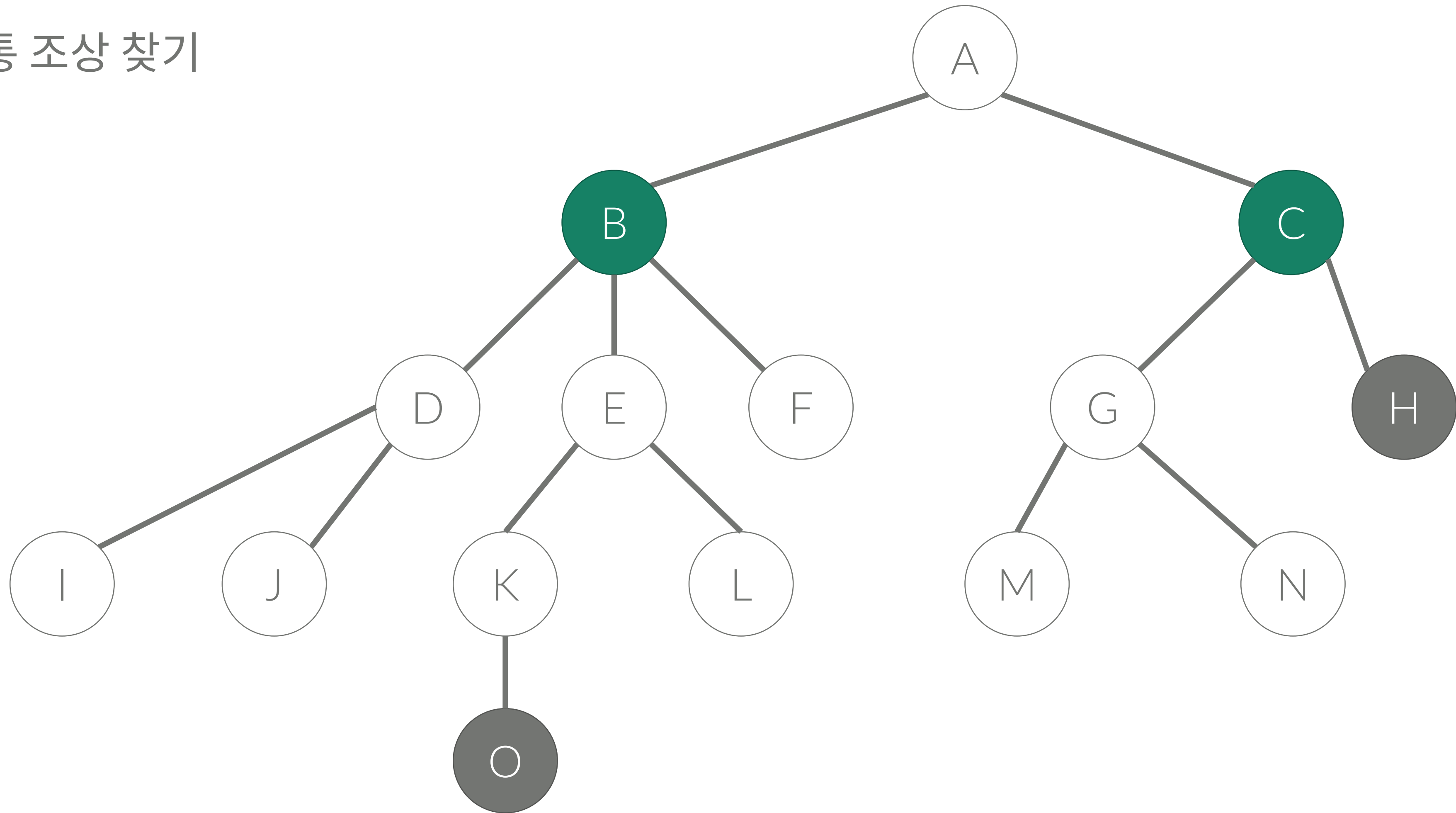


가장 가까운 공통 조상 찾기

17

LCA (Lowest Common Ancestor)

- 가장 가까운 공통 조상 찾기

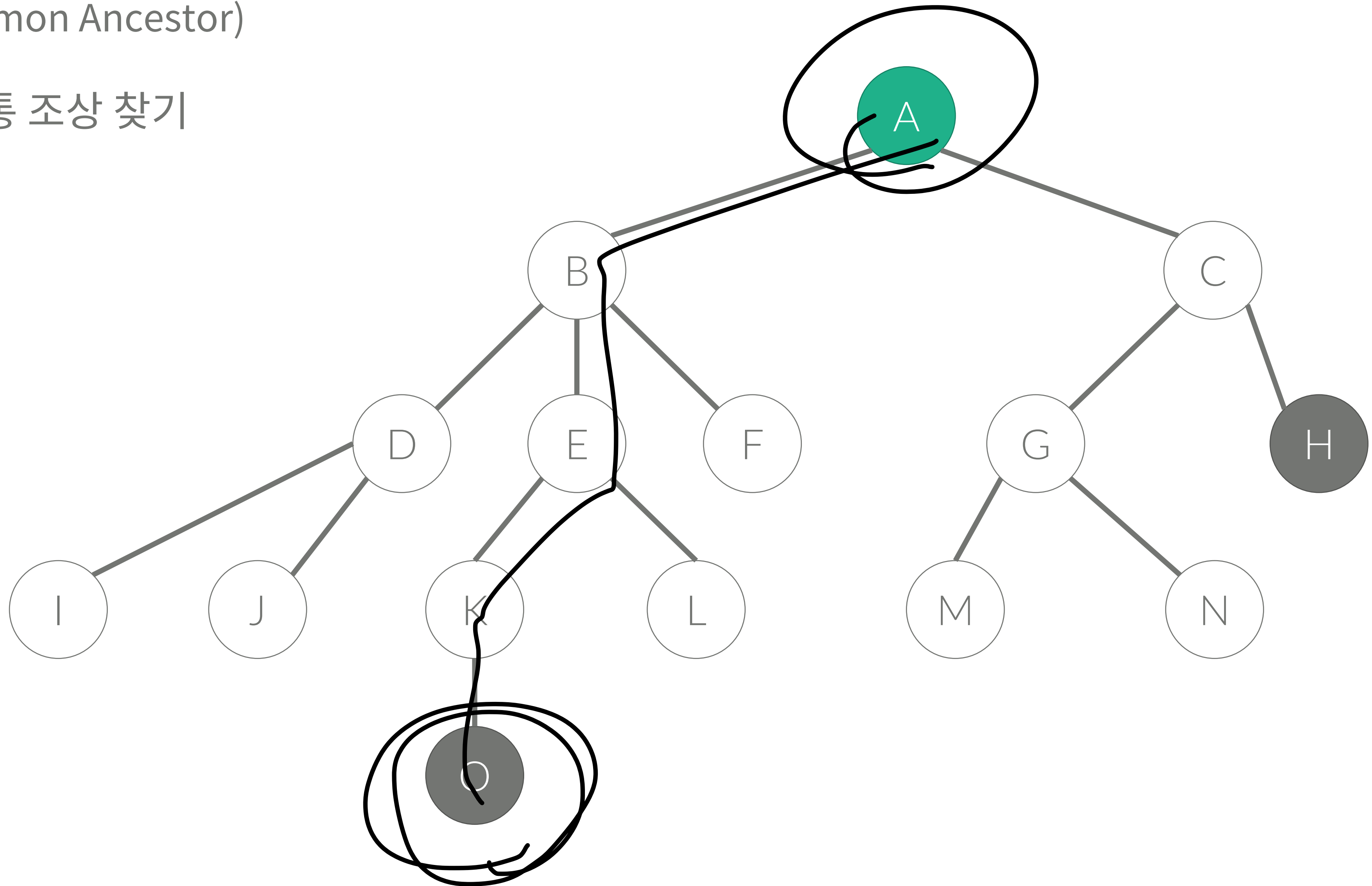


가장 가까운 공통 조상 찾기

18

LCA (Lowest Common Ancestor)

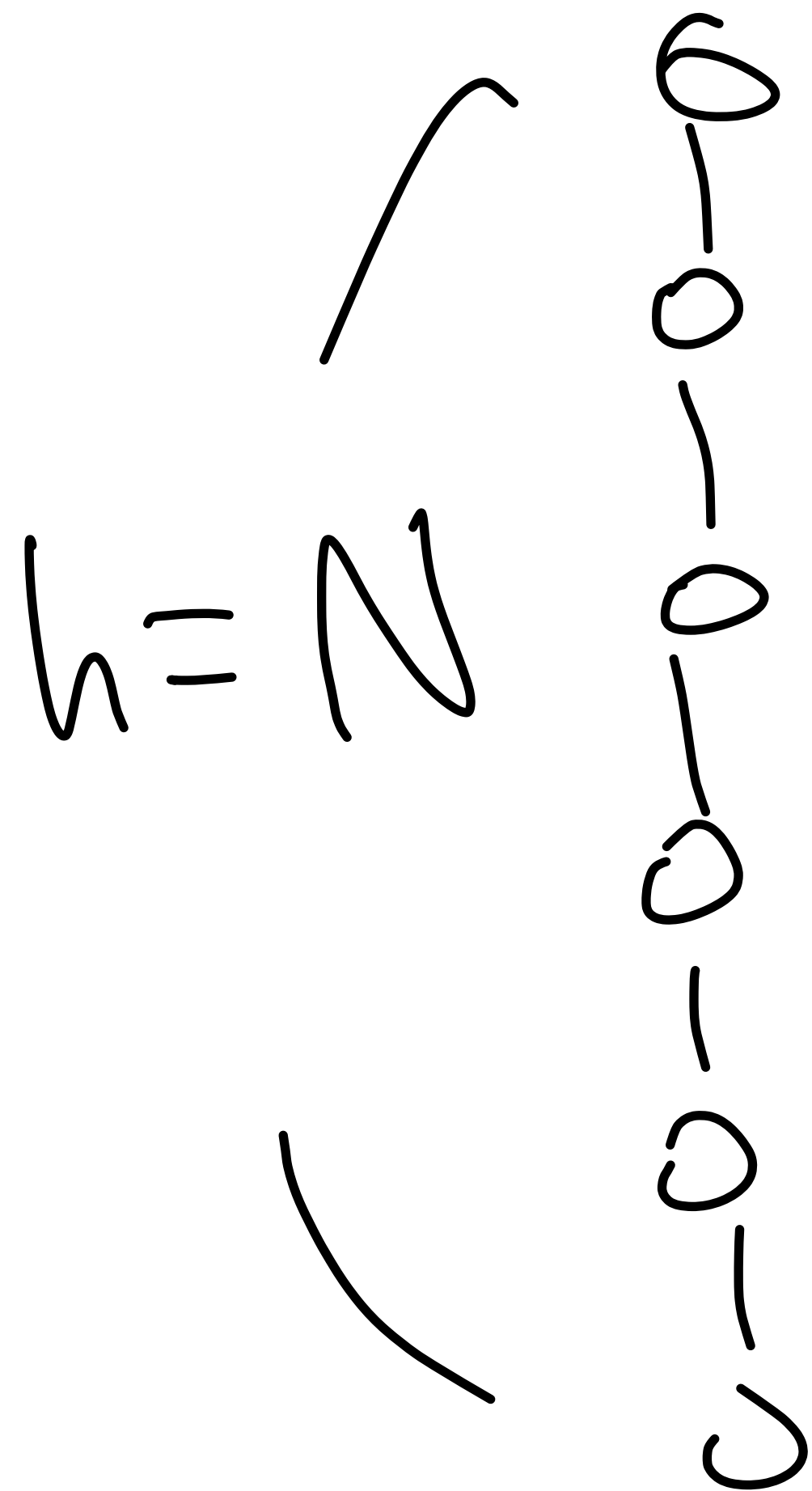
- 가장 가까운 공통 조상 찾기



LCA

<https://www.acmicpc.net/problem/11437>

- 소스: <http://codeplus.codes/377c3995bf874a928bb42d0fca31efc8>



$O(h)$

$O(N)$

$N \times h \leq \sum$

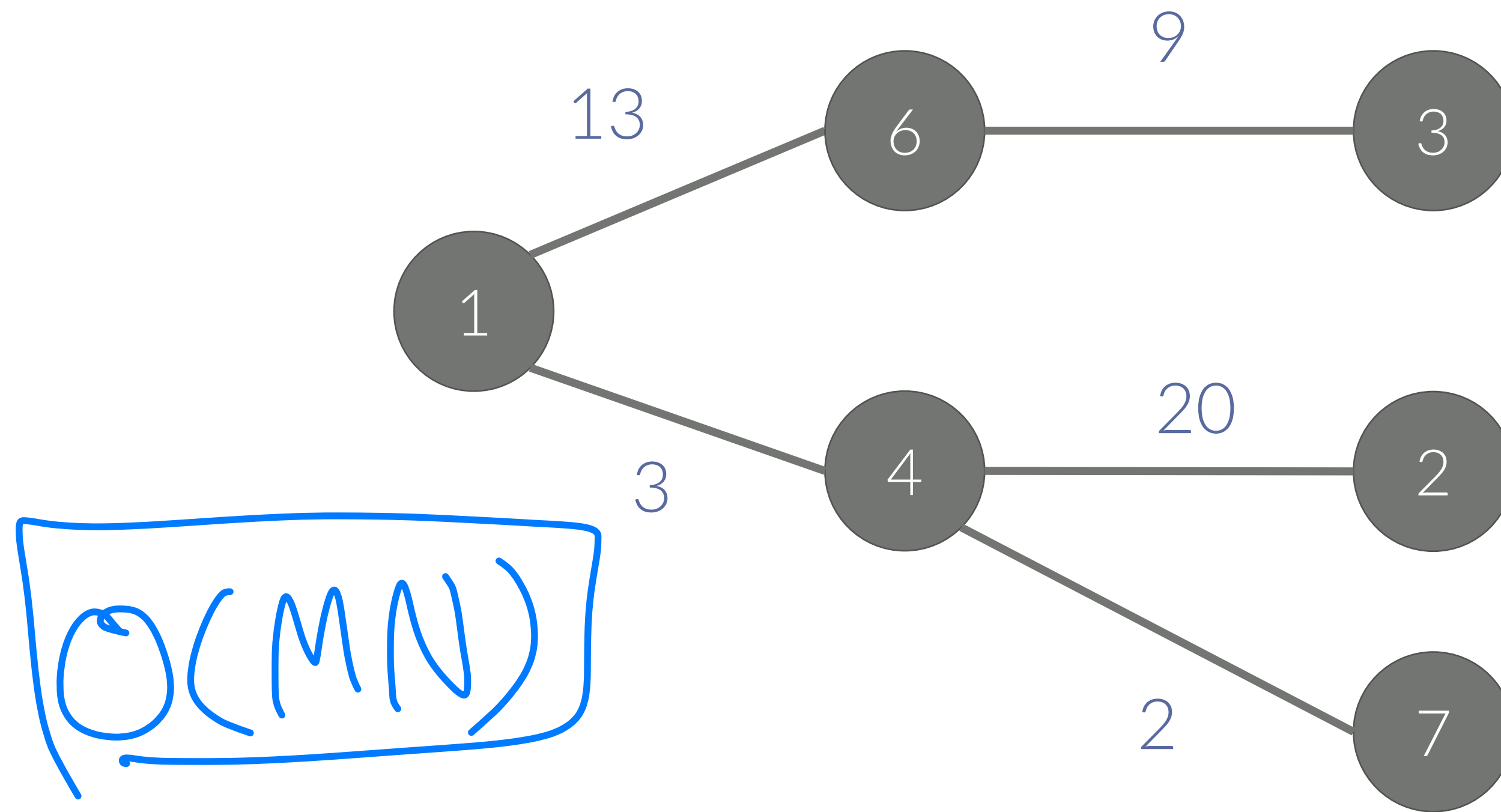
Tree

$h \leq \sqrt{N}$

정점들의 거리

<https://www.acmicpc.net/problem/1761>

- $N(2 \leq N \leq 40,000)$ 개의 정점으로 이루어진 트리가 주어지고 $M(M \leq 10,000)$ 개의 두 노드 쌍을 입력받을 때 두 노드 사이의 거리 구하기



$O(MN)$

Tree: 이진트리 두 정점
사이의 경로가 1개
사이를 찾는 것
DFS/BFS
 $O(N)$
거리 1

정점들의 거리

<https://www.acmicpc.net/problem/1761>

- 트리에서는 모든 정점 쌍 사이의 경로가 1개만 존재하기 때문에
- 어떤 정점 x 에서 y 로 가는 경로가 곧 최단거리가 된다.
- 거리를 구하는데 필요한 시간: $O(N)$
- 총 쿼리의 개수: M 개
- 시간 복잡도: $O(MN)$

정점들의 거리

<https://www.acmicpc.net/problem/1761>

- 또 다른 방법

- $\text{dist}[i]$ = 루트에서 i 까지의 거리

- 라고 정의하면

- u 와 v 사이의 거리는 $\text{dist}[u] + \text{dist}[v] - \text{dist}[\text{lca}(u,v)]$ 로 구할 수 있다

$$Z_7 \rightarrow U \quad Z_7 \rightarrow V \quad LCA$$
 $2x$


GCN

~~BFS/DFS~~
~~list 28/2~~

22

1년 반

두 장의 사진



$O(h)$

A hand-drawn diagram of a circle. Inside the circle, the letter 'C' is on the left and the letter 'N' is on the right. The letters are drawn with thick, black, slightly irregular lines. The circle itself is also drawn with a thick, black, irregular line.

$$O(MN)$$

정점들의 거리

<https://www.acmicpc.net/problem/1761>

- 소스: <http://codeplus.codes/8b40f3ded68648d289dc2b385e77a032>

가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

- 앞의 방법은 최악의 경우에 시간복잡도가 $O(h) = O(N)$ 이다.
- 다이나믹 프로그래밍을 이용하면 $O(\lg N)$ 이다.

가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

• $P[i][j]$ = 노드 i의 2^j 번째 조상

$P[i][1] =$ 노드 i의 $2^0 = 1$ 번째
조상

$P[i][0] = \text{Parent}[i]$ ← 초기화

• $P[i][j] = P[P[i][j-1]][j-1]$

$2^j = 2^{j-1} + 2^{j-1} = 2 \times 2^{j-1}$

• 노드 i의 2^j 번째 조상은 노드 i의 2^{j-1} 번째 조상의 2^{j-1} 번째 조상이다.

$= P[i][j] = P[P[i][j-1]][j-1]$

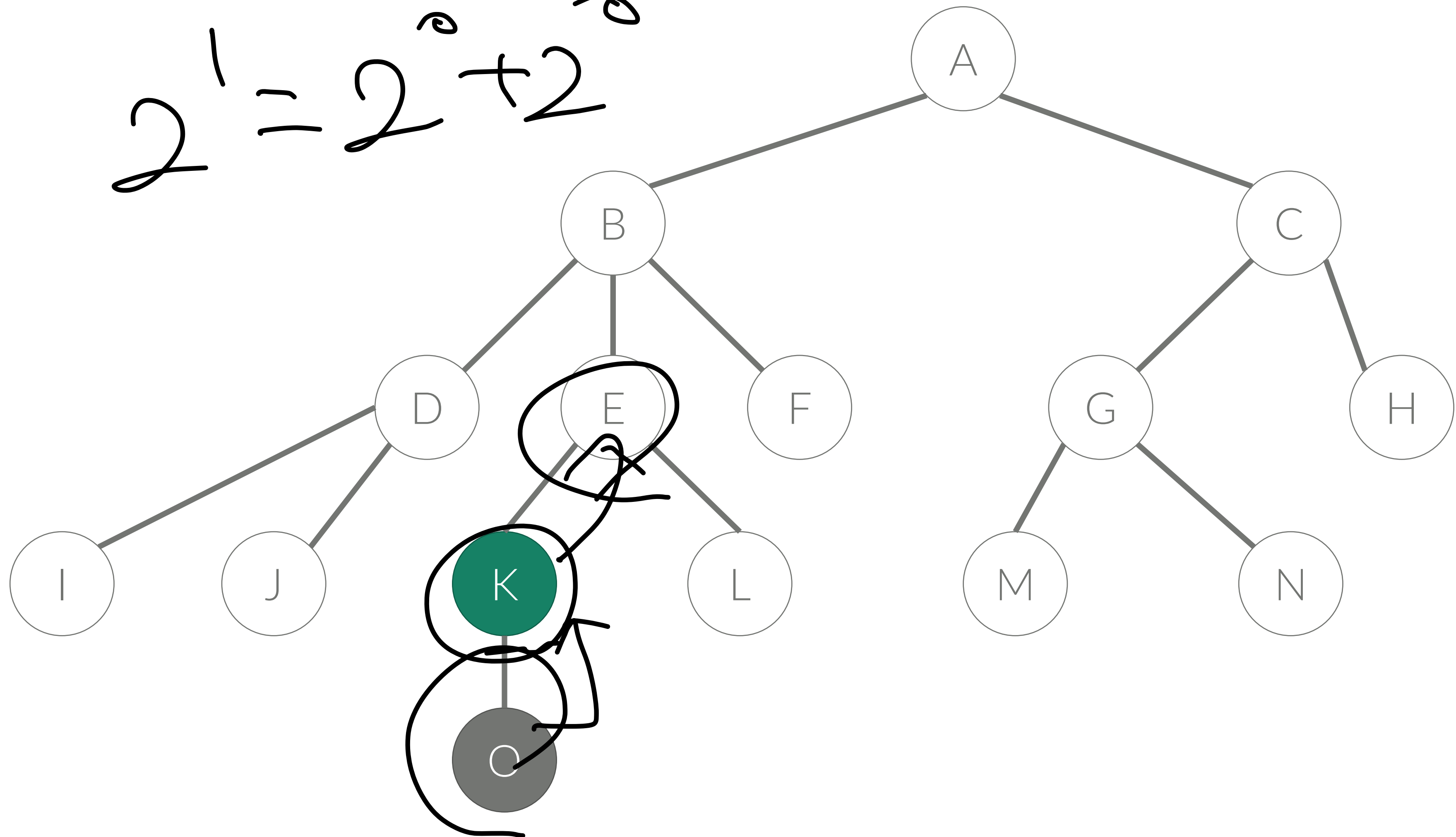
가장 가까운 공통 조상 찾기

26

LCA (Lowest Common Ancestor)

- $j = 0$
- $2^0 = 1$ 번째 조상

$$2^1 = 2^0 + 2^0$$



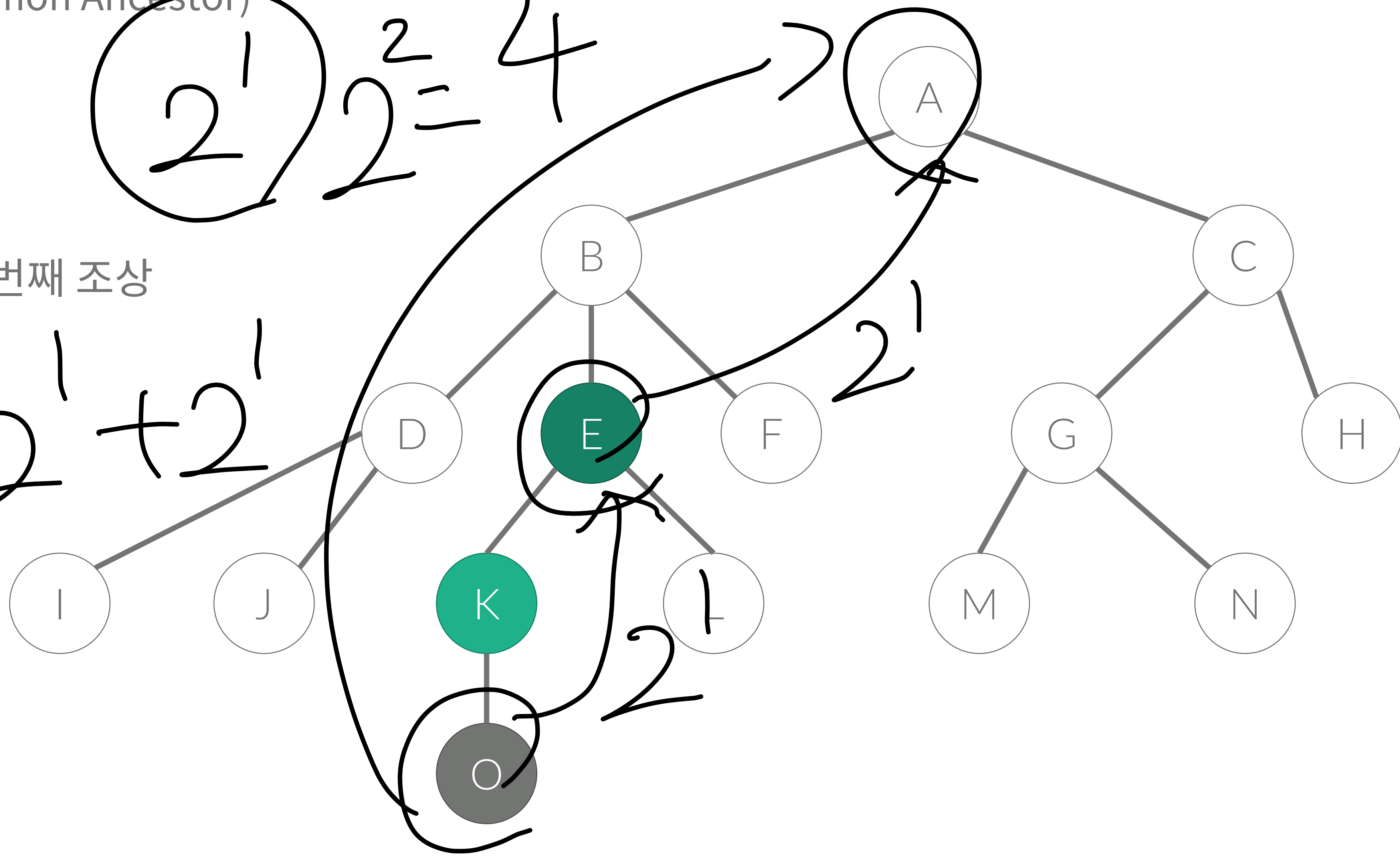
가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

- $j = 1$
- $2^1 = 2$ 번째 조상
- 1번째 조상의 1번째 조상

2^1
 $2^2 = 4$

$2^2 = 2^1 + 2^1$

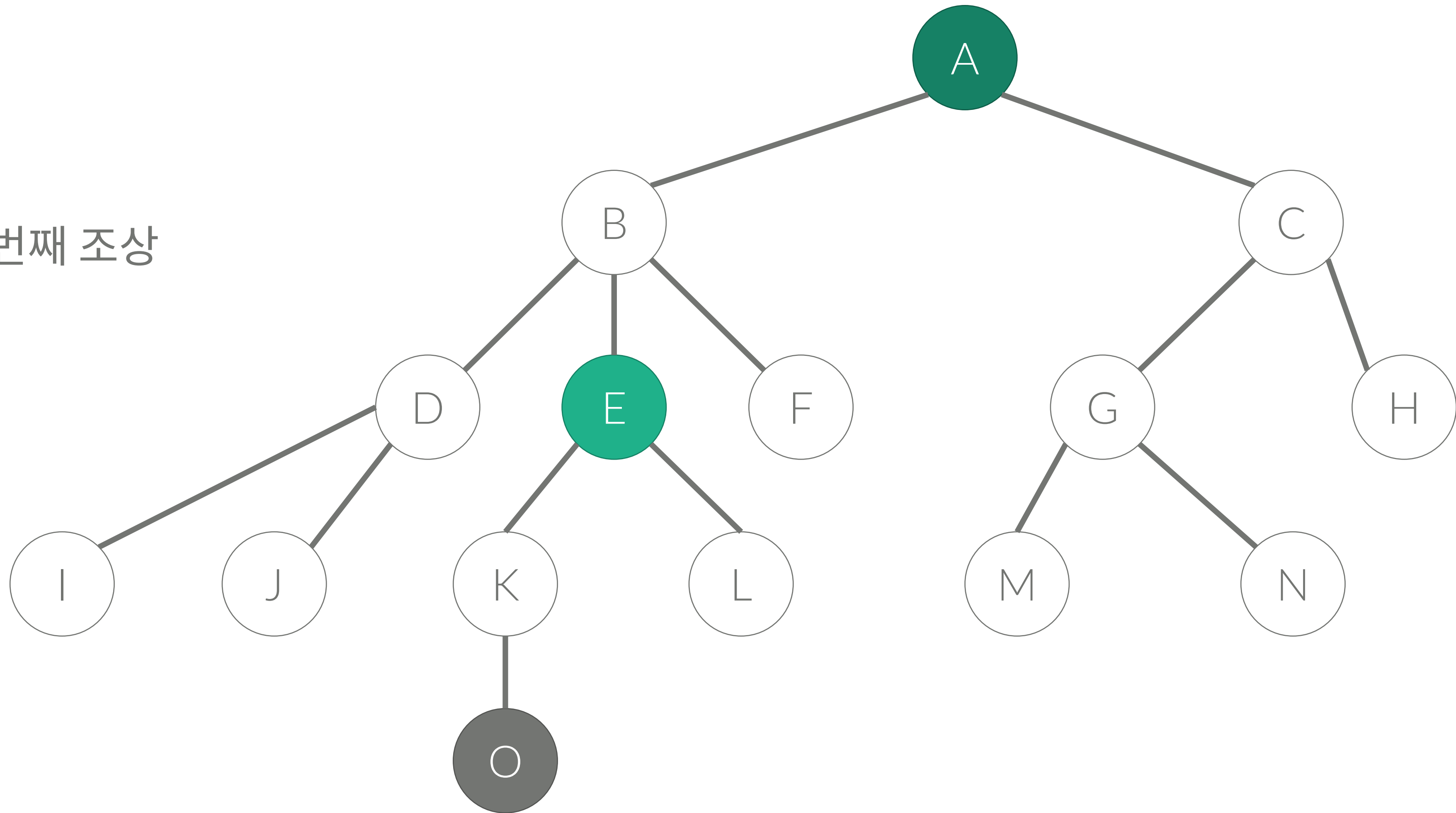


가장 가까운 공통 조상 찾기

28

LCA (Lowest Common Ancestor)

- $j = 2$
- $2^2 = 4$ 번째 조상
- 2번째 조상의 2번째 조상



가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

```
for (int i=1; i<=n; i++) {  
    p[i][0] = parent[i];  
}
```

가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

```
for (int j=1; (1<<j) < n; j++) {  
    for (int i=1; i<=n; i++) {  
        if (p[i][j-1] != 0) {  
            p[i][j] = p[p[i][j-1]][j-1];  
        }  
    }  
}
```

가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

- x와 y의 LCA 구하기

- 두 노드의 레벨이 다르면

- 레벨이 같을 때까지 레벨이 큰 것을 2^k 칸씩 위로 올린다. (k는 1씩 감소)

- 두 노드의 레벨이 같아졌으면

- 같은 노드가 되지 않을 때까지 2^k 칸씩 위로 올린다. (k는 1씩 감소)

- 그 다음 마지막으로 1칸 올린다.

가장 가까운 공통 조상 찾기

높이: 4

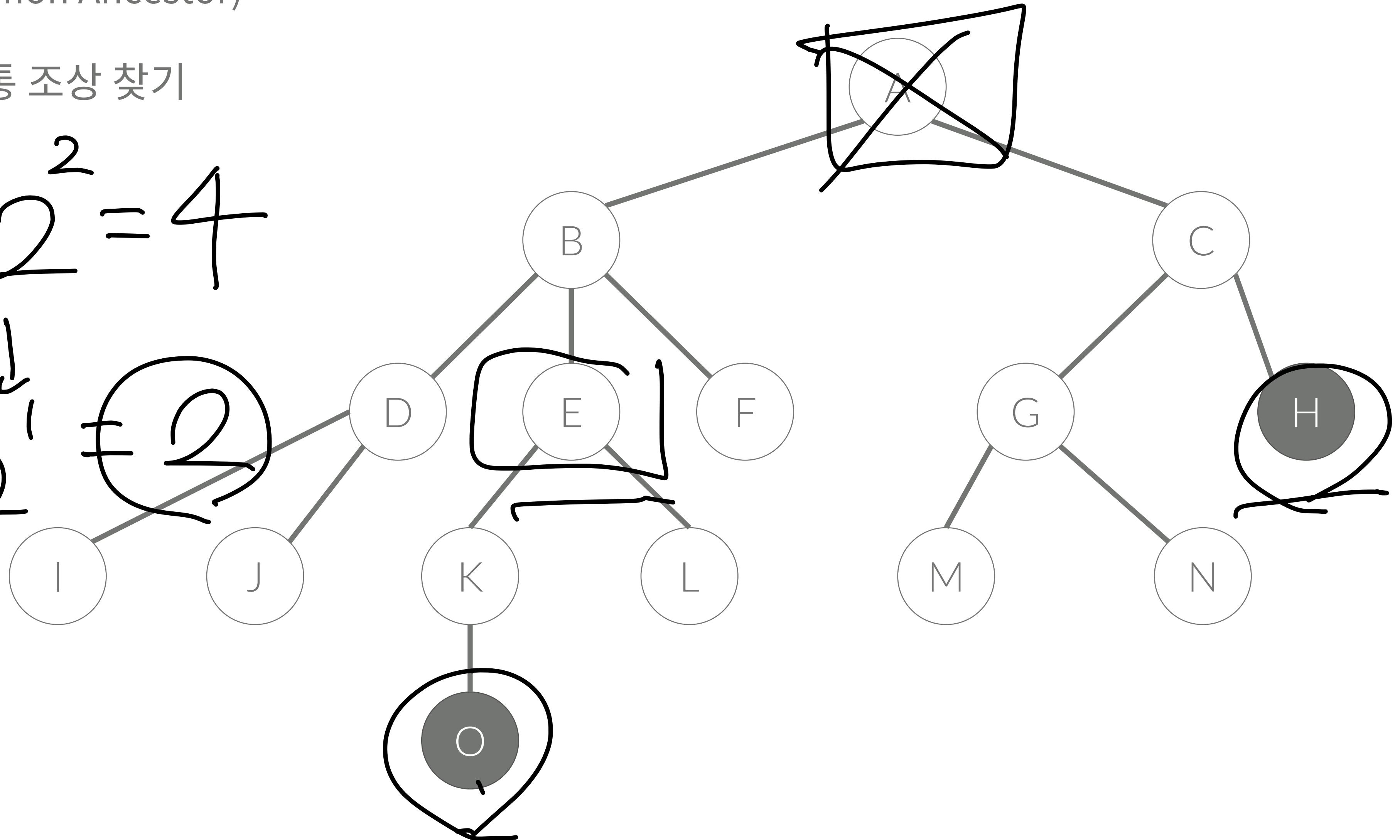
32

LCA (Lowest Common Ancestor)

- 가장 가까운 공통 조상 찾기

$$2^k = 2^2 = 4$$

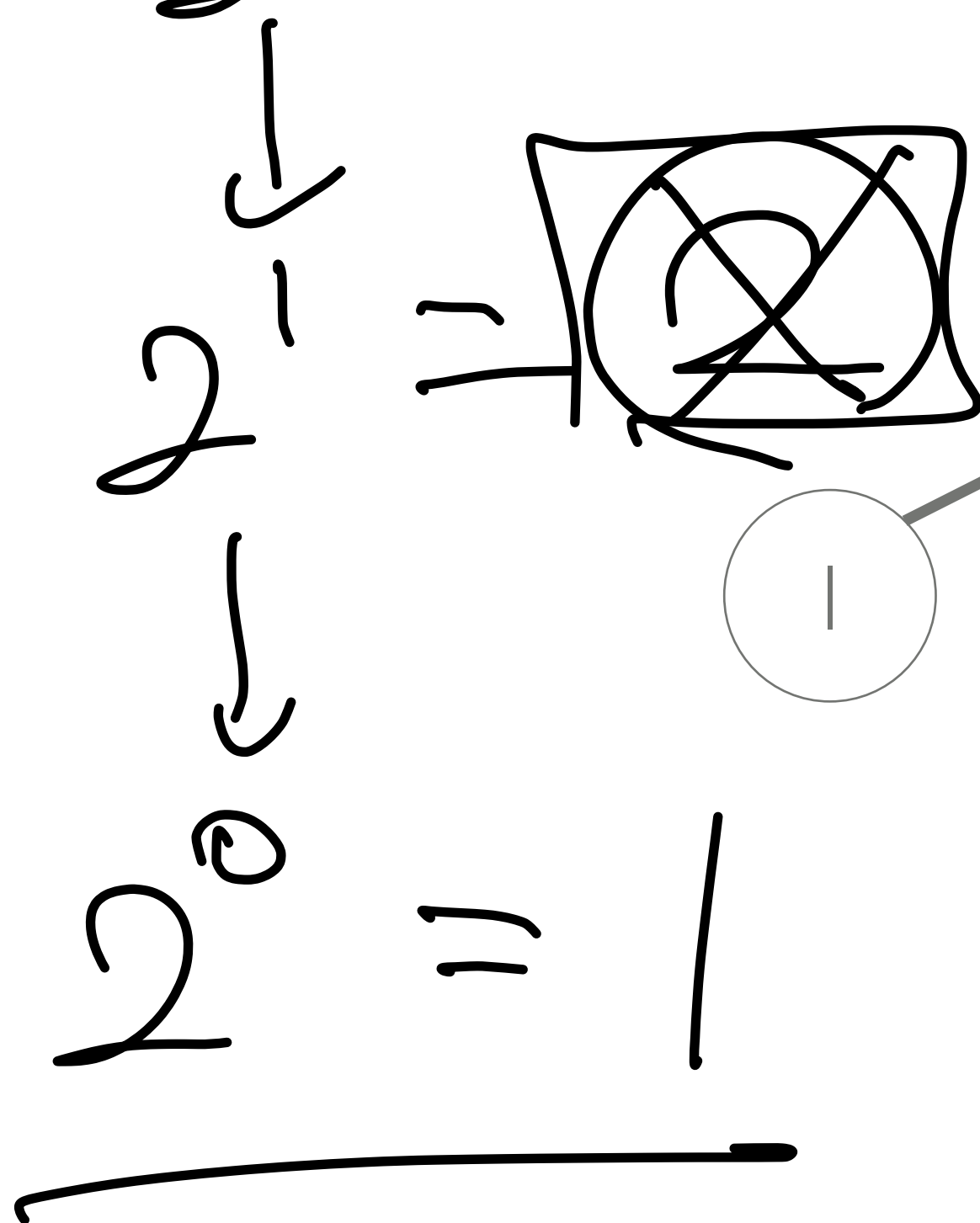
$$2^1 = 2$$



$$h = 4$$

LCA (Lowest Common Ancestor)

- $$2^2 = \cancel{4}$$

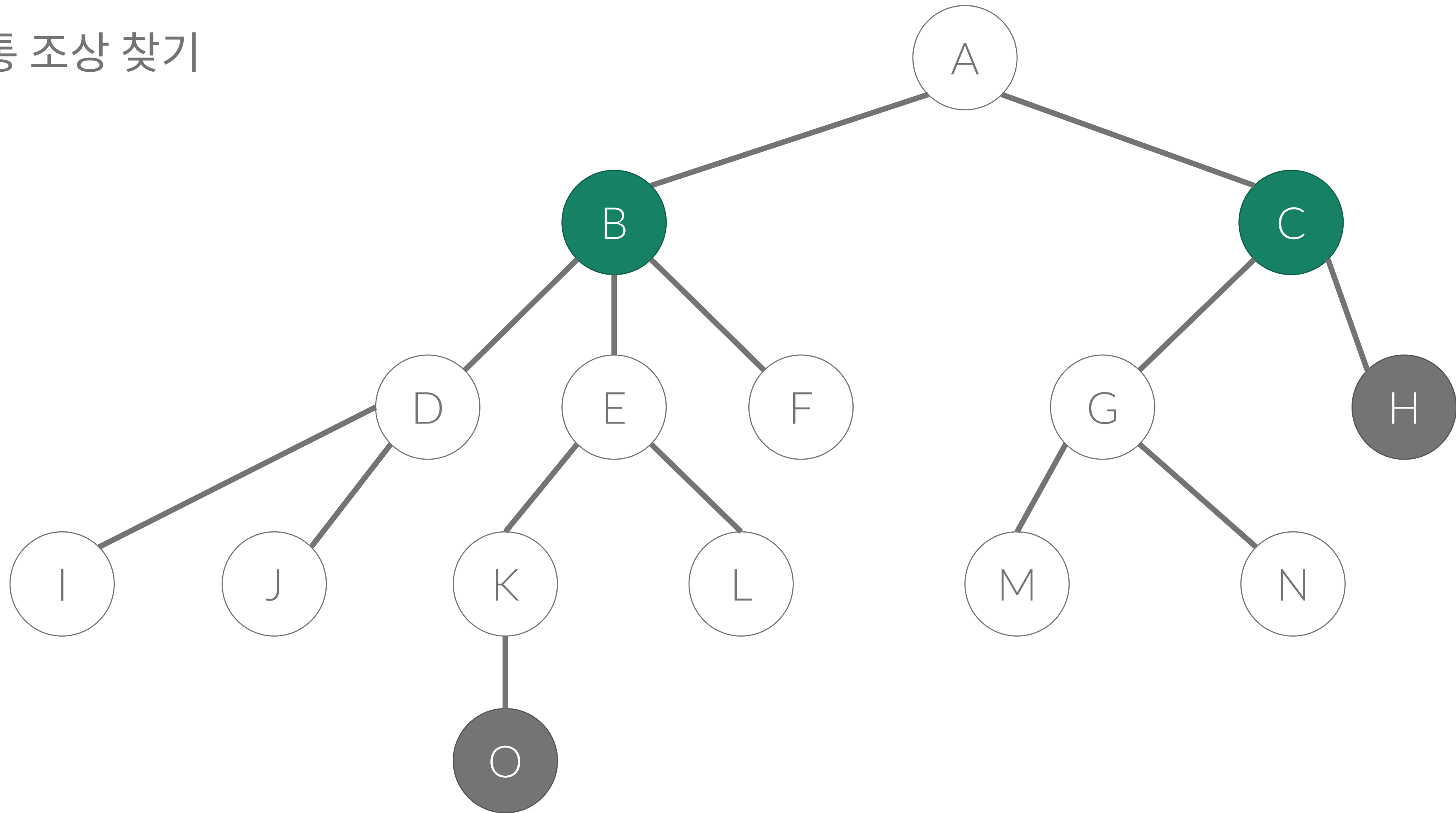


가장 가까운 공통 조상 찾기

34

LCA (Lowest Common Ancestor)

- 가장 가까운 공통 조상 찾기



가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

- x 와 y 의 LCA 구하기
- 두 노드의 레벨이 다르면
- 레벨이 같을 때까지 레벨이 큰 것을 2^k 칸 씩 위로 올린다. (k 는 1씩 감소)
- 두 노드의 레벨이 같아졌으면
- 같은 노드가 되지 않을 때까지 2^k 칸씩 위로 올린다. (k 는 1씩 감소)
- 그 다음 마지막으로 1 칸 올린다.

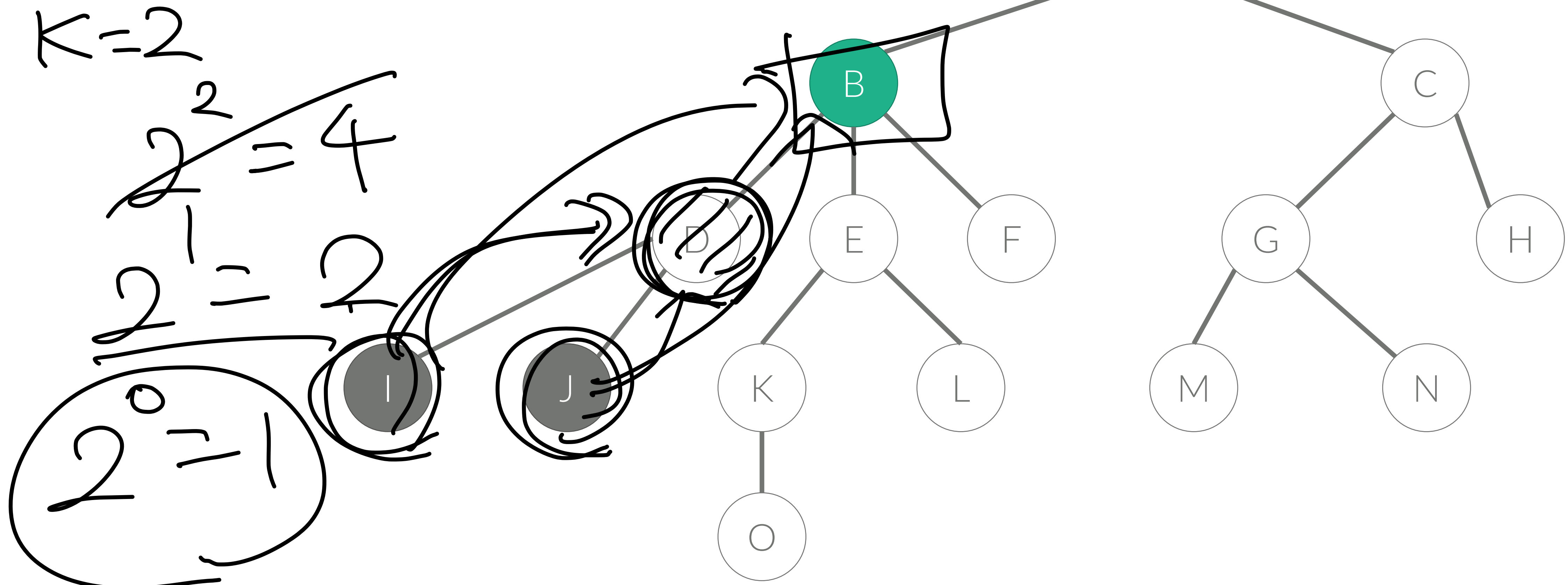
가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

$h=4$

36

- 가장 가까운 공통 조상 찾기

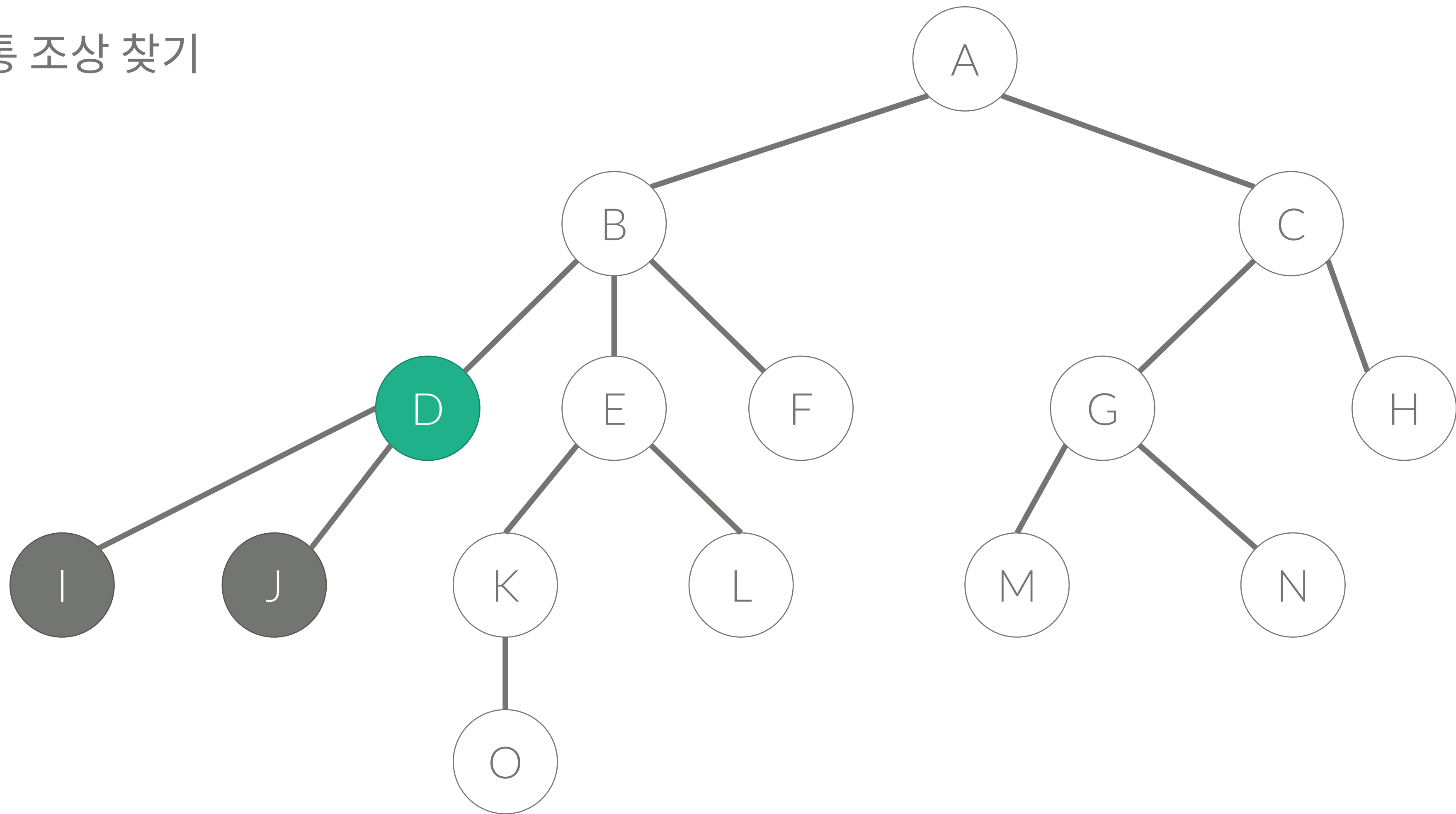


가장 가까운 공통 조상 찾기

37

LCA (Lowest Common Ancestor)

- 가장 가까운 공통 조상 찾기

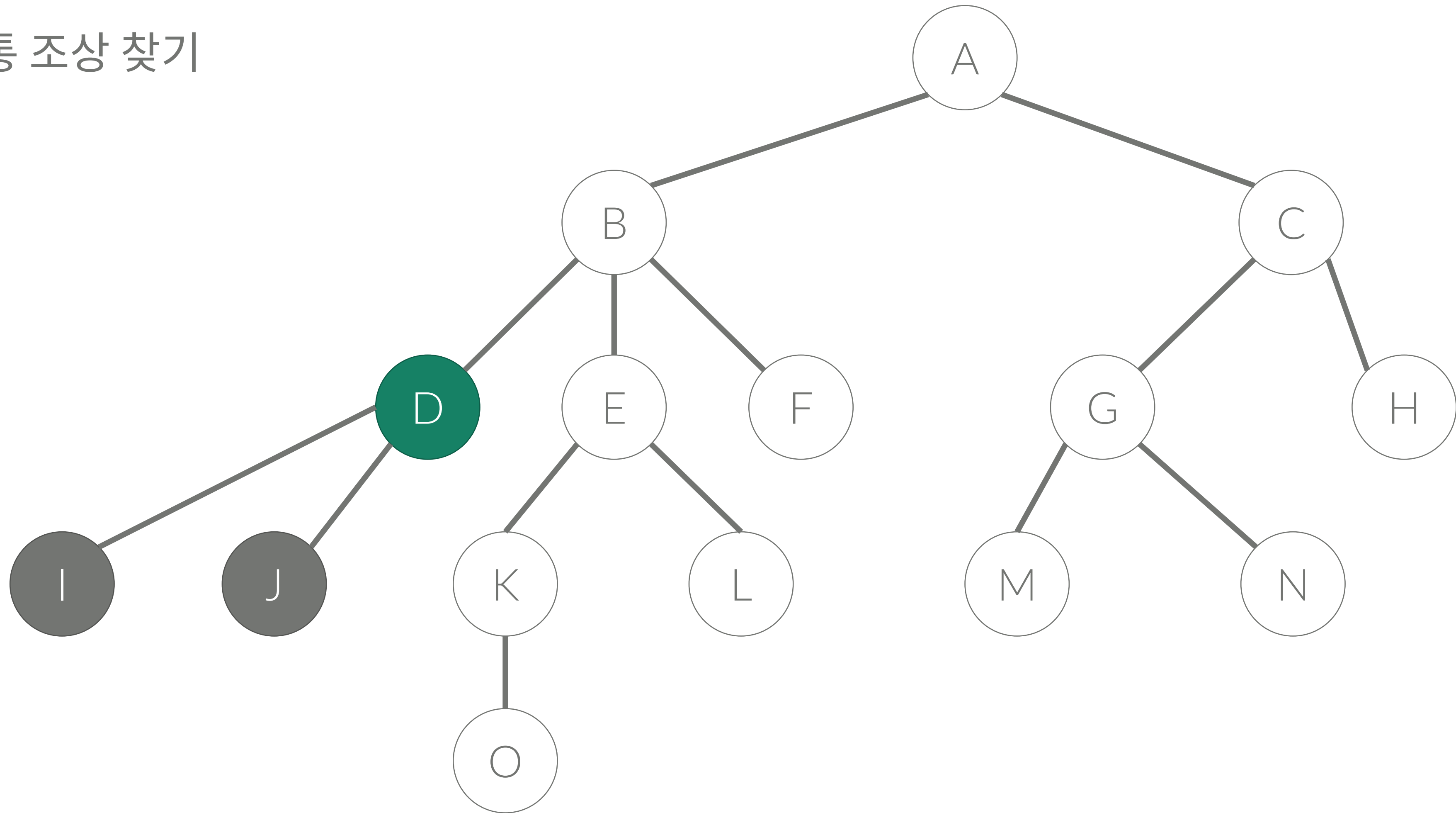


가장 가까운 공통 조상 찾기

38

LCA (Lowest Common Ancestor)

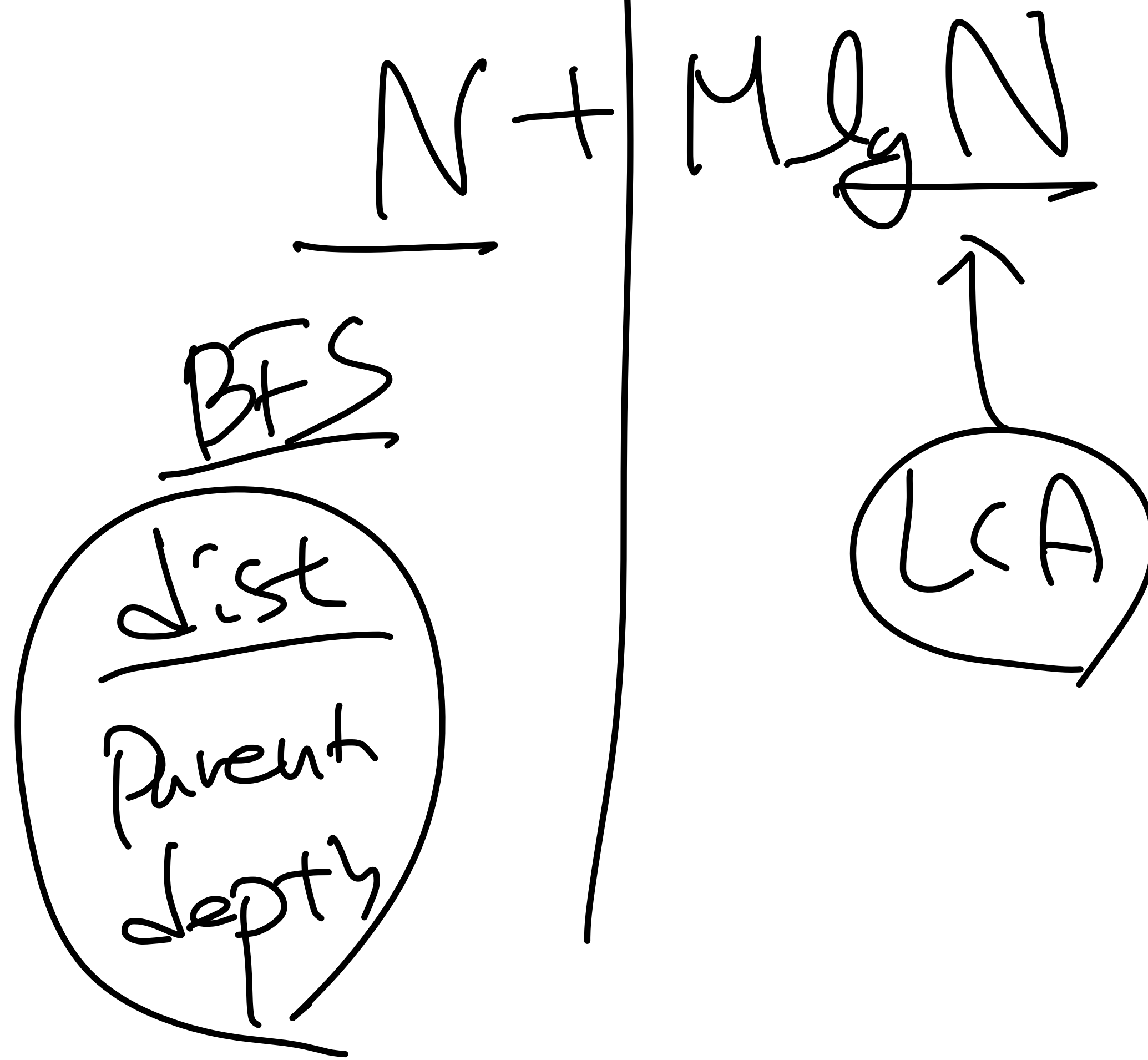
- 가장 가까운 공통 조상 찾기



LCA 2

<https://www.acmicpc.net/problem/11438>

- 소스: <http://codeplus.codes/6139e37a99dd4fca8183340c24fd5d7b>



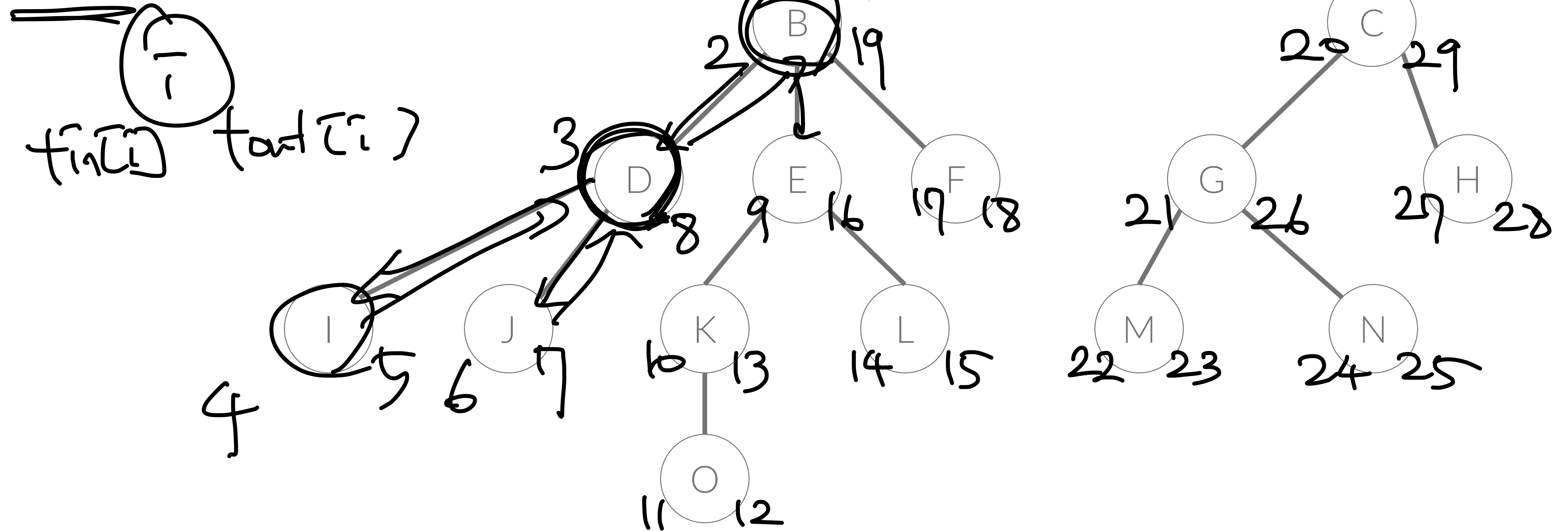
가장 가까운 공통 조상 찾기

DFS 2회

40

LCA (Lowest Common Ancestor)

- $tin[i]$ = dfs로 i 에 방문할 때, 몇 번째였는지
- $tout[i]$ = dfs로 i 에서 나갈 때, 몇 번째였는지

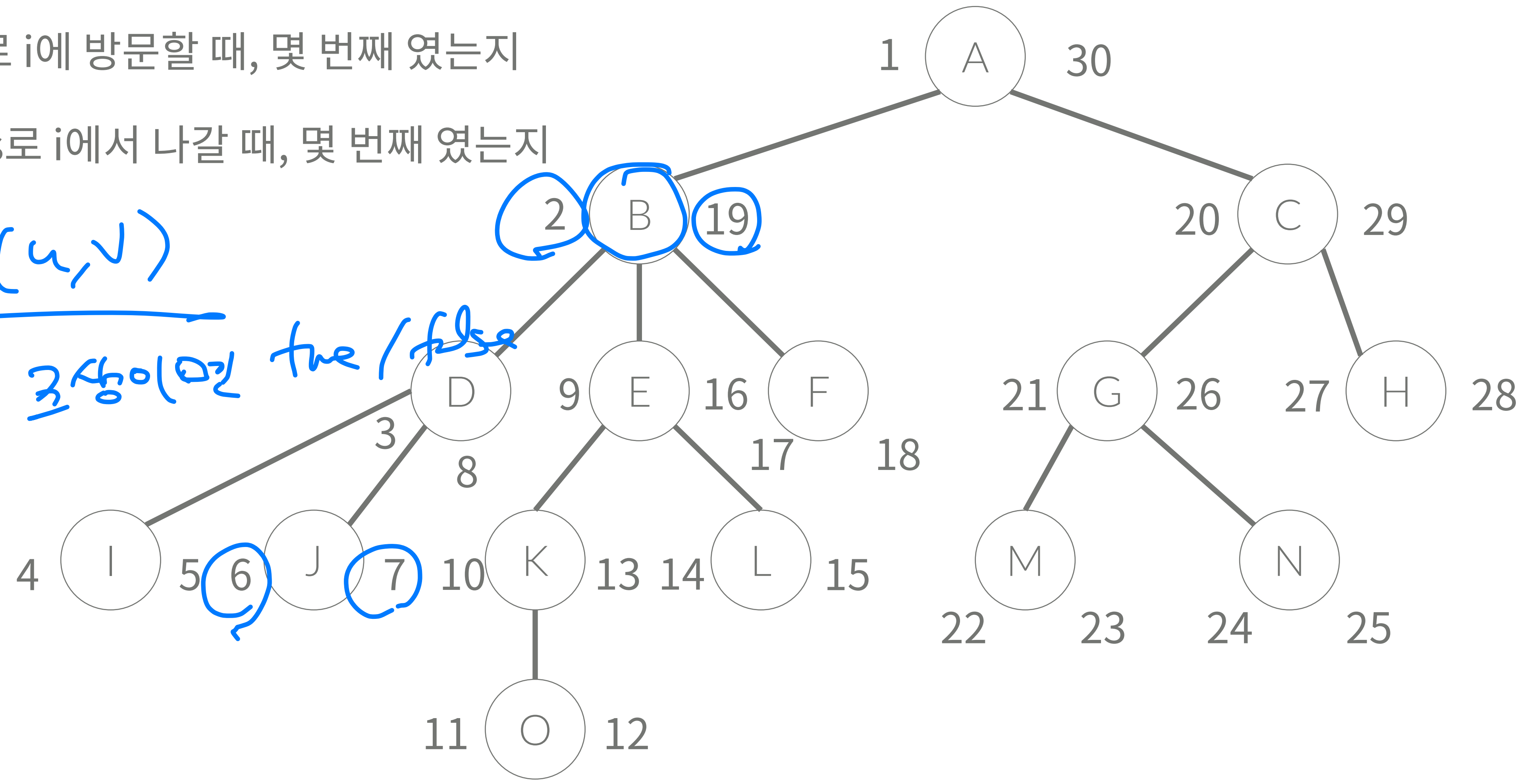


가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

- $tin[i]$ = dfs로 i 에 방문할 때, 몇 번째였는지
- $tout[i]$ = dfs로 i 에서 나갈 때, 몇 번째였는지

Upper(u,v)
u가 ~ v의 조상이면 true/false



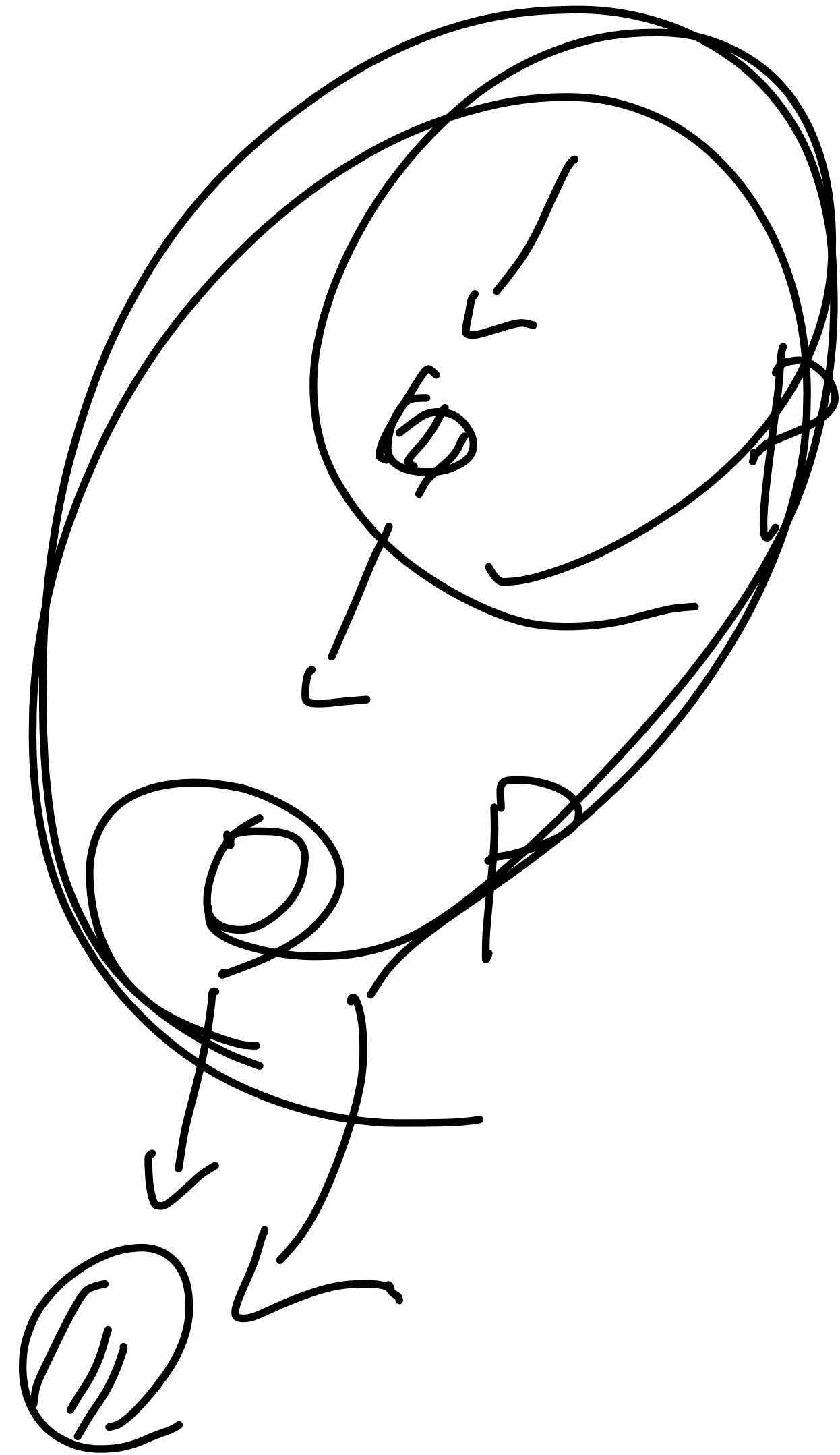
가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

```
void dfs(int v, int parent) {
    tin[v] = ++timer;
    p[v][0] = parent;
    for (int i=1; i<=l; i++) {
        p[v][i] = p[p[v][i-1]][i-1];
    }
    for (int to : a[v]) {
        if (to != parent) {
            dfs(to, v);
        }
    }
    tout[v] = ++timer;
}
```

DP

DFS



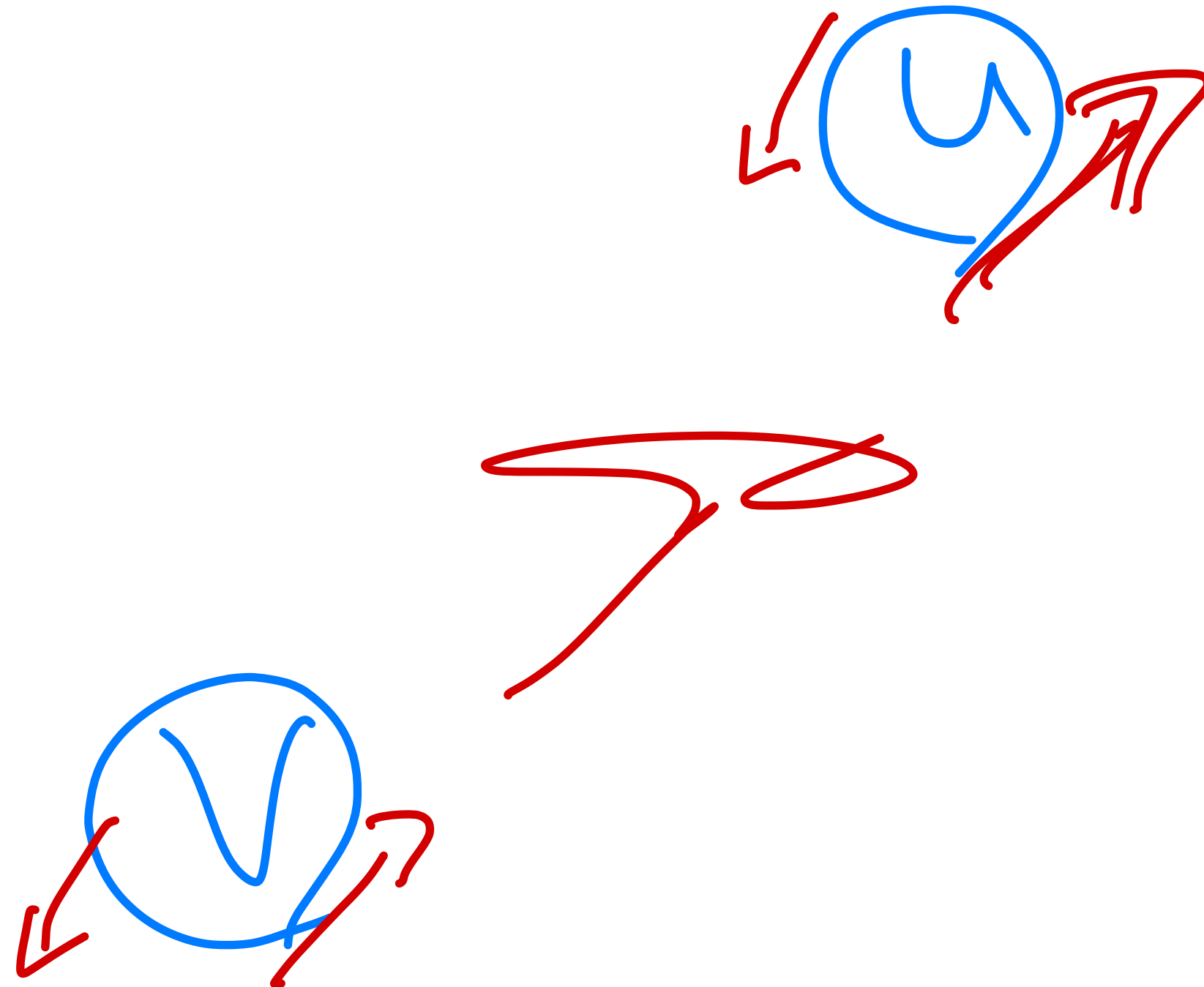
가장 가까운 공통 조상 찾기

LCA (Lowest Common Ancestor)

```
bool upper(int u, int v) {
    return (tin[u] <= tin[v] && tout[u] >= tout[v]);
}
```

u가
v의 조상?

O(1)



가장 가까운 공통 조상 찾기

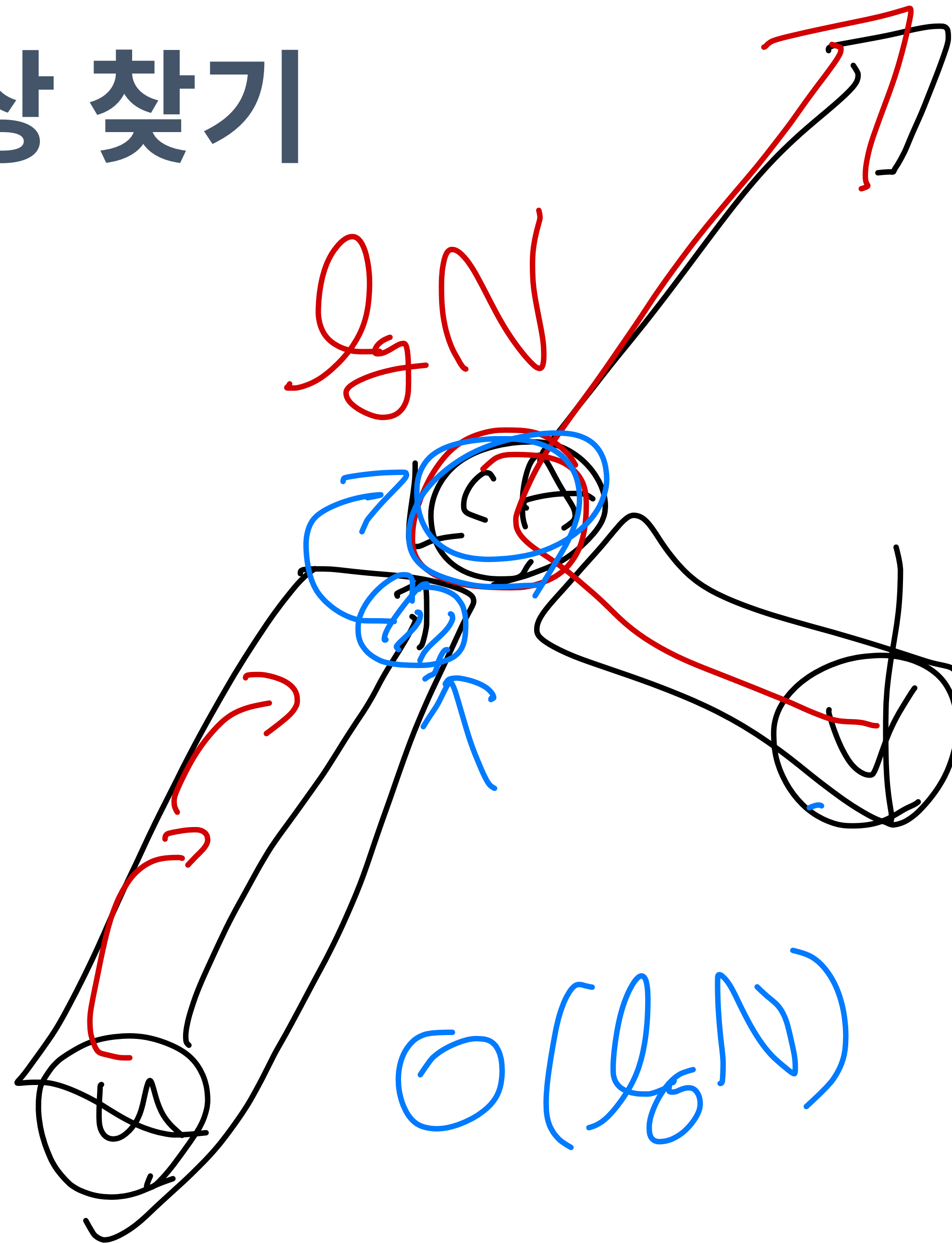
LCA (Lowest Common Ancestor)

```

int lca(int u, int v) {
    if (upper(u, v)) return u;
    if (upper(v, u)) return v;
    for (int i=l; i>=0; i--) {
        if (!upper(p[u][i], v)) {
            u = p[u][i];
        }
    }
    return p[u][0];
}
  
```

Handwritten annotations on the code:

- `upper(u, v)` and `upper(v, u)` are annotated with `u` and `v` respectively, with arrows pointing to the nodes in the diagram.
- `p[u][i]` is annotated with `u` and `i`, with arrows pointing to the node and the index in the diagram.
- `p[u][0]` is annotated with `u`, with an arrow pointing to the node.



LCA 2

<https://www.acmicpc.net/problem/11438>

- 소스: <http://codeplus.codes/ec66b9508a3f403cbb018e141ef0f073>