Chapter 04.
트리

Clip 01 | 트리 이론
트리의 이론과 용어 정리

Clip 02 | [11725] 트리의 부모 찾기
트리의 표현과 탐색

Clip 03 | [15681] 트리와 쿼리
서브 트리 탐색과 최적화

Clip 04 | [14267] 회사 문화 1
트리의 가공과 일괄 처리

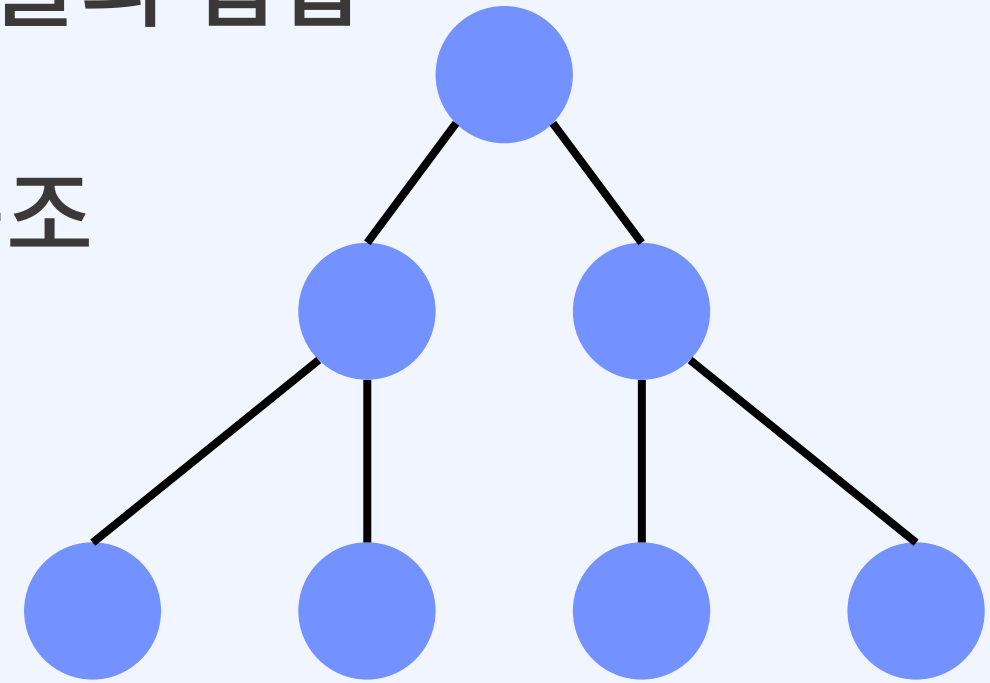
Ch04. 트리

1 이론과 용어 정리

트리

특징

- 계층적인 자료 구조
- 부모 – 자식 관계를 갖는 노드들의 집합
- 최상위 노드를 루트라고 함
- 사이클이 없는 비 순환적인 구조



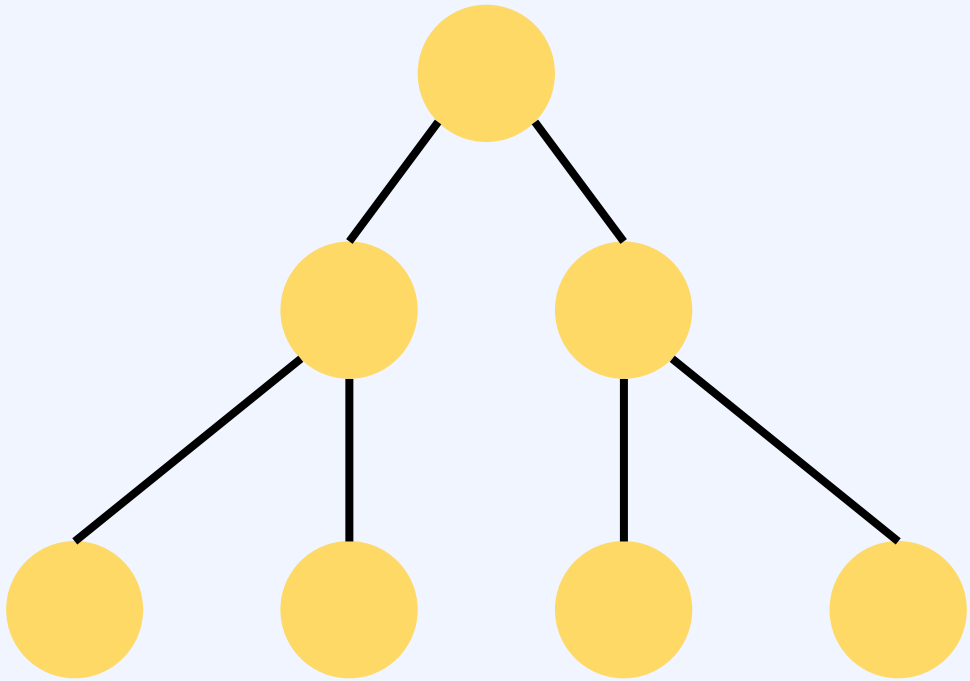
트리

용어

노드(Node)	트리를 구성하는 각각의 요소
루트(Root)	트리의 최상위 노드
부모 노드(Parent Node)	어떤 노드의 바로 위 노드
자식 노드(Child Node)	어떤 노드의 바로 아래 노드들
리프(Leaf)	자식 노드가 없는 노드
서브 트리(Subtree)	트리 내에서 특정 노드를 루트로 하는 트리
레벨(Level)	루트를 Level 0으로 하였을 때, 각 노드의 깊이

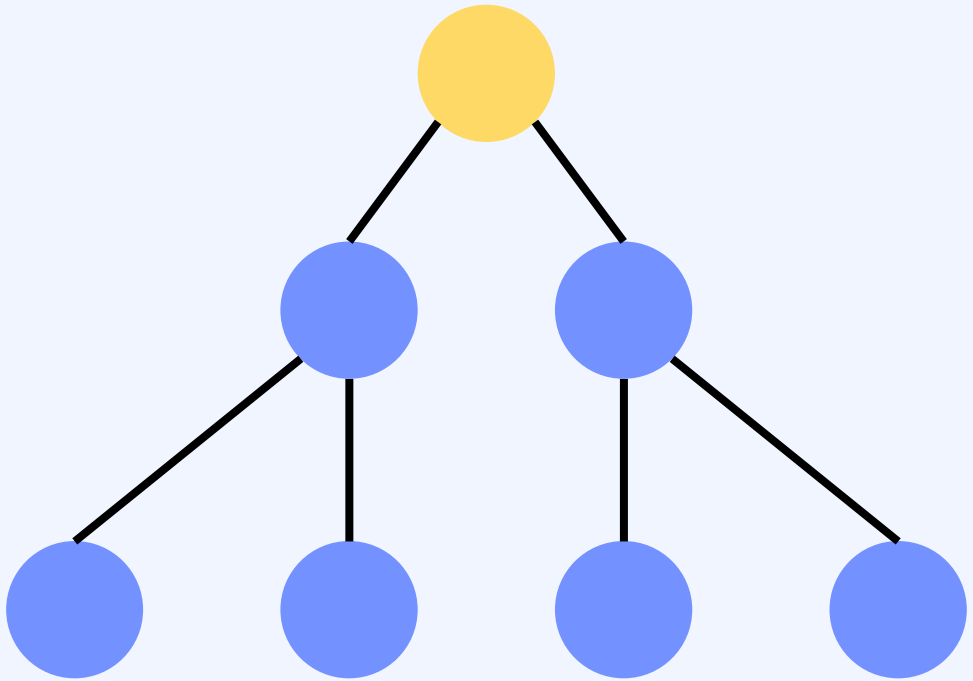
트리

노드(Node)	트리를 구성하는 각각의 요소
루트(Root)	트리의 최상위 노드
부모 노드(Parent Node)	어떤 노드의 바로 위 노드
자식 노드(Child Node)	어떤 노드의 바로 아래 노드들
리프(Leaf)	자식 노드가 없는 노드
서브 트리(Subtree)	트리 내에서 특정 노드를 루트로 하는 트리
레벨(Level)	루트를 Level 0으로 하였을 때, 각 노드의 깊이



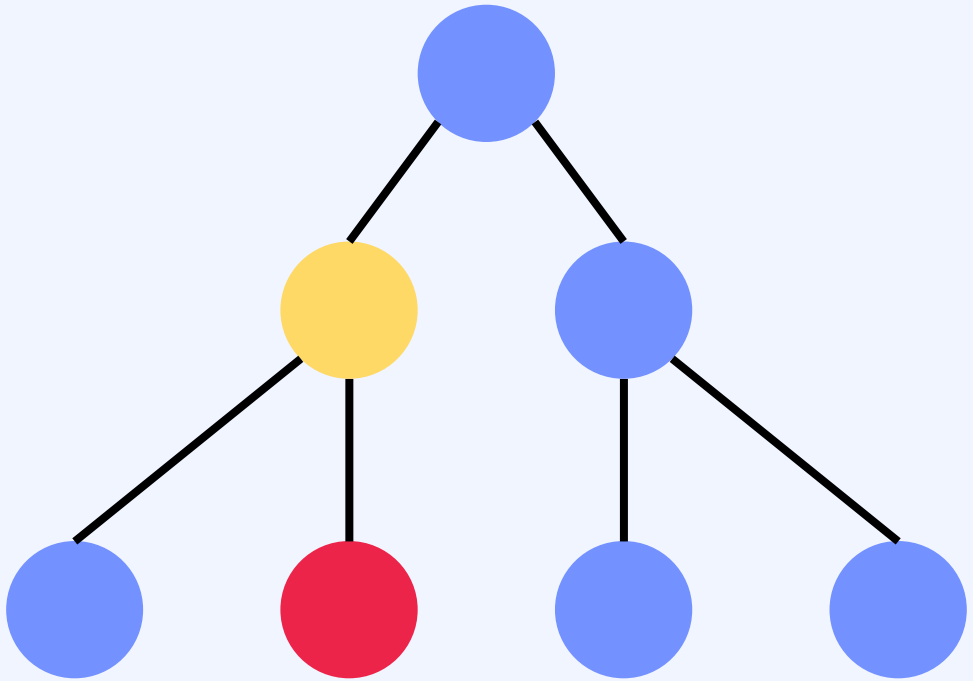
트리

노드(Node)	트리를 구성하는 각각의 요소
루트(Root)	트리의 최상위 노드
부모 노드(Parent Node)	어떤 노드의 바로 위 노드
자식 노드(Child Node)	어떤 노드의 바로 아래 노드들
리프(Leaf)	자식 노드가 없는 노드
서브 트리(Subtree)	트리 내에서 특정 노드를 루트로 하는 트리
레벨(Level)	루트를 Level 0으로 하였을 때, 각 노드의 깊이



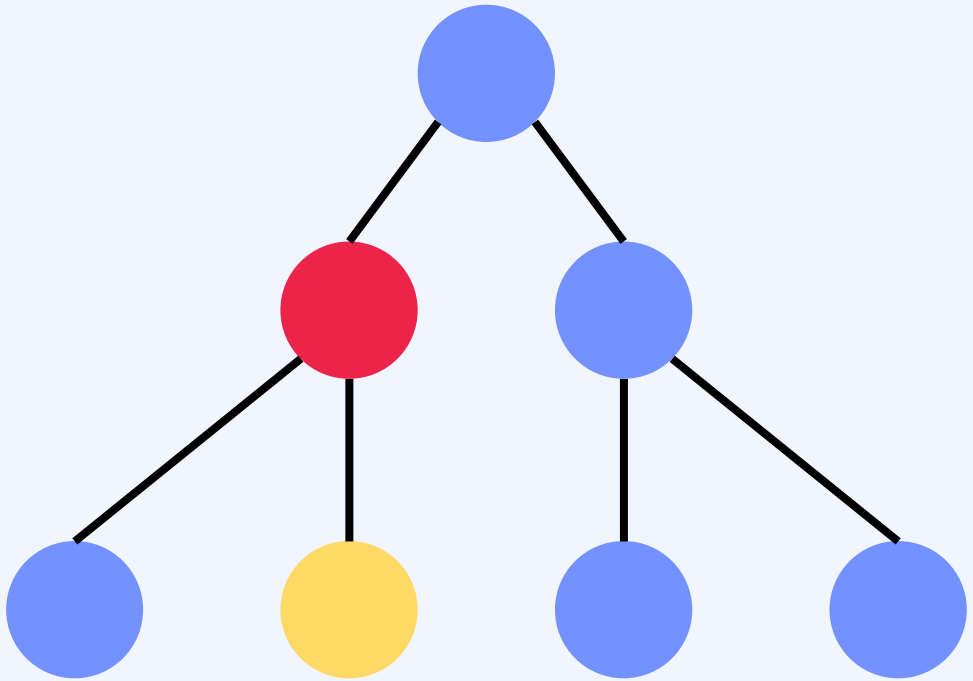
트리

노드(Node)	트리를 구성하는 각각의 요소
루트(Root)	트리의 최상위 노드
부모 노드(Parent Node)	어떤 노드의 바로 위 노드
자식 노드(Child Node)	어떤 노드의 바로 아래 노드들
리프(Leaf)	자식 노드가 없는 노드
서브 트리(Subtree)	트리 내에서 특정 노드를 루트로 하는 트리
레벨(Level)	루트를 Level 0으로 하였을 때, 각 노드의 깊이



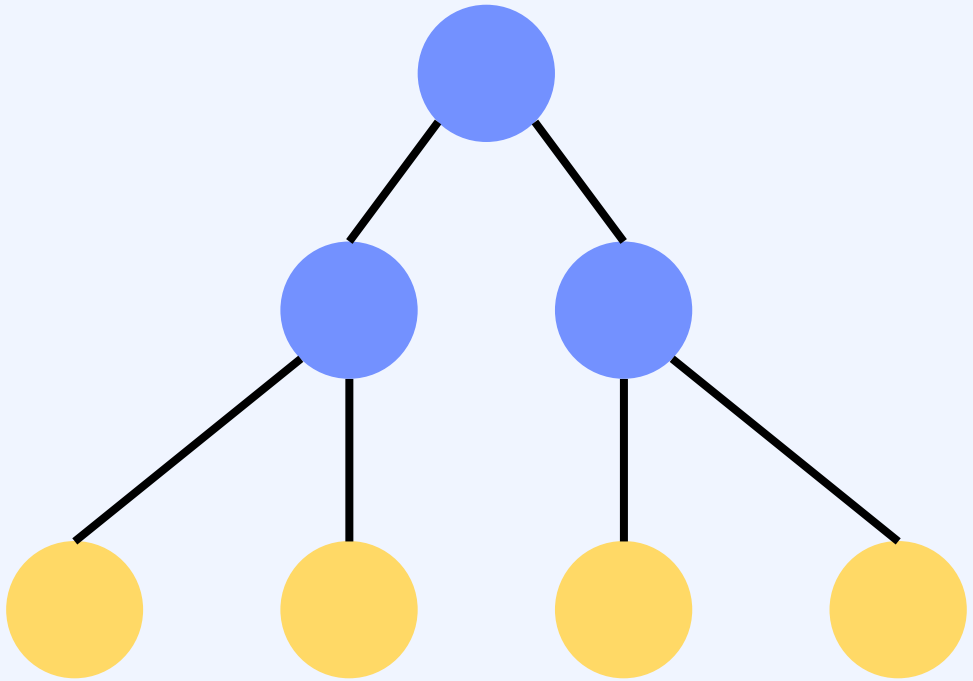
트리

노드(Node)	트리를 구성하는 각각의 요소
루트(Root)	트리의 최상위 노드
부모 노드(Parent Node)	어떤 노드의 바로 위 노드
자식 노드(Child Node)	어떤 노드의 바로 아래 노드들
리프(Leaf)	자식 노드가 없는 노드
서브 트리(Subtree)	트리 내에서 특정 노드를 루트로 하는 트리
레벨(Level)	루트를 Level 0으로 하였을 때, 각 노드의 깊이



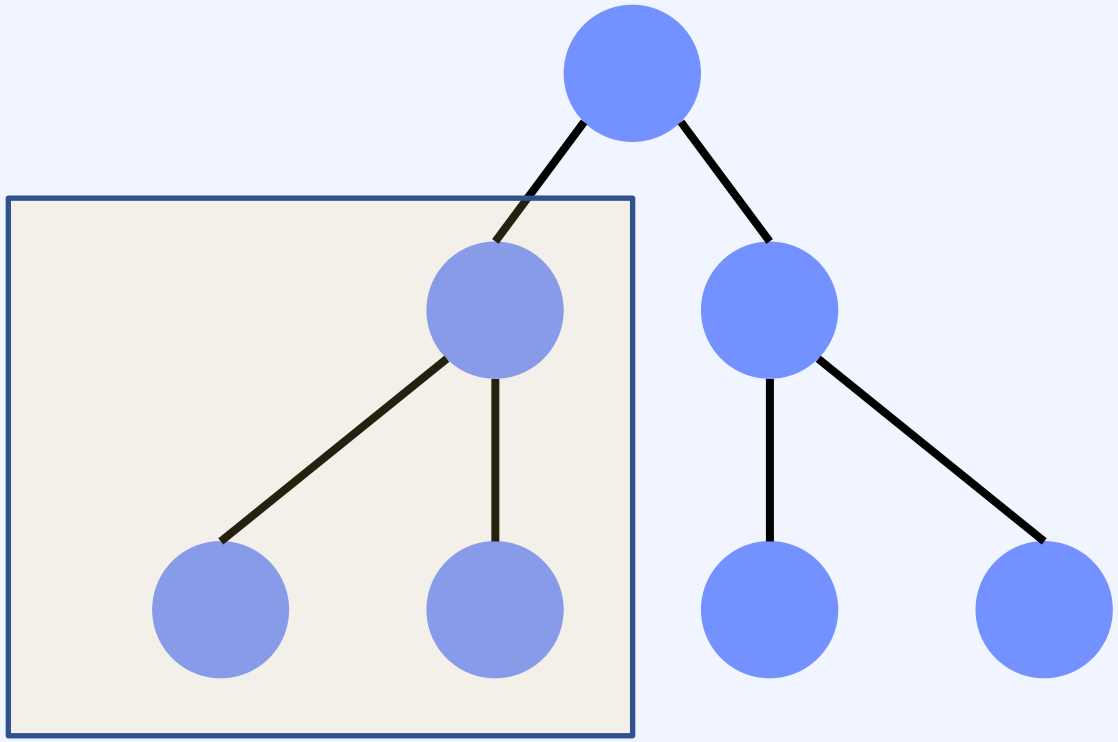
트리

노드(Node)	트리를 구성하는 각각의 요소
루트(Root)	트리의 최상위 노드
부모 노드(Parent Node)	어떤 노드의 바로 위 노드
자식 노드(Child Node)	어떤 노드의 바로 아래 노드들
리프(Leaf)	자식 노드가 없는 노드
서브 트리(Subtree)	트리 내에서 특정 노드를 루트로 하는 트리
레벨(Level)	루트를 Level 0으로 하였을 때, 각 노드의 깊이



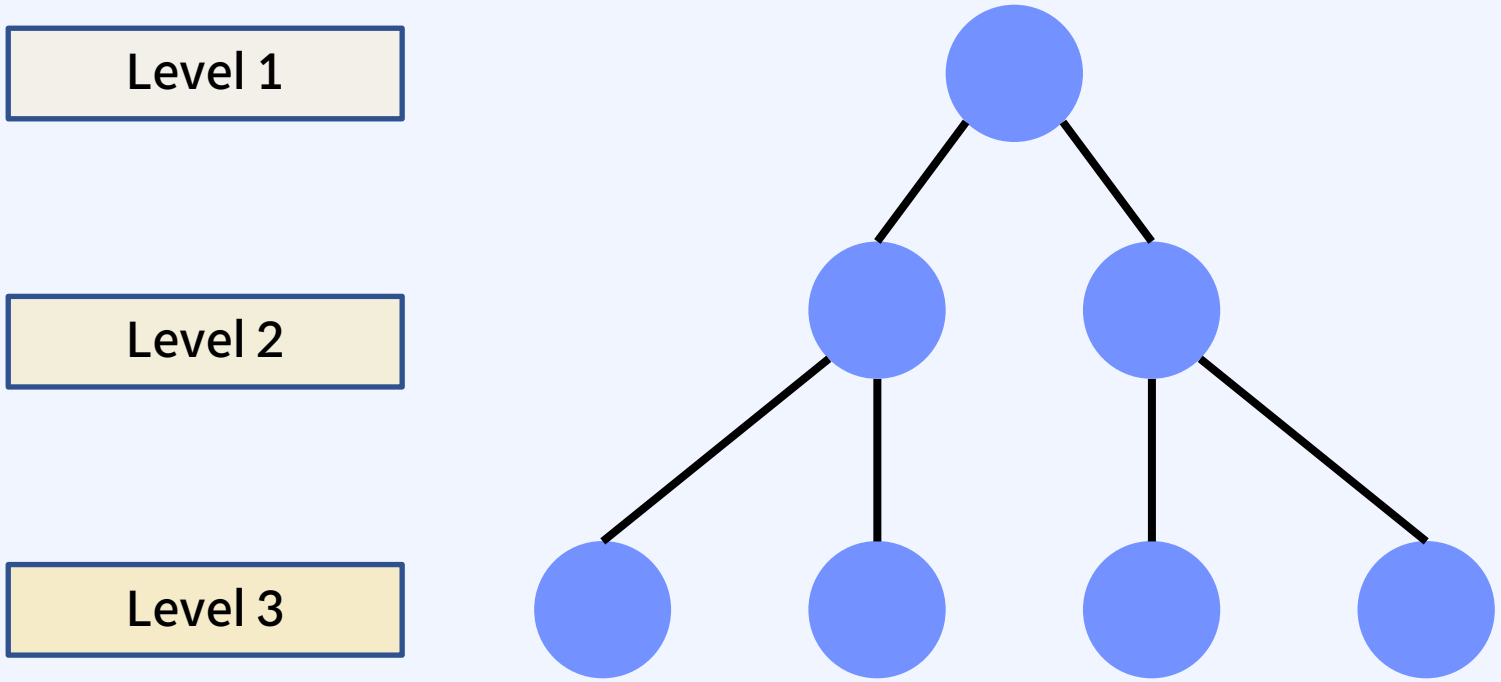
트리

노드(Node)	트리를 구성하는 각각의 요소
루트(Root)	트리의 최상위 노드
부모 노드(Parent Node)	어떤 노드의 바로 위 노드
자식 노드(Child Node)	어떤 노드의 바로 아래 노드들
리프(Leaf)	자식 노드가 없는 노드
서브 트리(Subtree)	트리 내에서 특정 노드를 루트로 하는 트리
레벨(Level)	루트를 Level 0으로 하였을 때, 각 노드의 깊이



트리

노드(Node)	트리를 구성하는 각각의 요소
루트(Root)	트리의 최상위 노드
부모 노드(Parent Node)	어떤 노드의 바로 위 노드
자식 노드(Child Node)	어떤 노드의 바로 아래 노드들
리프(Leaf)	자식 노드가 없는 노드
서브 트리(Subtree)	트리 내에서 특정 노드를 루트로 하는 트리
레벨(Level)	루트를 Level 0으로 하였을 때, 각 노드의 깊이



Ch04. 트리

2. [11725] 트리의 부모 찾기

BOJ11725: 트리의 부모 찾기

문제

루트 없는 트리가 주어진다. 이때, 트리의 루트를 1이라고 정했을 때, 각 노드의 부모를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 노드의 개수 N (2 ≤ N ≤ 100,000)이 주어진다. 둘째 줄부터 N-1개의 줄에 트리 상에서 연결된 두 정점이 주어진다.

출력

첫째 줄부터 N-1개의 줄에 각 노드의 부모 노드 번호를 2번 노드부터 순서대로 출력한다.

노드 간의 연결 관계가 입력으로 주어진다
정보를 이용해 트리를 저장하고, 탐색하는 문제

BOJ11725: 트리의 부모 찾기

tree[i] : i 번 노드에 연결된 다른 노드

예제입력
7 1 6 6 3 3 5 4 1 2 4 4 7

1	6	4	
2	4		
3	6	5	
4	1	2	7
5	3		
6	1	3	
7	4		

BOJ11725: 트리의 부모 찾기

tree[i] : i 번 노드에 연결된 다른 노드

1	6	4	
2	4		
3	6	5	
4	1	2	7
5	3		
6	1	3	
7	4		

1번 노드에 연결된 노드
- 6번 노드
- 4번 노드

- tree[1].size() : 2

BOJ11725: 트리의 부모 찾기

```java
public class Main {
    public static List<Integer>[] tree;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        tree = new ArrayList[n + 1];
        for (int i = 0; i < n + 1; i++) {
            tree[i] = new ArrayList<>();
        }
        for (int i = 0; i < n - 1; i++) {
            int node1 = sc.nextInt();
            int node2 = sc.nextInt();
            tree[node1].add(node2);
            tree[node2].add(node1);
        }
}
```

BOJ11725: 트리의 부모 찾기

- 문제의 조건에 의해 1번 노드가 루트 노드로 지정되었다
- 1번 노드부터 연결된 노드를 탐색하며 부모를 매핑하자
- 단, 이미 탐색했던 노드를 재방문해서
  사이클이 발생하지 않도록 한다
  - check[] 를 활용해 특정 노드의 방문 여부를 기록한다

BOJ11725: 트리의 부모 찾기

```
public static void find(int node) {
    check[node] = true;
    for (int i = 0; i < tree[node].size(); i++) {
        int child = tree[node].get(i);
        if (!check[child]) {
            parents[child] = node;
            find(child);
        }
    }
}
```

1번 노드에 연결된 {i}번째 child는?

1	6	4	
2	4		
3	6	5	
4	1	2	7
5	3		
6	1	3	
7	4		

BOJ11725: 트리의 부모 찾기

```
public static void find(int node) {
    check[node] = true;
    for (int i = 0; i < tree[node].size(); i++) {
        int child = tree[node].get(i);
        if (!check[child]) {
            parents[child] = node;
            find(child);
        }
    }
}
```

1번 노드에 연결된 {i}번째 child는?
→ node[1].get(i)

node[1]			
1	6	4	
2	4		
3	6	5	
4	1	2	7
5	3		
6	1	3	
7	4		

BOJ11725: 트리의 부모 찾기

```
public static void find(int node) {
    check[node] = true;
    for (int i = 0; i < tree[node].size(); i++) {
        int child = tree[node].get(i);
        if (!check[child]) {
            parents[child] = node;
            find(child);
        }
    }
}
```

child 의 부모 노드는
매개변수로 주어진
{node} 값임을 알 수 있다

1. 정답을 기록할 parent[]
   배열에 정보를 저장

2. 불필요한 반복 탐색을
   하지 않도록 체크

Ch04. 트리

3 [15681] 트리와 쿼리

BOJ15681: 트리와 쿼리

문제 요약

- 특정 정점 U를 루트로 하는 서브트리에 속한 정점의 수를 출력하는 쿼리에 답하라
- 주어진 트리는 항상 올바른 트리임이 보장된다.
- 입력
  - $2 \leq N \leq 100,000$
  - $1 \leq R \leq N$
  - $1 \leq Q \leq 100,000$

BOJ15681: 트리와 쿼리

앞서 풀이했던
BOJ11725:트리의 부모 찾기 문제와 유사하다

단, 이번에는 자신의 자식들(서브트리)에 속한
정점의 수를 구해야 한다

동일하게 그래프 자료구조를 생성하자

```
tree = new ArrayList[n + 1];
for(int i = 0; i < n - 1; i++) {
    int u = sc.nextInt(), v = sc.nextInt();
    tree[u].add(v);
    tree[v].add(u);
}
```

BOJ15681: 트리와 쿼리

자신의 자식 노드의 수를 계산하는
getSum() 함수가 있다고 가정해보자

문제에서 트리는 한번 입력으로 주어진 뒤 불변하다
getSum() 함수는 여러 번 호출해도 동일한 결과를 출력한다
→ 수학적 의미의 함수

BOJ15681: 트리와 쿼리

따라서, {i}번째 노드의 자식의 수를 구했다면
이를 저장해두고

다시 호출되었을 때 재귀가 아닌
저장된 값을 불러오도록 작성할 수 있다

```
public static int[] sum;
public static int getSum(int node) {
    if(sum[node] != 0) return sum[node];
    // do something
}
```

BOJ15681: 트리와 쿼리

1. 반복되는 탐색이 되지 않도록 check[] 를 통해 방문여부를 기록한다
2. {i}를 루트로 하는 서브 트리 노드의 합은 1로 초기화하고 탐색을 진행한다 (자기 자신이 존재하므로)
3. child 를 루트로 하는 루트로 하는 노드의 수를 재귀로 구하고, 자신에게 더한다
   - 해당 값이 이미 구해졌다면 sum[] 으로부터 획득한다
   - 처음 탐색을 한다면 재귀함수를 반복 호출하며 계산한다

BOJ15681: 트리와 쿼리

1. 반복되는 탐색이 되지 않도록 check[] 를 통해 방문여부를 기록한다 1
2. {i}를 루트로 하는 서브 트리 노드의 합은 1 로 초기화하고 탐색을 진행한다 (자기 자신이 존재하므로) 2
3. child 를 루트로 하는 루트로 하는 노드의 수를 재귀로 구하고, 자신에게 더한다 3
   - 해당 값이 이미 구해졌다면 sum[] 으로부터 획득한다 4
   - 처음 탐색을 한다면 재귀함수를 반복 호출하며 계산한다 1 2 3

```
public static int getSum(int node) {
4   if(sum[node] != 0) return sum[node];
1   check[node] = true;
2   int result = 1;
    for (int child : tree[node]) {
        if(!check[child]) {
3       result += getSum(child);
        }
    }
    return sum[node] = result;
}
```

Ch04. 트리

4. [14267] 회사 문화 1

BOJ14267: 회사 문화 1

문제

영선회사에는 매우 좋은 문화가 있는데, 바로 상사가 직속 부하를 칭찬하면 그 부하가 부하의 직속 부하를 연쇄적으로 칭찬하는 내리 칭찬이 있다. 즉, 상사가 한 직속 부하를 칭찬하면 그 부하의 모든 부하들이 칭찬을 받는다.

모든 칭찬에는 칭찬의 정도를 의미하는 수치가 있는데, 이 수치 또한 부하들에게 똑같이 칭찬 받는다.

직속 상사와 직속 부하관계에 대해 주어지고, 칭찬에 대한 정보가 주어질 때, 각자 얼마의 칭찬을 받았는지 출력하시오,

입력

첫째 줄에는 회사의 직원 수 n명, 최초의 칭찬의 횟수 m이 주어진다. 직원은 1번부터 n번까지 번호가 매겨져 있다. ($2 \le n, m \le 100,000$)

둘째 줄에는 직원 n명의 직속 상사의 번호가 주어진다. 직속 상사의 번호는 자신의 번호보다 작으며, 최종적으로 1번이 사장이다. 1번의 경우, 상사가 없으므로 -1이 입력된다.

다음 m줄에는 직속 상사로부터 칭찬을 받은 직원 번호 i, 칭찬의 수치 w가 주어진다. ($2 \le i \le n, 1 \le w \le 1,000$)

사장은 상사가 없으므로 칭찬을 받지 않는다.

직속 부하들의 연결관계를 파악하고 칭찬을 기록해야 한다

BOJ14267: 회사 문화 1

입력은 직속상사의 노드 번호가 주어진다.
탐색은 {상사} → {부하} 방향으로 진행해야 하므로
트리로 자료를 저장한다

```
for (int i = 1; i <= n; i++) {
    tree[i] = new ArrayList<>();
    parent[i] = sc.nextInt();

    if (parent[i] != -1) {
        tree[parent[i]].add(i);
    }
}
```

{i}의 상사가 입력으로 주어지므로

1. 입력을 parent[i]에 받고

2. tree[parent[i]] 에
   부하의 노드 번호 {i} 를 넣는다

BOJ14267: 회사 문화 1

칭찬을 기록하는 방식

- 칭찬이 발생하다 모든 부하들을 찾아다니기?
  - 동일 조직에 칭찬이 여러 번 부여되면
    동일한 경로를 반복 탐색한다

- 나에게 발생한 칭찬을 기록해 두고
  모든 입력을 다 받은 뒤, 부하직원에게 한번에 뿌려주면
  탐색 횟수를 줄일 수 있다.

BOJ14267: 회사 문화 1

직원에게 발생한 칭찬을 기록해 두고
모든 입력을 다 받은 뒤, 부하직원에게 한번에 뿌려주면
탐색 횟수를 줄일 수 있다.

```
for (int i = 0; i < m; i++) {
    int employee = sc.nextInt();
    int point = sc.nextInt();
    like[employee] += point;
}
```

BOJ14267: 회사 문화 1

나(node)의 부하직원(child)들을 트리에서 조회하고
부하들에게 누적해 두었던 칭찬 수치를 더해준다 1

탐색 대상을 부하직원으로 변경하여 재귀를 수행한다 2

```
public static void next(int node) {
    for (int i = 0; i < tree[node].size(); i++) {
        int child = tree[node].get(i);
        like[child] += like[node]; 1
        next(child); 2
    }
}
```