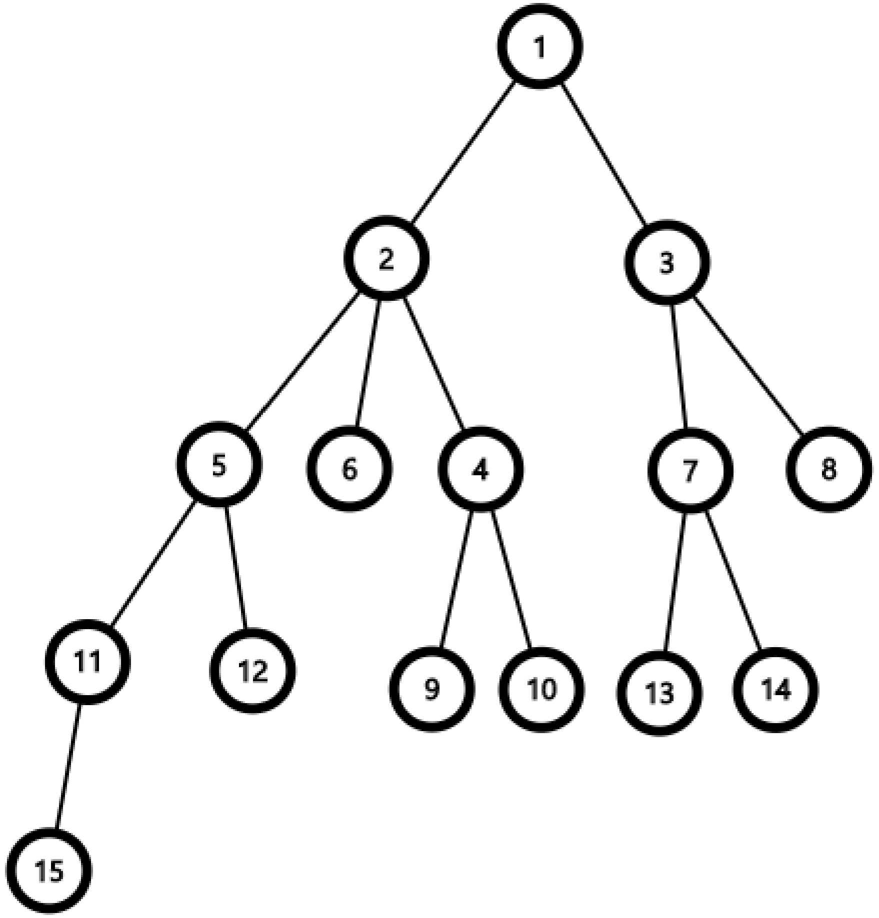
Lowest Common Ancestor

2019-2020 Winter

20141574 임지환 (Sogang University)

- 최소 공통 조상(Lowest Common Ancestor; LCA)
  - rooted tree에서 임의의 두 노드 쌍(u,v)를 모두 자손으로 갖는 노드(조상) 중 가장 아래 있는 노드



- 두 노드 사이의 거리
- 두 노드 사이의 간선 중 가중치가 최대(최소)인 것의 크기
- 두 노드 사이 경로에서 k번째 노드

- 희소 테이블

  배열 $arr[1 \dots N]$에 대해 $[L, R]$ 구간 내 쿼리 처리를 빠르게 하는 구조

- Performance
  - Construction : $O(NlogN)$
  - Query : $O(logN)$ ($O(1)$ in some case)

- Condition
  - Static Data
  - Associativity

- Basic Form

$Table[i][j] \coloneqq i\sim(i+2^j-1)$ 구간 내 (어떤) 값

$$Table[i][j] = f(arr[i], arr[i+1], \ldots, arr[i+2^j-1])$$

- Basic Form

$Table[i][j] \coloneqq i \sim (i + 2^j - 1)$ 구간 내 (어떤) 값

$$Table[i][j] = f(arr[i], arr[i+1], \dots, arr[i + 2^j - 1])$$

1	2	3	4	5	6	7	8

$\boldsymbol{Table[1][0]}$

- Basic Form

$Table[i][j] \coloneqq i\sim(i+2^j-1)$ 구간 내 (어떤) 값

$$Table[i][j] = f(arr[i], arr[i+1], \ldots, arr\left[i+2^j-1\right])$$

1	2	3	4	5	6	7	8

$\boldsymbol{Table[1][1]}$

- Basic Form

$Table[i][j] \coloneqq i \sim (i + 2^j - 1)$ 구간 내 (어떤) 값

$$Table[i][j] = f\left(arr[i], arr[i+1], \ldots, arr\left[i + 2^j - 1\right]\right)$$

1	2	3	4	5	6	7	8

$\boldsymbol{Table[1][2]}$

- Basic Form

$Table[i][j] \coloneqq i \sim (i + 2^j - 1)$ 구간 내 (어떤) 값

$$Table[i][j] = f(arr[i], arr[i+1], \dots, arr\left[i + 2^j - 1\right])$$

1	2	3	4	5	6	7	8

$\boldsymbol{Table[3][2]}$

- Basic Form

$Table[i][j] \coloneqq i\sim(i+2^j-1)$ 구간 내 (어떤) 값

$$Table[i][j] = f(arr[i], arr[i+1], \dots, arr[i+2^j-1])$$

1	2	3	4	5	6	7	8

$\boldsymbol{Table[1][3]}$

$$Table[i][j] = f\left(arr[i], arr[i+1], \ldots, arr\left[i+2^j-1\right]\right)$$

i	$i+1$	$\ldots$	$i+2^{j-1}-1$	$i+2^{j-1}$	$i+2^{j-1}+1$	$\ldots$	$i+2^j-1$

$$Table[i][j] = f\left(arr[i], arr[i+1], \dots, arr\left[i+2^j-1\right]\right)$$

$$Table[i][j-1] = f\left(arr[i], arr[i+1], \dots, arr\left[i+2^{j-1}-1\right]\right)$$

$$Table\left[i+2^{j-1}\right][j-1] = f(arr\left[i+2^{j-1}\right], arr\left[i+2^{j-1}+1\right], \dots, arr\left[i+2^j-1\right])$$

i	$i+1$	…	$i+2^{j-1}-1$	$i+2^{j-1}$	$i+2^{j-1}+1$	…	$i+2^j-1$

Sparse Table

$$Table[i][j] = f\left(arr[i], arr[i+1], \dots, arr\left[i+2^j-1\right]\right)$$

$$Table[i][j-1] = f\left(arr[i], arr[i+1], \dots, arr\left[i+2^{j-1}-1\right]\right)$$

$$Table\left[i+2^{j-1}\right][j-1] = f\left(arr\left[i+2^{j-1}\right], arr\left[i+2^{j-1}+1\right], \dots, arr\left[i+2^j-1\right]\right)$$

i	$i+1$	…	$i+2^{j-1}-1$	$i+2^{j-1}$	$i+2^{j-1}+1$	…	$i+2^j-1$

by associativity,

$$Table[i][j] = f\left(Table[i][j-1], Table\left[i+2^{j-1}\right][j-1]\right)$$

- $m(1 \leq m \leq 200{,}000)$개의 양의 정수 배열
- given $f(i)$ for each i $(1 \leq i \leq m)$

- $f_1(x) = f(x)$
- $f_{n+1}(x) = f\left(f_n(x)\right)$

- $Q(1 \leq Q \leq 200{,}000)$개의 $n(1 \leq n \leq 500{,}000), x(1 \leq x \leq m)$에 대하여 $f_n(x)$?

- **문제 해결 과정**

1. associativity가 성립하는가?
2. update가 있는가?

- 문제 해결 과정

1. associativity가 성립하는가?
   - $f_{n+m}(x) = f(f(f(\ldots f(f_m(x)) = f_n(f_m(x))$
2. update가 있는가?
   - 아니요.

- 문제 해결 과정

1. associativity가 성립하는가?
   - $f_{n+m}(x) = f(f(f(\dots f(f_m(x)) = f_n(f_m(x))$
2. update가 있는가?
   - 아니요.
3. $let \;\; n = 2^m, then\ f_n(x) = f_{2^m}(x)$
4. $d[n][x] \; := x$에 $n(= 2^m)$번 합성함수를 적용한 값

- Review for Exponential by Squaring

ex) N = 214

N = $128 + 64 + 16 + 4 + 2 = 2^7 + 2^6 + 2^4 + 2^2 + 2^1$

- Review for Exponential by Squaring

ex) N = 214

N = $128 + 64 + 16 + 4 + 2 = 2^7 + 2^6 + 2^4 + 2^2 + 2^1$

by associativity, $f_{214}(x) = f_{128+86}(x) = f_{128+64+22}(x) = \cdots = f_{2^7+2^6+\cdots+2^1}(x)$

$$f_{214}(x) = d[7][f_{86}(x)] = d[7]\big[d[6][f_{22}(x)]\big]$$
$$\ldots = d[7]\Big[d[6]\Big[d[4]\Big[d[2]\big[d[1][x]\big]\Big]\Big]\Big]$$

- Review for Exponential by Squaring

ex) N = 214

N = $128 + 64 + 16 + 4 + 2 = 2^7 + 2^6 + 2^4 + 2^2 + 2^1$

by associativity, $f_{214}(x) = f_{128+86}(x) = f_{128+64+22}(x) = \cdots = f_{2^7+2^6+\cdots+2^1}(x)$

$$f_{214}(x) = d[7][f_{86}(x)] = d[7]\big[d[6][f_{22}(x)]\big]$$
$$\ldots = d[7]\Big[d[6]\Big[d[4]\Big[d[2]\big[\underbrace{\boxed{d[1][x]}}_{f_2(x)}\big]\Big]\Big]\Big]$$

- Review for Exponential by Squaring

ex) N = 214

N = $128 + 64 + 16 + 4 + 2 = 2^7 + 2^6 + 2^4 + 2^2 + 2^1$

by associativity, $f_{214}(x) = f_{128+86}(x) = f_{128+64+22}(x) = \cdots = f_{2^7+2^6+\cdots+2^1}(x)$

$$f_{214}(x) = d[7][f_{86}(x)] = d[7]\left[d[6][f_{22}(x)]\right]$$

$$\ldots = d[7]\left[d[6]\left[d[4]\left[\underbrace{\boxed{d[2]\left[d[1][x]\right]}}_{f_6(x)}\right]\right]\right]$$

- Review for Exponential by Squaring

ex) N = 214

N = $128 + 64 + 16 + 4 + 2 = 2^7 + 2^6 + 2^4 + 2^2 + 2^1$

by associativity, $f_{214}(x) = f_{128+86}(x) = f_{128+64+22}(x) = \cdots = f_{2^7+2^6+\cdots+2^1}(x)$

$$f_{214}(x) = d[7][f_{86}(x)] = d[7]\left[d[6][f_{22}(x)]\right]$$

$$\ldots = d[7]\left[d[6]\left[\underbrace{\boxed{d[4]\left[d[2]\left[d[1][x]\right]\right]}}_{f_{22}(x)}\right]\right]$$

- filling sparse table

1. $d[0][x]$: $f(x)$ (given)

- filling sparse table

1. $d[0][x]$: $f(x)$ (given)
2. $f_{2n}(x) = f_n\left(f_n(x)\right)$,

$$d[n][x] = d[n-1]\left[d[n-1][x]\right]$$

- 문제 해결 과정

1. associativity가 성립하는가?
   - $f_{n+m}(x) = f(f(f(\dots f(f_m(x)) = f_n(f_m(x))$
2. update가 있는가?
   - 아니요.
3. $let\ \ n = 2^m, then\ f_n(x) = f_{2^m}(x)$
4. $d[n][x]\ \ := x$에 $n(= 2^m)$번 합성함수를 적용한 값

- **문제 해결 과정**

1. associativity가 성립하는가?
   - $f_{n+m}(x) = f(f(f(\dots f(f_m(x)) = f_n(f_m(x))$
2. update가 있는가?
   - 아니요.
3. $let\ \ n = 2^m, then\ f_n(x) = f_{2^m}(x)$
4. $d[n][x]\ \ := x$에 $n(= 2^m)$번 합성함수를 적용한 값
5. 임의의 양의 정수 n를 이진수로 표현하여 0이 될 때까지 n 축소 & 함수 대응

- 문제 해결 과정

1. associativity가 성립하는가?
   - $f_{n+m}(x) = f(f(f(\ldots f(f_m(x))) = f_n(f_m(x))$
2. update가 있는가?
   - 아니요.
3. $let \;\; n = 2^m, then\, f_n(x) = f_{2^m}(x)$
4. $d[n][x] \;\; := x$에 $n(= 2^m)$번 합성함수를 적용한 값
5. 임의의 양의 정수 n를 이진수로 표현하여 0이 될 때까지 n 축소 & 함수 대응
6. Time Complexity
   - filling $d[\log n][x]$: $O(m \log n)$
   - each query : $O(\log n)$

code here

- $N(2 \leq N \leq 100{,}000)$개의 정점으로 구성된 root가 1번 노드인 tree
- $M(1 \leq M \leq 100{,}000)$개의 두 노드 쌍이 주어졌을 때 가장 가까운 공통 조상의 번호?

- prerequisite

1. Tree이기 때문에 경로가 유일하지만, 단순 dfs는 사용 불가
2. LCA :

- prerequisite

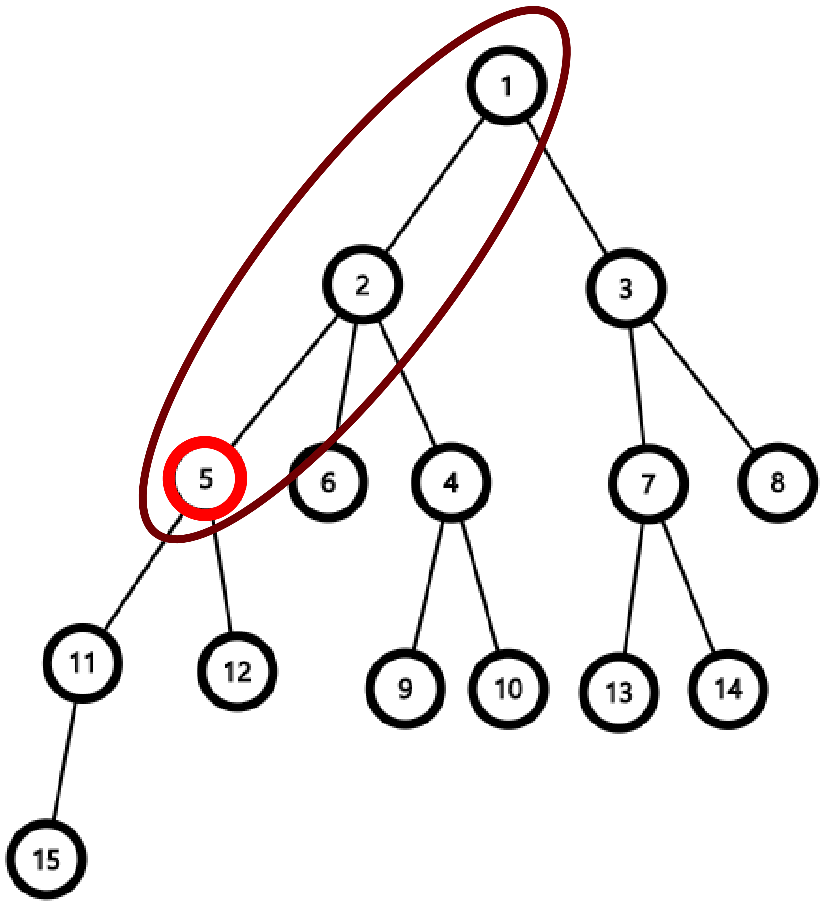
1. Tree이기 때문에 경로가 유일하지만, 단순 dfs는 사용 불가
2. LCA : rooted tree에서 임의의 두 노드 쌍(u,v)를 모두 자손으로 갖는 노드(조상) 중 가장 아래 있는 노드

- prerequisite

1. Tree이기 때문에 경로가 유일하지만, 단순 dfs는 사용 불가
2. LCA : rooted tree에서 임의의 두 노드 쌍(u,v)를 모두 자손으로 갖는 노드(조상) 중 가장 아래 있는 노드
3. root는 1로 고정이므로, 최소 공통 조상들 중 1로부터 가장 먼 점이 LCA

- ex) (12, 15)

공통 조상 : {1, 2, 5}
최소 공통 조상 : {5}



- prerequisite

1. Tree이기 때문에 경로가 유일하지만, 단순 dfs는 사용 불가
2. LCA : rooted tree에서 임의의 두 노드 쌍(u,v)를 모두 자손으로 갖는 노드(조상) 중 가장 아래 있는 노드
3. root는 1로 고정이므로, 최소 공통 조상들 중 1로부터 가장 먼 점이 LCA
4. 어떤 노드의 k번째 조상을 sparse table을 통해 정의

   $par[i][j]$:= i번 노드의 2^j번째 조상

   $par[i][j] = par\left[par[i][j-1]\right][j-1]$

- prerequisite

1. Tree이기 때문에 경로가 유일하지만, 단순 dfs는 사용 불가
2. LCA : rooted tree에서 임의의 두 노드 쌍(u,v)를 모두 자손으로 갖는 노드(조상) 중 가장 아래 있는 노드
3. root는 1로 고정이므로, 최소 공통 조상들 중 1로부터 가장 먼 점이 LCA
4. 어떤 노드의 k번째 조상을 sparse table을 통해 정의

   $par[i][j]$:= i번 노드의 2^j번째 조상

   $par[i][j] = par\left[par[i][j-1]\right][j-1]$
5. get $par[i][0]$: root를 시작점으로 하는 dfs

- prerequisite

1. Tree이기 때문에 경로가 유일하지만, 단순 dfs는 사용 불가
2. LCA : rooted tree에서 임의의 두 노드 쌍(u,v)를 모두 자손으로 갖는 노드(조상) 중 가장 아래 있는 노드
3. root는 1로 고정이므로, 최소 공통 조상들 중 1로부터 가장 먼 점이 LCA
4. 어떤 노드의 k번째 조상을 sparse table을 통해 정의

   $par[i][j]$:= i번 노드의 2^j번째 조상

   $par[i][j] = par\left[par[i][j-1]\right][j-1]$
5. get $par[i][0]$: root를 시작점으로 하는 dfs
6. 아래 위치한 노드의 2^j번째 조상에 대한 접근 : by depth

#11438 LCA 2

- get par[cur][0] by depth-first search

```
11 void get_parent_by_dfs(int curr, int prev = -1) {
12     if (prev != -1) dep[curr] = dep[prev] + 1;
13 
14     Dep = max(Dep, dep[curr]);
15     for (int next : adj[curr]) {
16         if (next == prev) continue;
17         par[next][0] = curr;
18         get_parent_by_dfs(next, curr);
19     }
20 }
21 
```

#12 : get dep[cur] except for root node

#17 : get par[i][0] except for 'root'

#11438 LCA 2

- filling sparse table

```cpp

void get_every_parents() {
    while (Dep)
        ++Exp, Dep >>= 1;

    for (int i = 1; i < Exp; ++i)
        for (int j = 0; j < V; ++j)
            par[j][i] = par[par[j][i - 1]][i - 1];
}

```

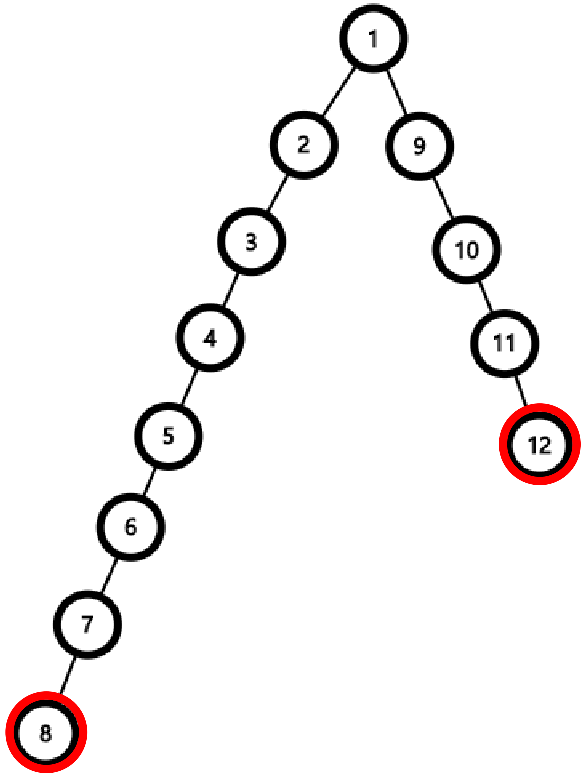
- **문제 해결 과정**

1. dfs를 이용한 2^0번째 부모, 각 node의 depth 구하기
2. 앞서 구한 정보를 이용해 2^k번째 부모 구하기
3. 각 노드에 대해 조상이 같아질 때까지 '끌어 올리기'

Ex) LCA(8, 12)

1. 노드 높이 맞추기

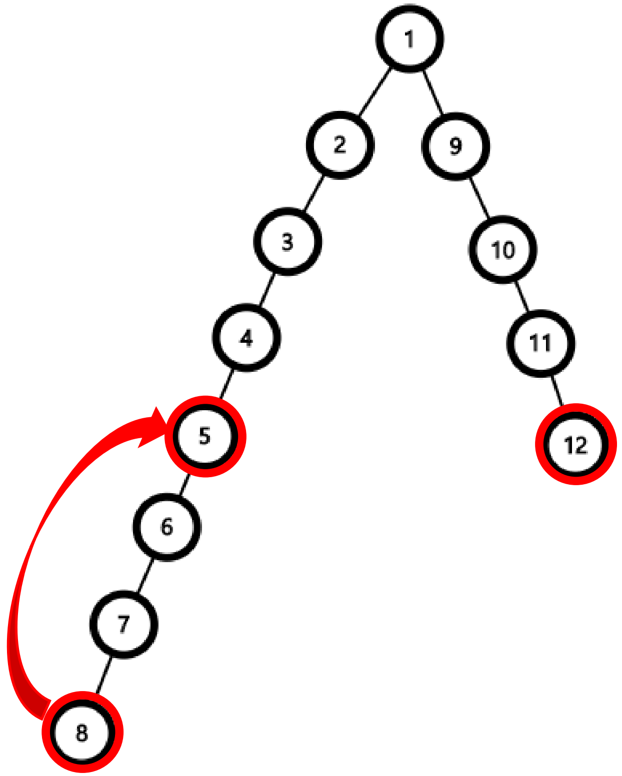
```
int LCA(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);

    for (int i = Exp; i >= 0; --i)
        if (dep[u] - dep[v] >= (1 << i))
            u = par[u][i];

    if (u == v) return u;

    for (int i = Exp; i >= 0; --i)
        if (par[u][i] != par[v][i]) {
            u = par[u][i];
            v = par[v][i];
        }
    return par[u][0];
}
```



Ex) LCA(8, 12)

1. 노드 높이 맞추기

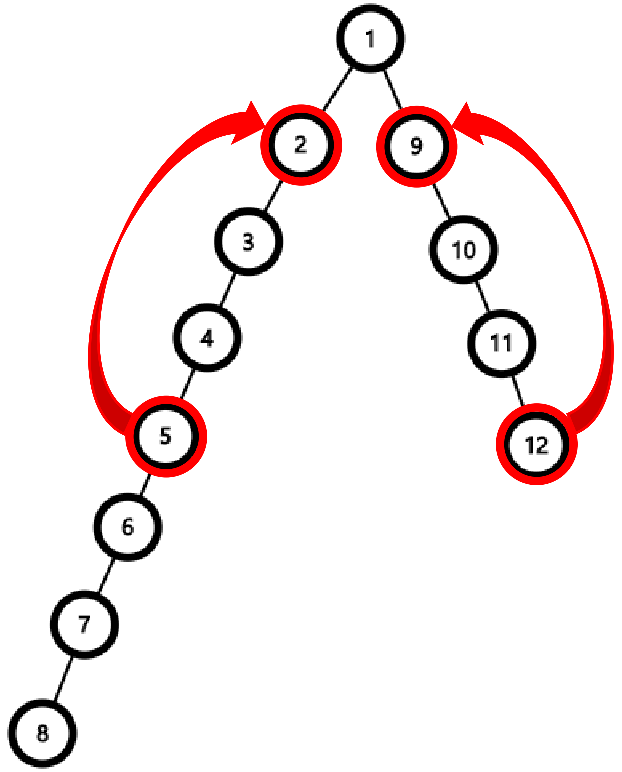
```cpp
int LCA(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);

    for (int i = Exp; i >= 0; --i)
        if (dep[u] - dep[v] >= (1 << i))
            u = par[u][i];

    if (u == v) return u;

    for (int i = Exp; i >= 0; --i)
        if (par[u][i] != par[v][i]) {
            u = par[u][i];
            v = par[v][i];
        }
    return par[u][0];
}
```



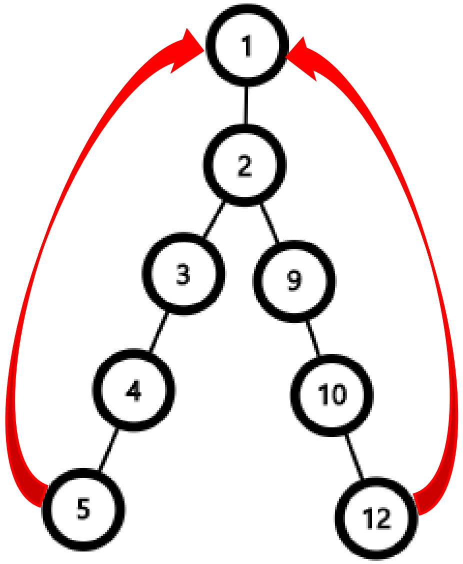
Ex) LCA(8, 12)

2. Common ancestor 찾기

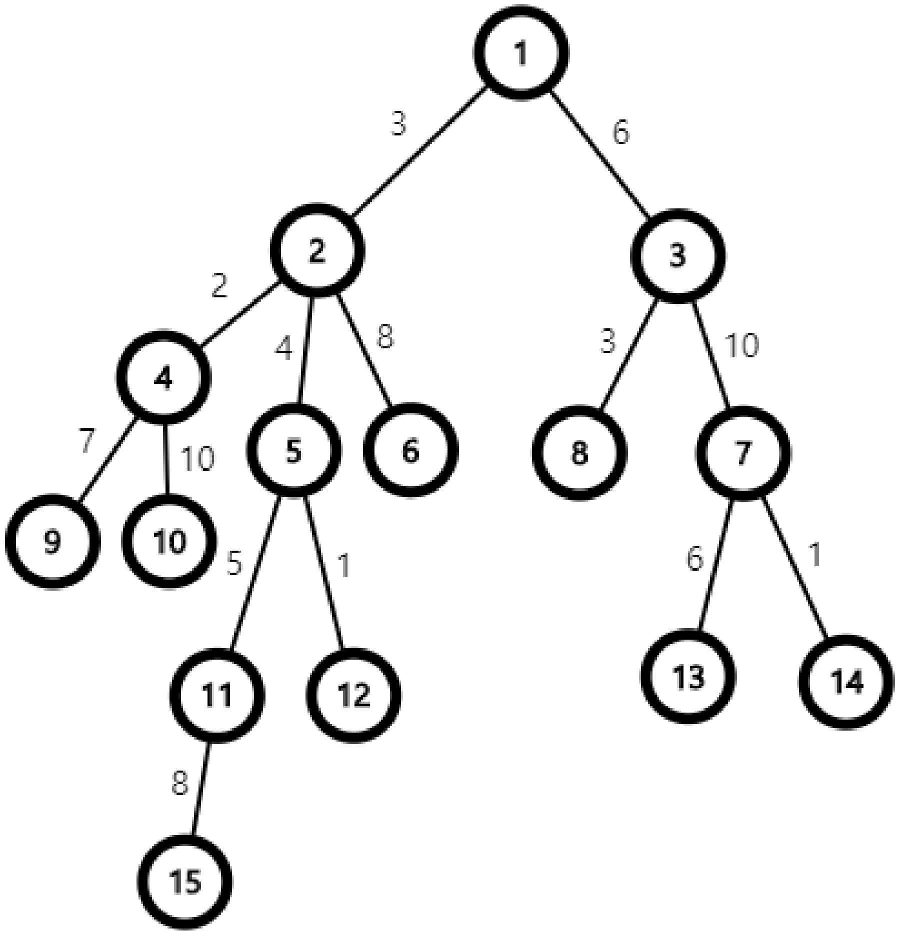
```
int LCA(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);

    for (int i = Exp; i >= 0; --i)
        if (dep[u] - dep[v] >= (1 << i))
            u = par[u][i];

    if (u == v) return u;

    for (int i = Exp; i >= 0; --i)
        if (par[u][i] != par[v][i]) {
            u = par[u][i];
            v = par[v][i];
        }
    return par[u][0];
}
```



- $par[i][u] == par[i][v]$가 유효하지 않은 경우

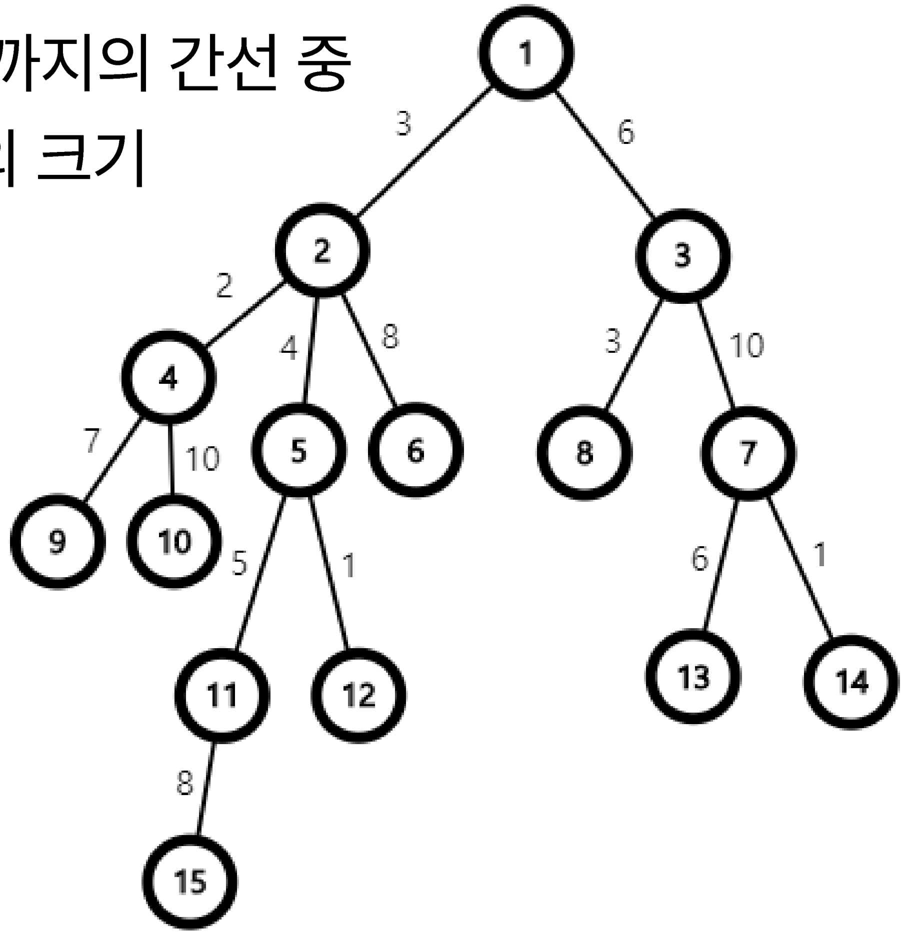


- 경로 상의 간선 가중치의 최대(최소)

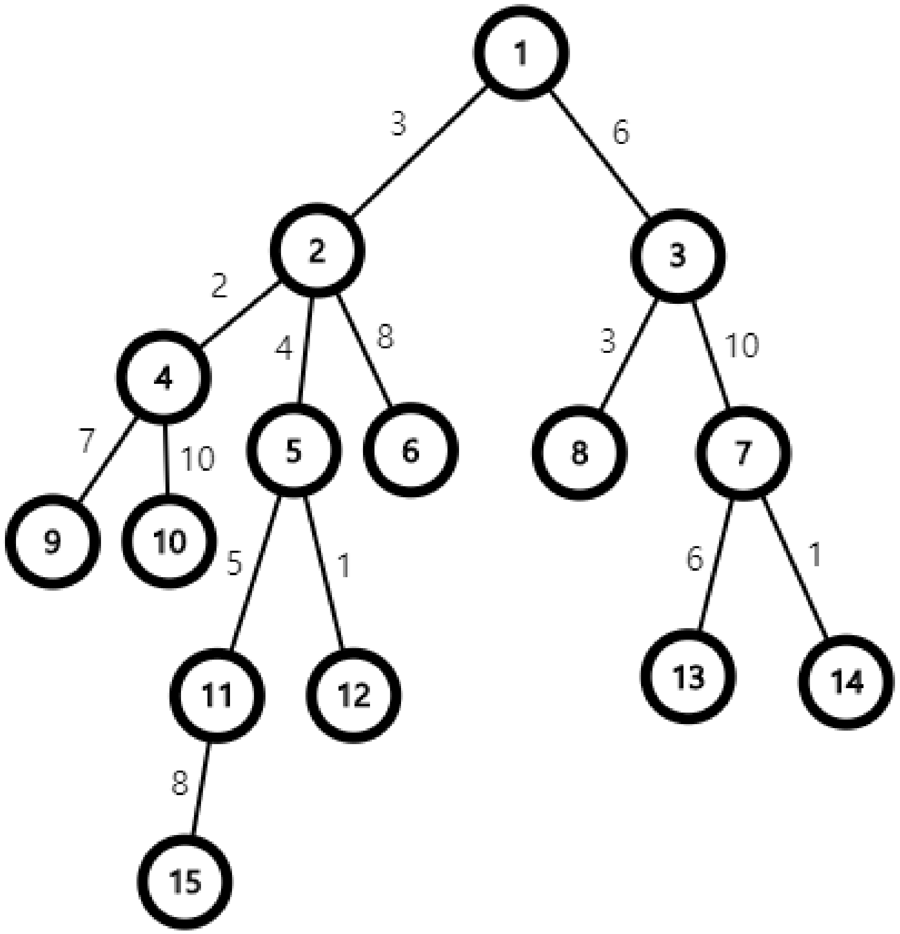


- 경로 상의 간선 가중치의 최대(최소)

$mx[i][j]$:= j의 2^i번째 부모까지의 간선 중 가중치가 최대인 것의 크기



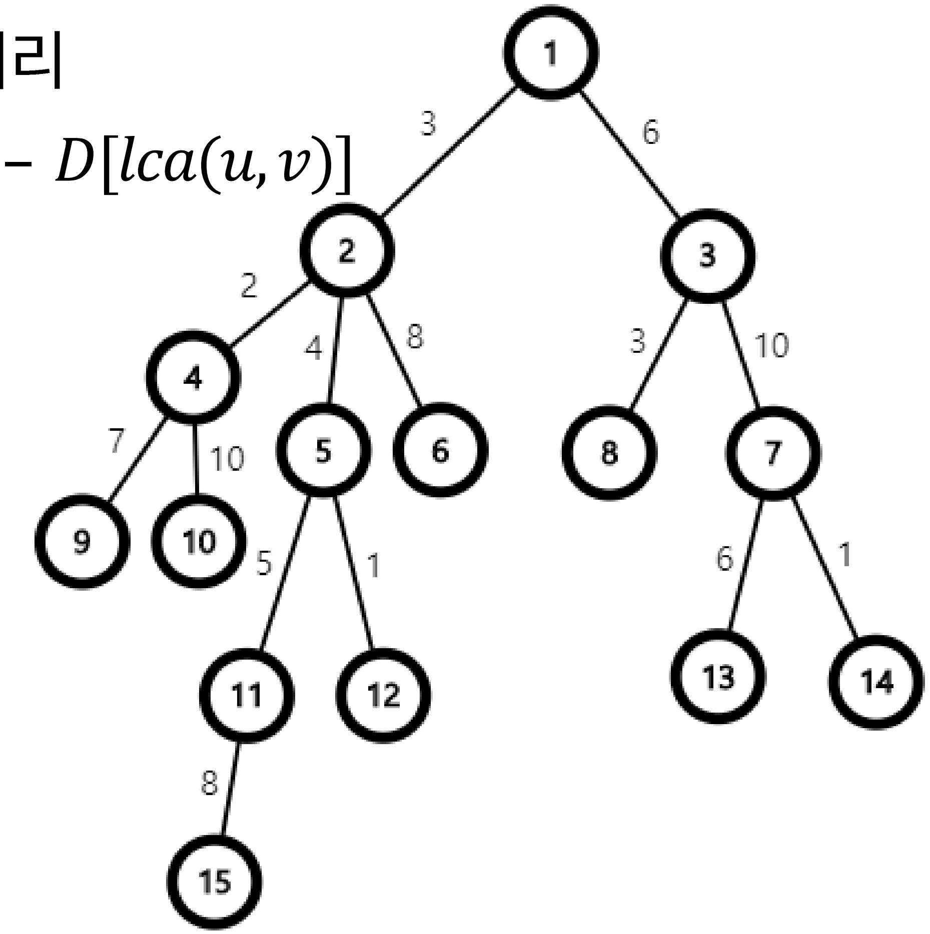
- 두 노드 사이의 거리



- 두 노드 사이의 거리

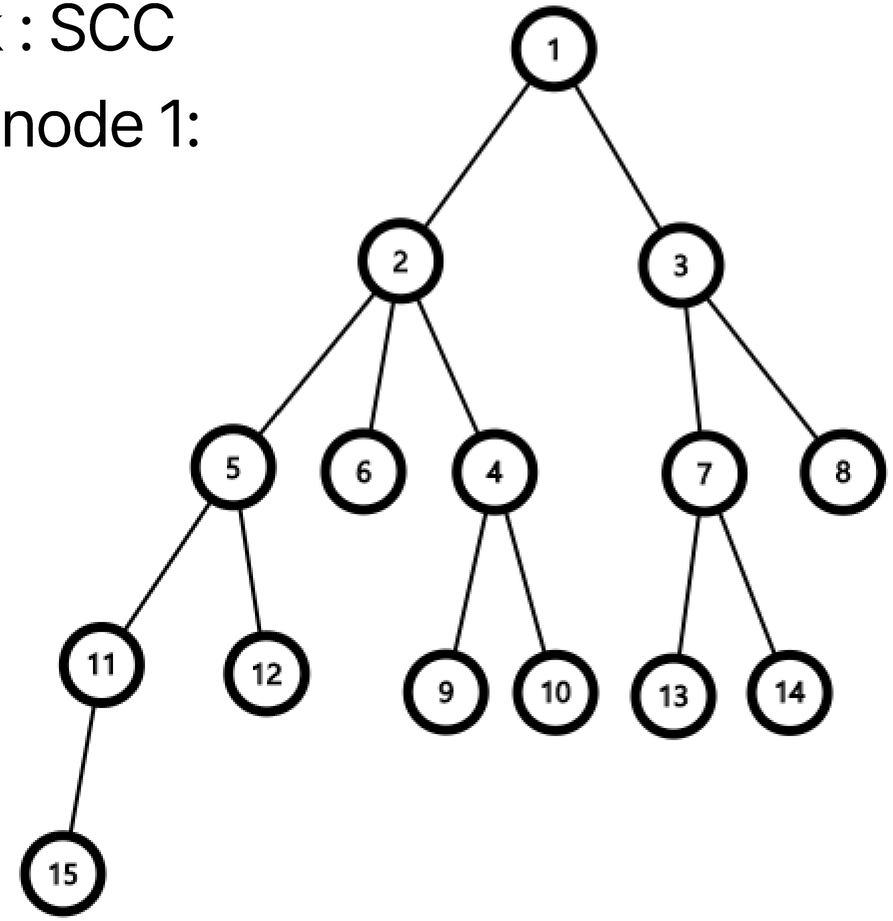
$D[i]$:= root에서 i까지의 거리

$dist(u, v) = D[u] + D[v] - D[lca(u, v)]$



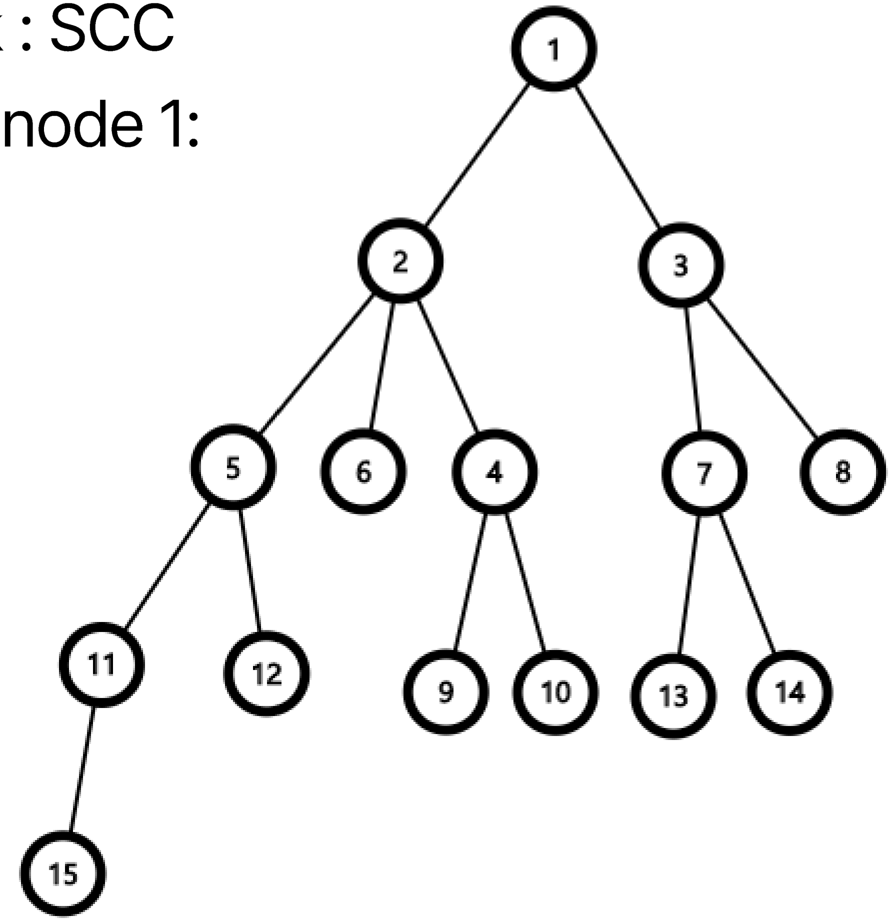
- Review for last week : SCC

Traverse by DFS from node 1:



- Review for last week : SCC

Traverse by DFS from node 1:

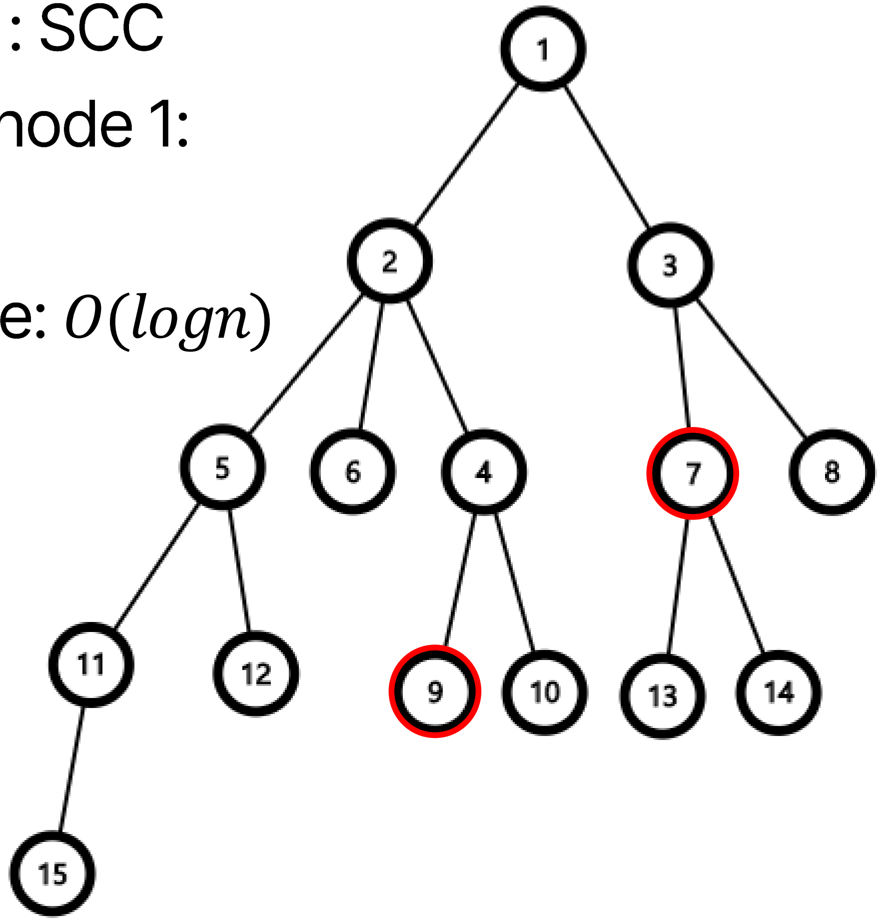


node	1	2	5	11	15	11	5	12	5	2	6	2	4	9	4	10	4	2	1	3	7	13	7	14	7	3	8	3	1
dep	1	2	3	4	5	4	3	4	3	2	3	2	3	4	3	4	3	2	1	2	3	4	3	4	3	2	3	2	1

- Review for last week : SCC

Traverse by DFS from node 1:

RMQ with segment tree: $O(logn)$

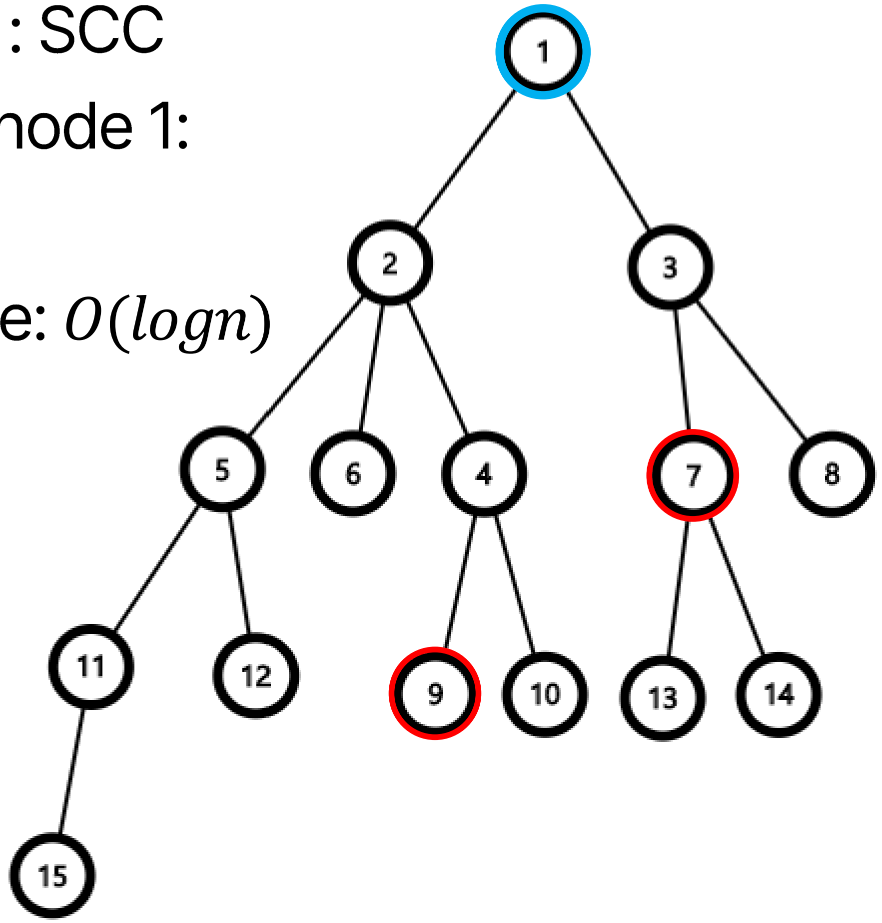


node	1	2	5	11	15	11	5	12	5	2	6	2	4	9	4	10	4	2	1	3	7	13	7	14	7	3	8	3	1
dep	1	2	3	4	5	4	3	4	3	2	3	2	3	4	3	4	3	2	1	2	3	4	3	4	3	2	3	2	1

- Review for last week : SCC

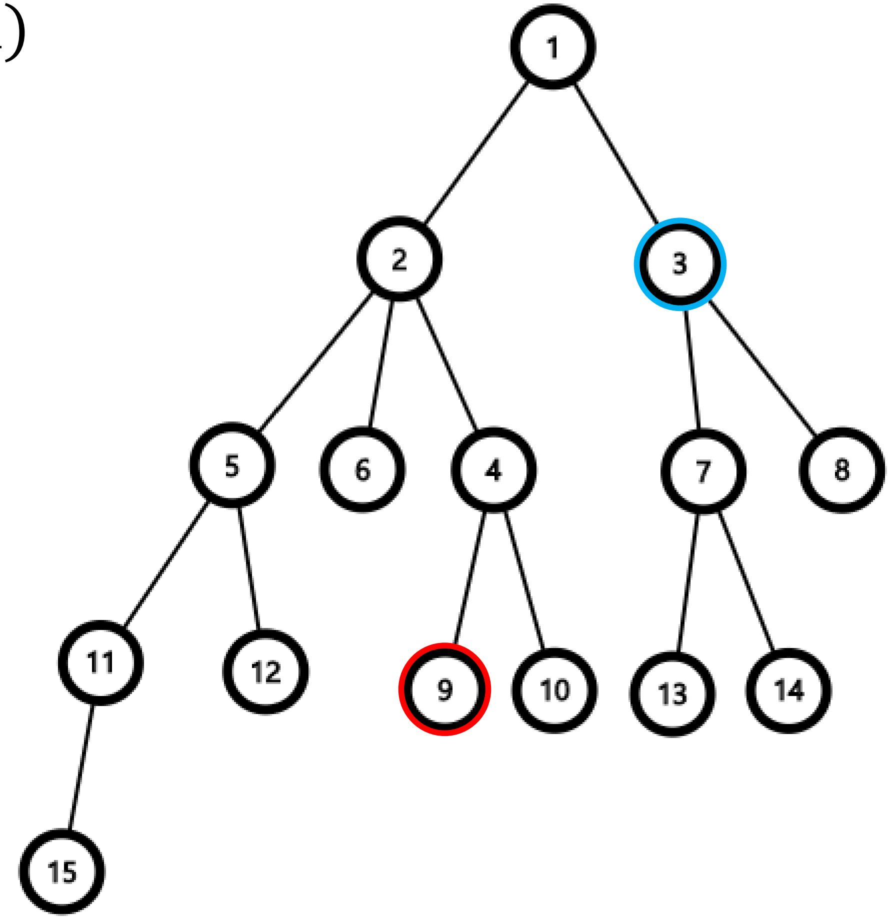
Traverse by DFS from node 1:

RMQ with segment tree: $O(logn)$



node	1	2	5	11	15	11	5	12	5	2	6	2	4	9	4	10	4	2	1	3	7	13	7	14	7	3	8	3	1
dep	1	2	3	4	5	4	3	4	3	2	3	2	3	4	3	4	3	2	1	2	3	4	3	4	3	2	3	2	1

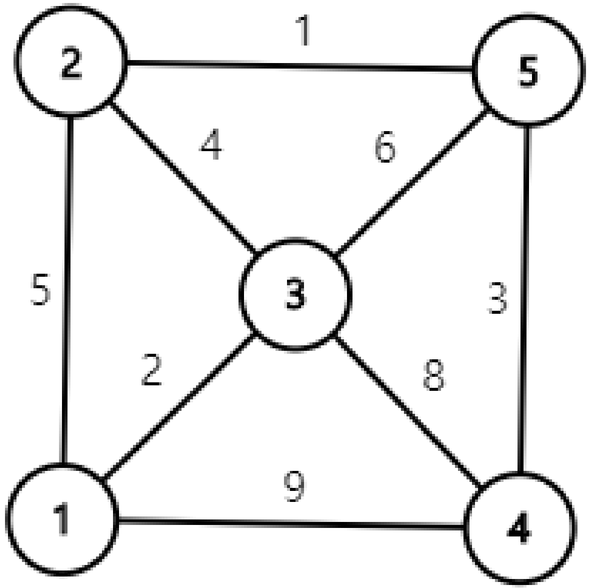
- Can get query in $O(1)$



node	1	2	5	11	15	11	5	12	5	2	6	2	4	9	4	10	4	2	1	3	7	13	7	14	7	3	8	3	1
dep	1	2	3	4	5	4	3	4	3	2	3	2	3	4	3	4	3	2	1	2	3	4	3	4	3	2	3	2	1

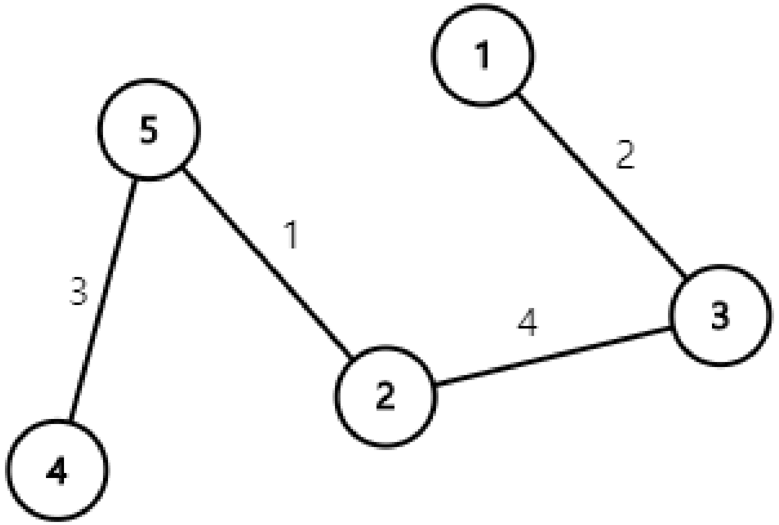
- $N(2 \leq \mathrm{N} \leq 200{,}000)$개의 정점과 $M(N-1 \leq M \leq 200{,}000)$개의 간선으로 구성된 undirected, weighted, connected simple graph G

- 각각의 간선 (u,v)에 대해 해당 간선을 포함하는 스패닝 트리 중 최소 가중치?



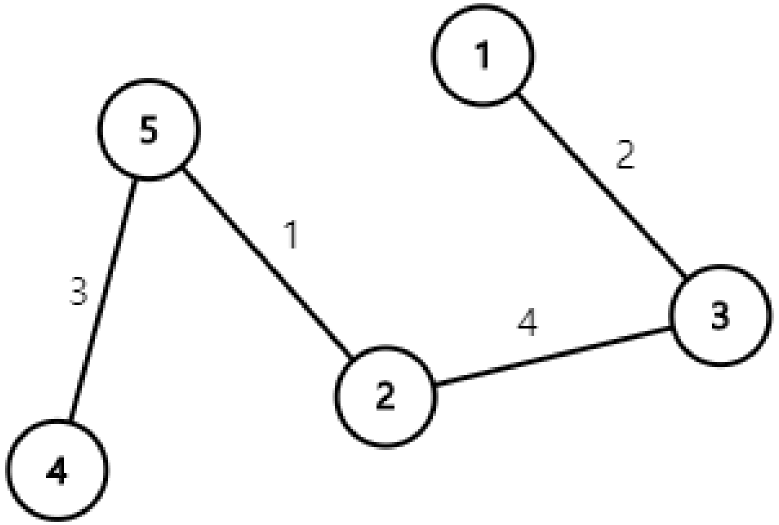
- **문제 해결 과정**

1. MST 구하기



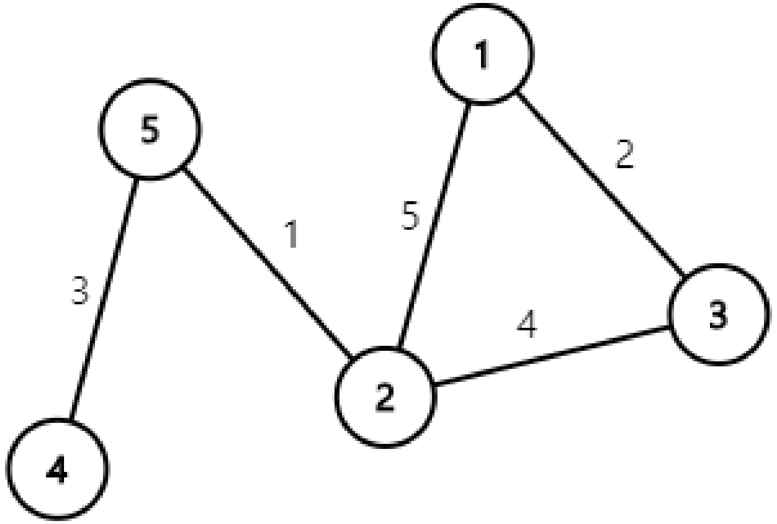
- **문제 해결 과정**

1. MST 구하기
2. 각 간선에 대해 케이스 분류
    - 기존 MST에 포함되는 간선의 경우 : MST의 가중치 합



- **문제 해결 과정**

1. MST 구하기
2. 각 간선에 대해 케이스 분류
    - 기존 MST에 포함되는 간선의 경우 : MST의 가중치 합
    - 기존 MST에 포함되지 않는 경우

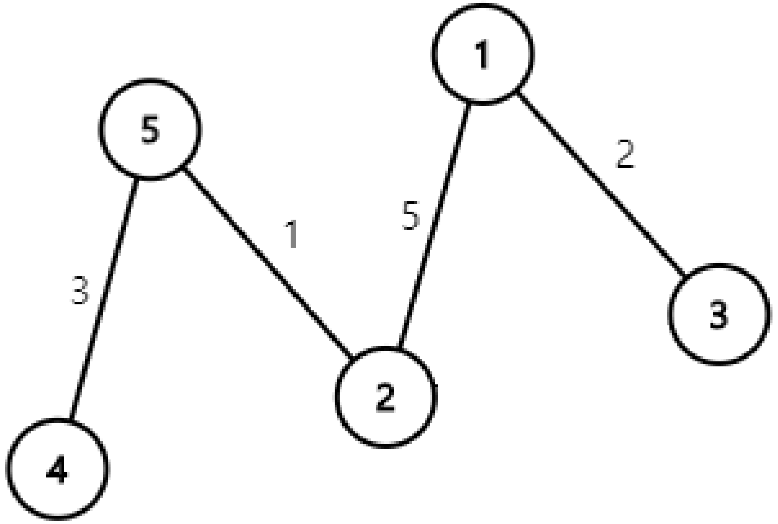


- **문제 해결 과정**

1. MST 구하기
2. 각 간선에 대해 케이스 분류
    - 기존 MST에 포함되는 간선의 경우 : MST의 가중치 합
    - 기존 MST에 포함되지 않는 경우

      간선을 구성하는 두 점 (u,v)에 대하여

      MST에서의 두 점 (u,v)상의 간선 중 가중치가 가장 큰 간선 제거



#17435 합성함수와 쿼리
#14942 개미

#3584 가장 가까운 공통 조상
#11438 LCA 2
#3176 도로 네트워크

#13511 트리와 쿼리 2
#12746 Traffic (Large)
#15481 그래프와 MST
#15480 LCA와 쿼리
#17399 트리의 외심