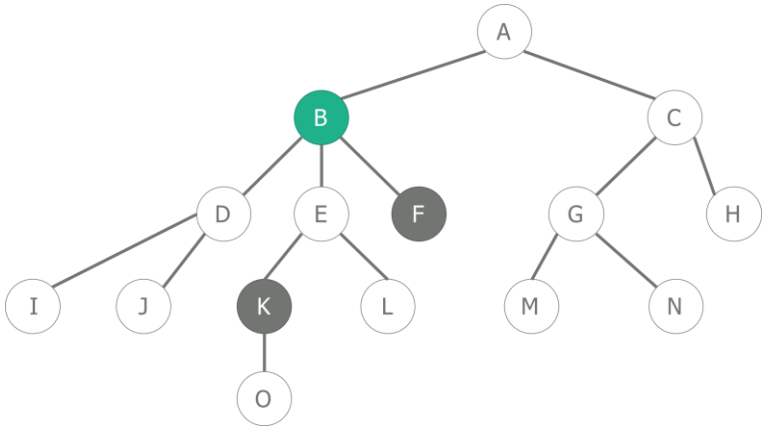
LCA(Lowest Common Ancestor)

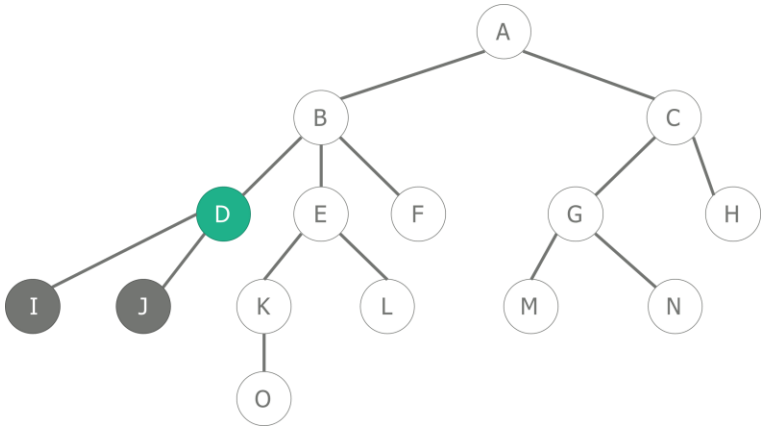
LCA

- 두 노드의 가장 가까운 조상을 찾는 문제를 LCA 라고 합니다.

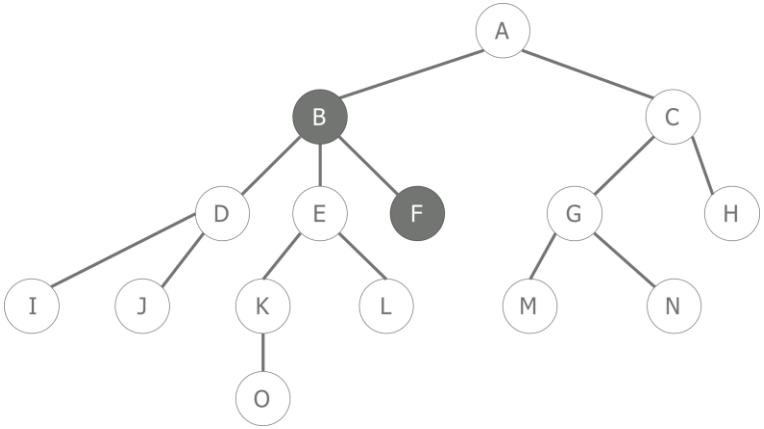
- K와 F의 LCA는 B 입니다.



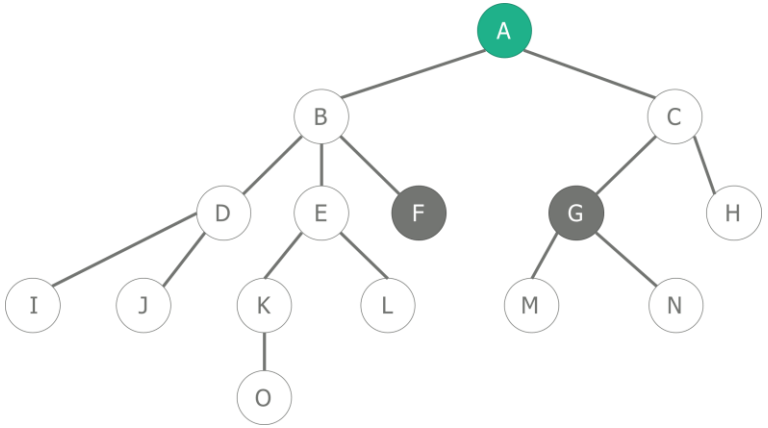
- I와 J의 LCA는 D 입니다.



- B와 F의 LCA는 B 입니다.



- F와 G의 LCA는 A 입니다.

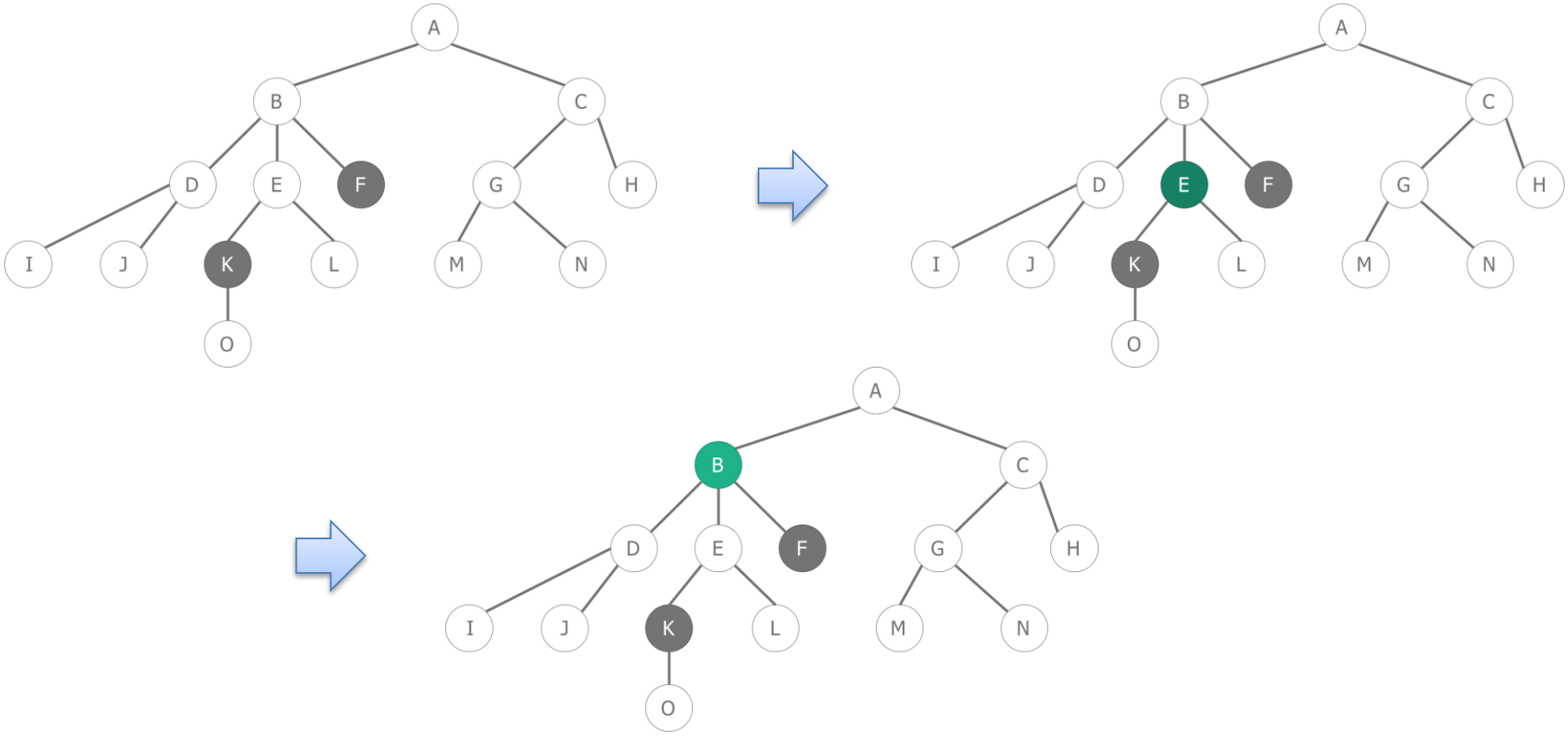


Gaussian

LCA(Lowest Common Ancestor)

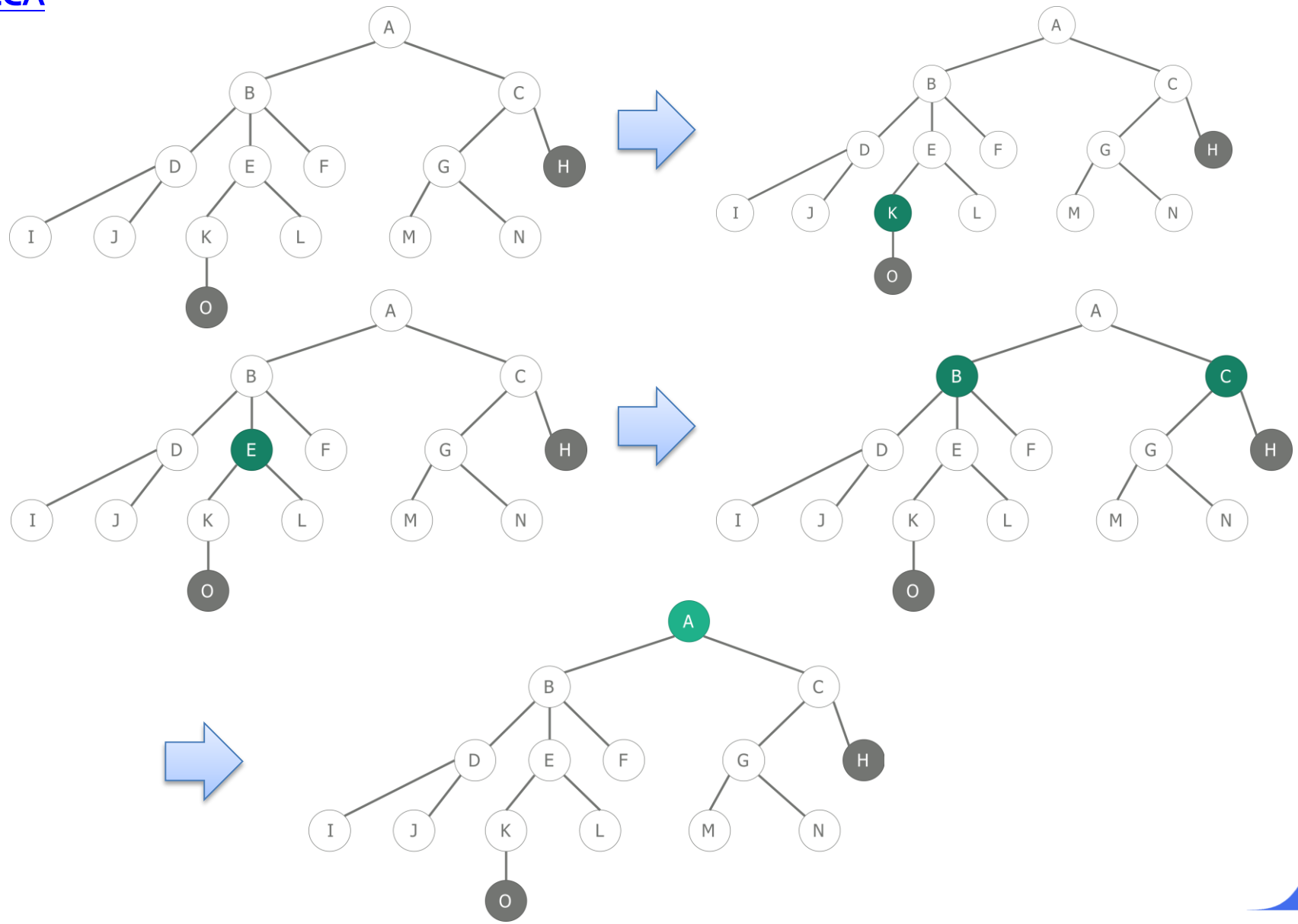
LCA

- x와 y의 LCA를 구하기 위해 두 노드의 레벨이 다르면 레벨이 같을 때 까지 레벨이 큰 것을 한 칸씩 위로 올린다.
- 두 노드의 레벨이 같아 졌으면 같은 노드가 될 때 까지 한 칸씩 위로 올린다.



LCA(Lowest Common Ancestor)

LCA



Gaussian

LCA(Lowest Common Ancestor)

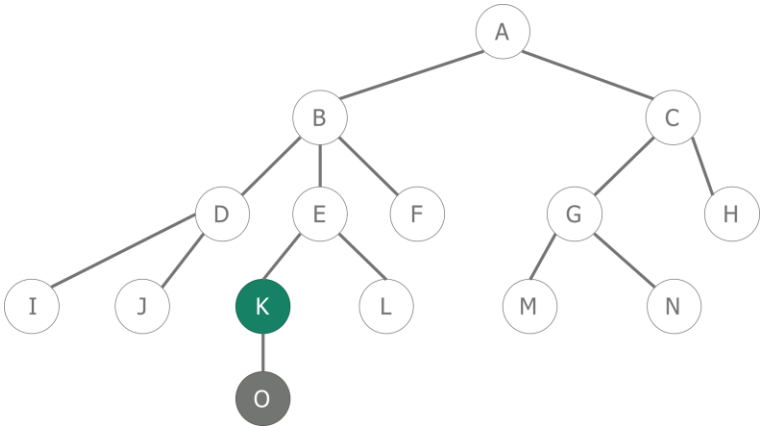
LCA

- 트리에서는 모든 정점 쌍 사이의 경로가 1개이기 때문에 어떤 정점 x에서 y로가는 경로가 최단 경로가 된다.
- 거리를 구하는 데 필요한 시간 : O(N)
- 총 쿼리의 개수 : M개
  ☞ 시간 복잡도 O(NM) : 정점의 개수와 쿼리의 개수가 많으면 복잡도가 너무 커진다.
- Dynamic Programming을 이용하여 시간 복잡도를 줄인다.
- p[i][j] = 노드 i의 2^j번째 조상
  ex) p[i][0] = 노드 i의 2^0 번째 조상 = 노드 i의 1번째 조상 = 부모
  p[i][1] = 노드 i의 2^1 번째 조상 = 노드 i의 2번째 조상 = 부모의 부모
  p[i][j] = 노드 i의 2^j번째 조상 = 노드 i의 2^{j-1}번째 조상의 2^{j-1} 번째 조상
  = p[p[i][j-1]][j-1]
- p[i][j] = p[p[i][j-1]][j-1]
- 초깃값 : p[i][0] = parent[i]

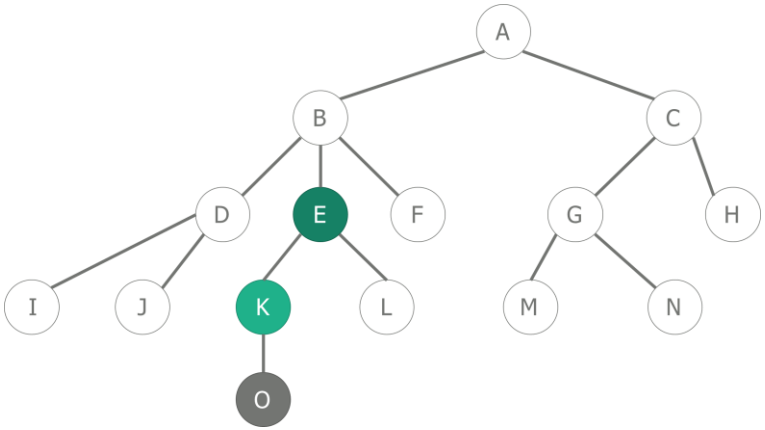
LCA(Lowest Common Ancestor)

[LCA]

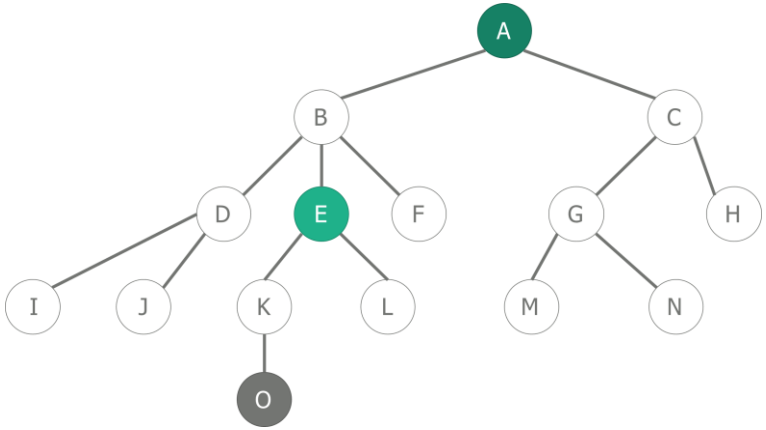
- j = 0 : 2^0 = 1 번째 조상



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



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

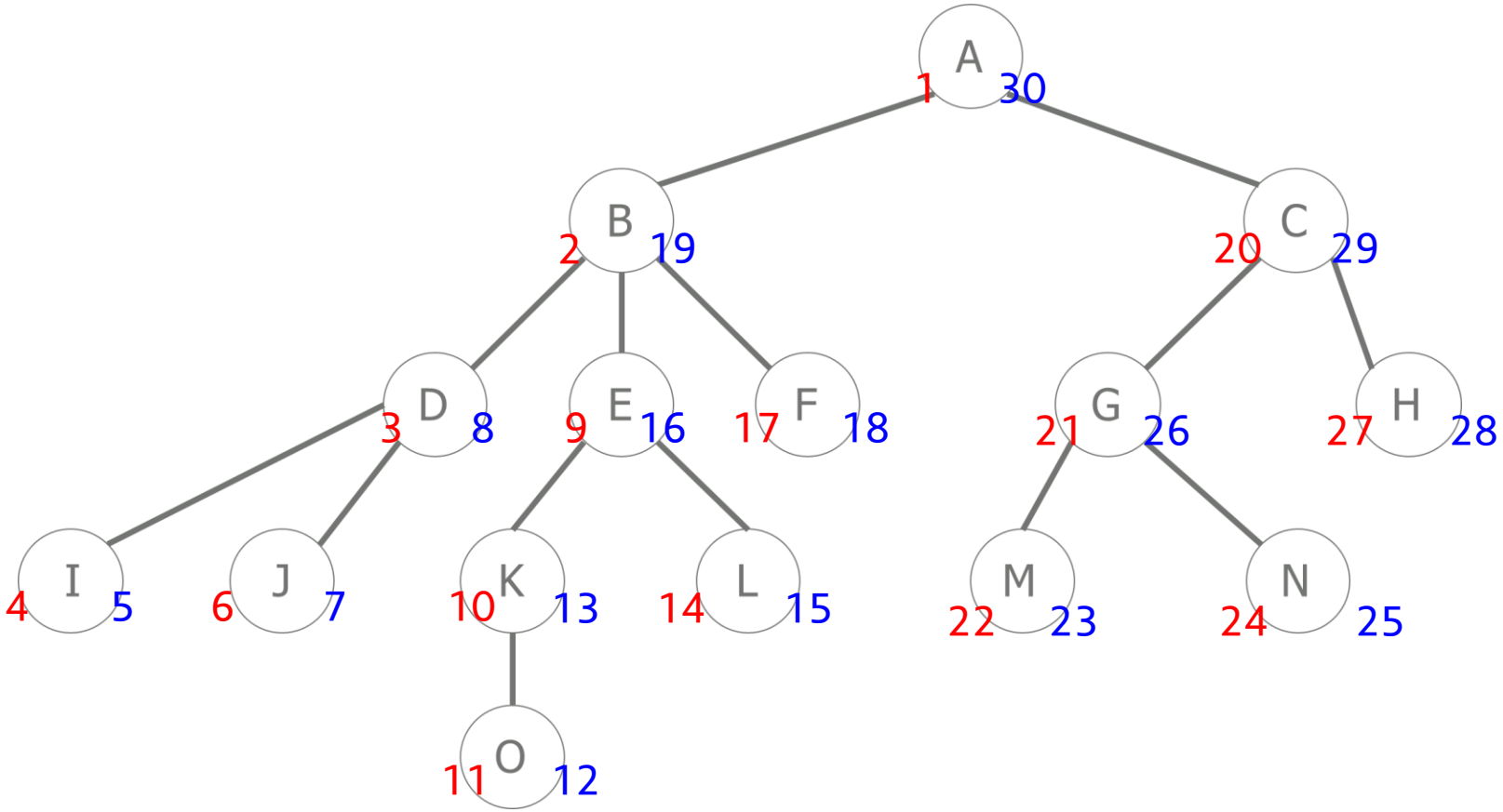


Gaussian

LCA(Lowest Common Ancestor)

LCA

- tin[i] = dfs로 i에 방문하였을 때, 몇 번째로 방문하였는지 저장
- tout[i] = dfs로 i에서 나갈 때, 몇 번째로 방문하였는지 저장



LCA(Lowest Common Ancestor)

LCA

- tin[i] = dfs로 i에 방문하였을 때, 몇 번째로 방문하였는지 저장
- tout[i] = dfs로 i에서 나갈 때, 몇 번째로 방문하였는지 저장

```
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;
}
```

timer 및 dfs 탐색 부

p[i][j] = 노드 i의 2^j번째 조상

timer 및 dfs 탐색 부

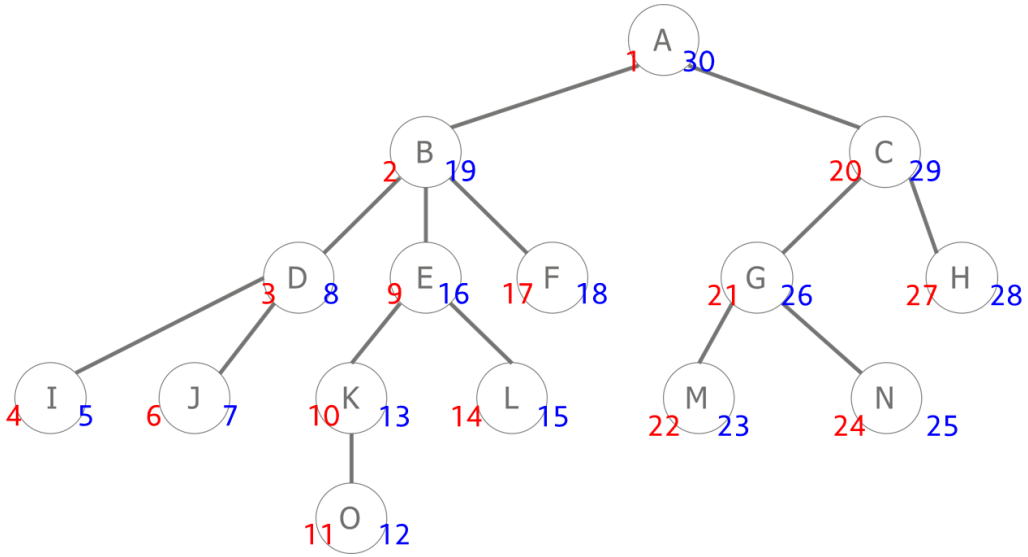
timer 및 dfs 탐색 부

LCA(Lowest Common Ancestor)

LCA

- tin[u] <= tin[v] 이면 u는 v의 조상이 될 수도 있다.
- tout[u] >= tout[v] 이면 u는 v의 조상이 될 수도 있다.

∴ **(tin[u] <= tin[v]) && (tout[u] >= tout[v])** 조건을 만족하면 반드시 조상이 된다.



- upper(u, v) : u가 v의 조상인가?

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

LCA(Lowest Common Ancestor)

LCA

```
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];
}
```

u가 v의 조상이면 u를 리턴

v가 u의 조상이면 v를 리턴

u가 v의 위에 있지 않을 때 까지
u를 조상으로 update

update 된 u의 부모가 LCA가 된다.

LCA(Lowest Common Ancestor)

LCA

- J와 L의 LCA를 찾으려고 한다.
  ① J의 2^2 조상은 A 이므로 L의 조상이 된다.
  ② J의 2^1조상은 B 이므로 L의 조상이 된다.
  ③ J의 2^0 조상은 D 이므로 L의 조상이 되지 않는다.
  ④ 조상이 안되는 노드를 찾으면 그 노드의 parent = B가 LCA가 된다.