Chapter. 02

알고리즘

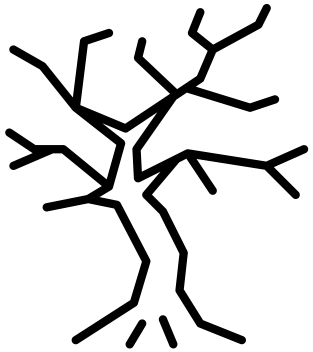
트리
Tree

FAST CAMPUS
ONLINE
알고리즘 공채 대비반!

강사. 류호석

Chapter. 02

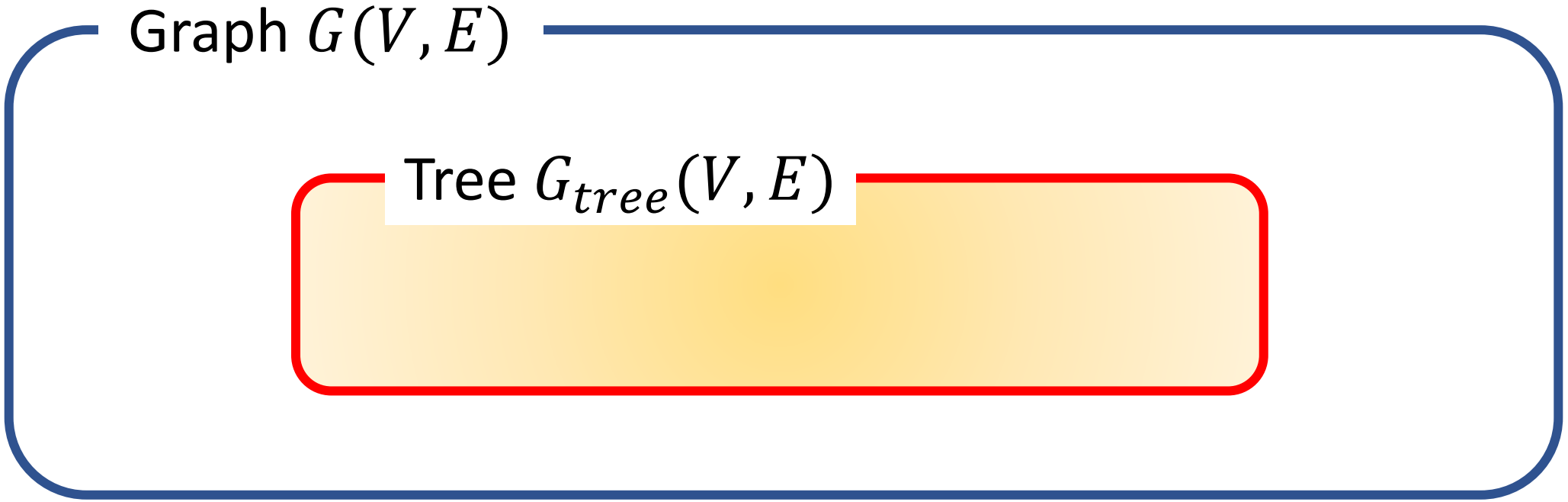
알고리즘 트리(Tree)



FAST CAMPUS
ONLINE
류호석 강사.

I 트리(Tree)란?

트리(Tree) := Graph의 특수한 형태



I 트리(Tree)란?

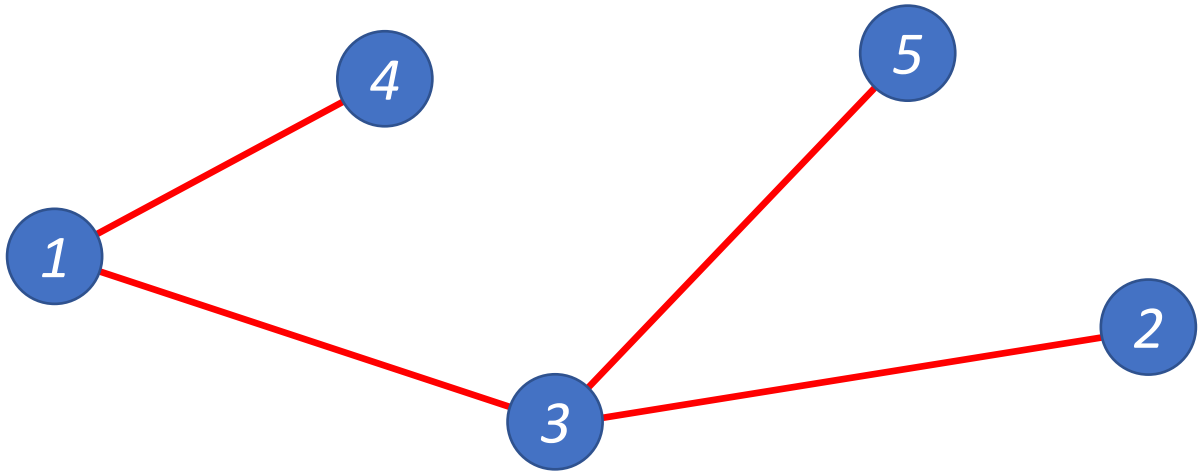
Tree := V + E + 다음의 특성(中 2개 이상 만족)

1. 모두가 연결되어 있는 그래프

   ➔ 어떤 두 점을 골라도 간선을 타고 사이를 이동 가능

2. 사이클이 존재하지 않음

3. 정점 개수 |V| = 간선 개수 |E| + 1



I 트리(Tree)란?

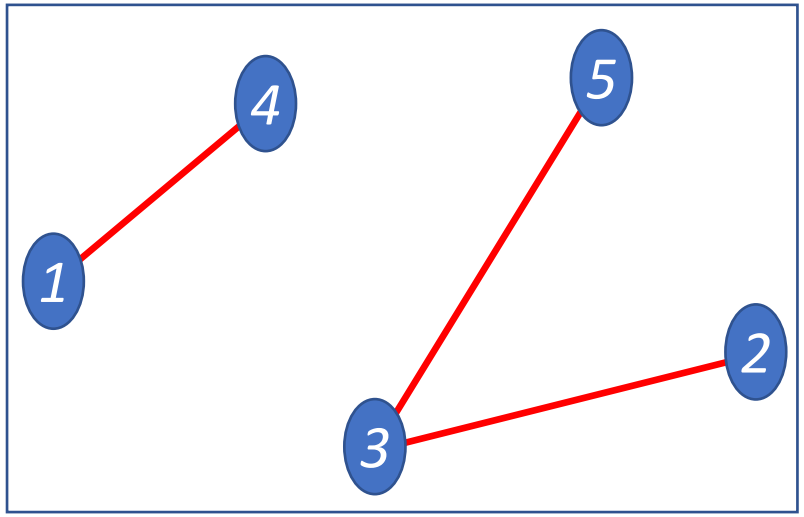
Tree := V + E + 다음의 특성(中 2개 이상 만족)

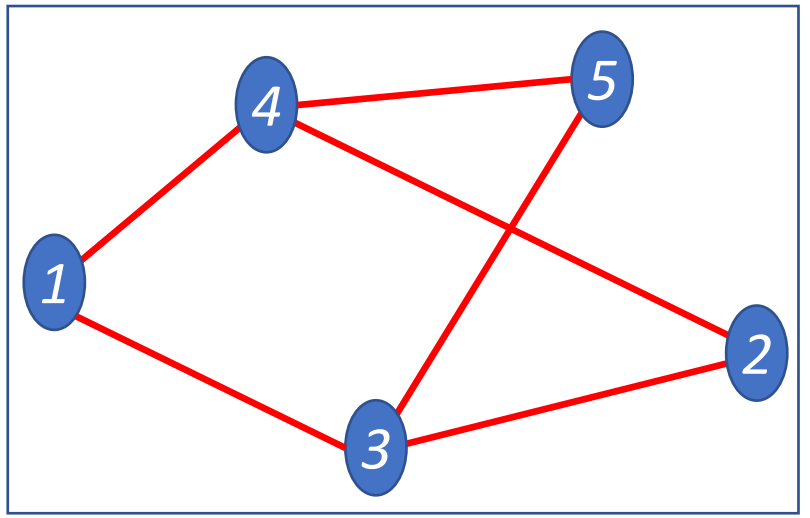
1. 모두가 연결되어 있는 그래프

   ➔ 어떤 두 점을 골라도 간선을 타고 사이를 이동 가능

2. 사이클이 존재하지 않음

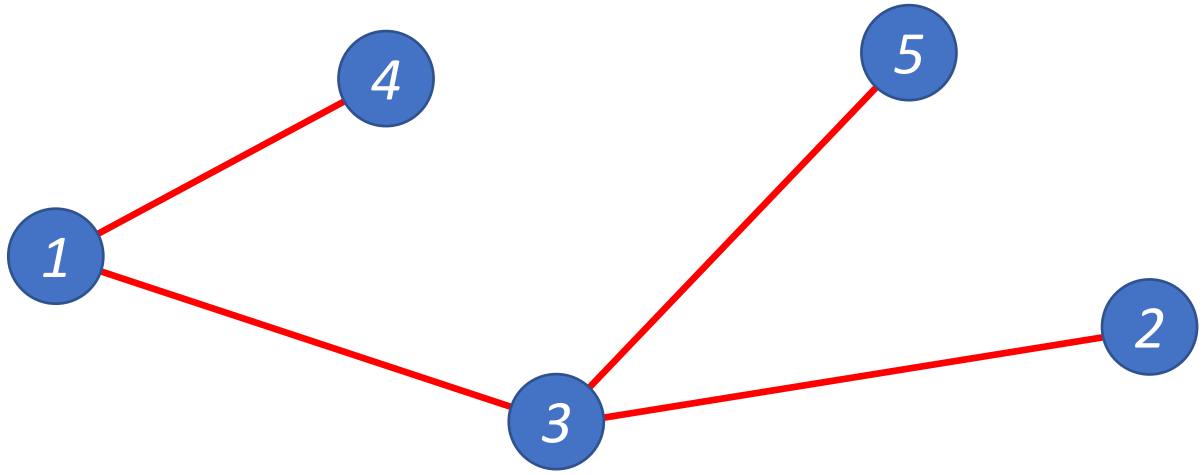
3. 정점 개수 |V| = 간선 개수 |E| + 1



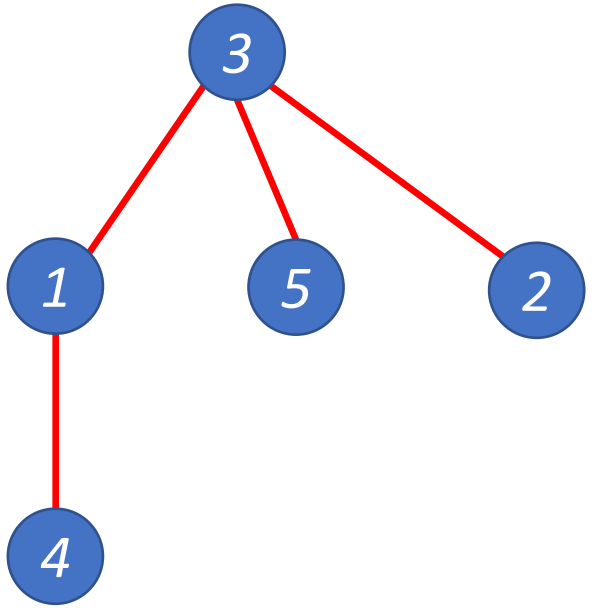


I 트리(Tree)란?

Tree



Rooted Tree

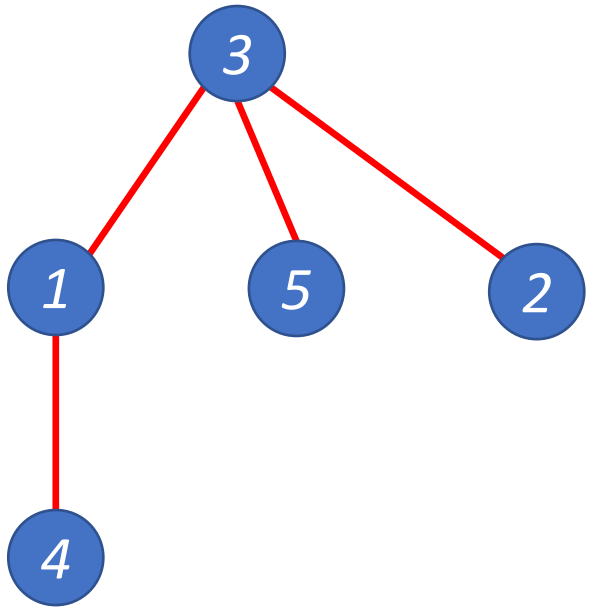


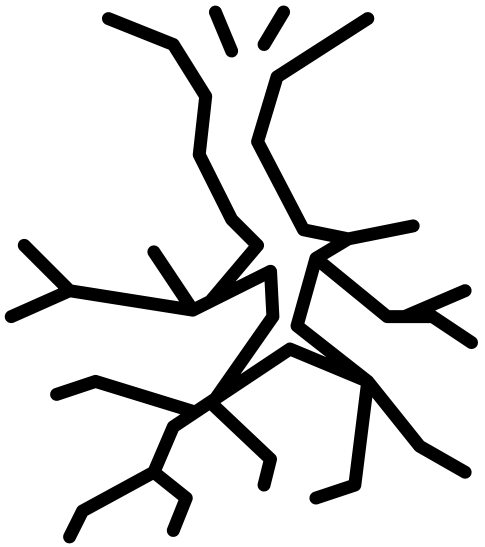
FAST CAMPUS
ONLINE
류호석 강사.

I Rooted Tree 란?

1. Node
2. Root
3. Depth
4. Parent, Child, Ancestor, Sibling
5. Leaf Node

Rooted Tree

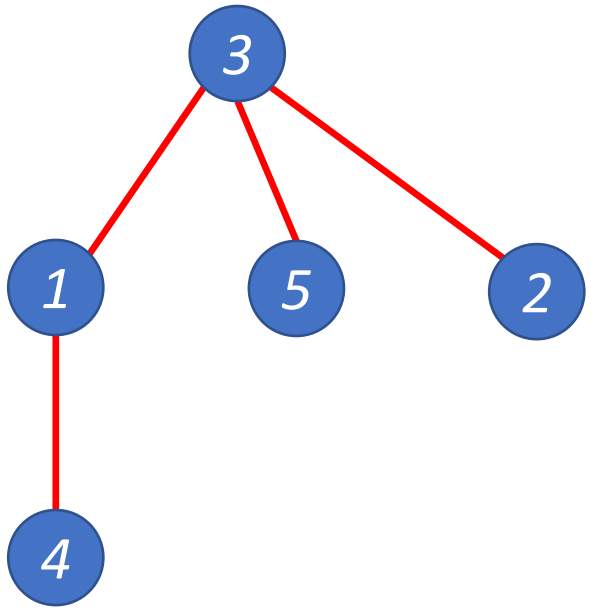




I Rooted Tree 란?

1. **Node**
2. Root
3. Depth
4. Parent, Child, Ancestor, Sibling
5. Leaf Node

Rooted Tree

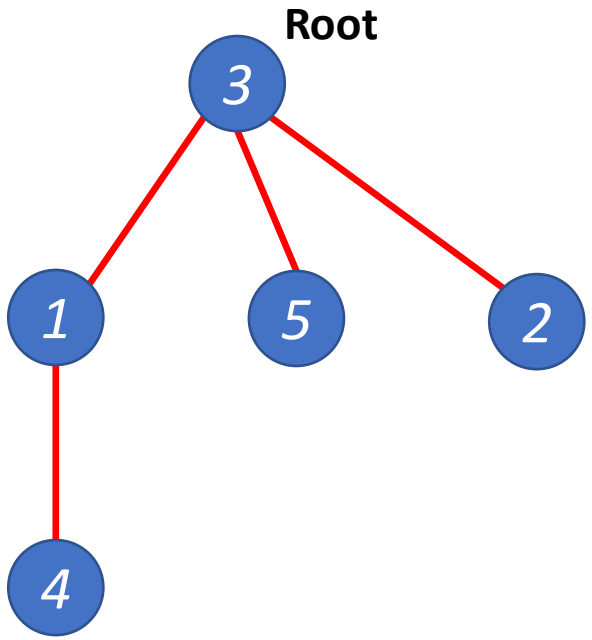


Node = Vertex

I Rooted Tree 란?

1. Node
2. **Root**
3. Depth
4. Parent, Child, Ancestor, Sibling
5. Leaf Node

Rooted Tree

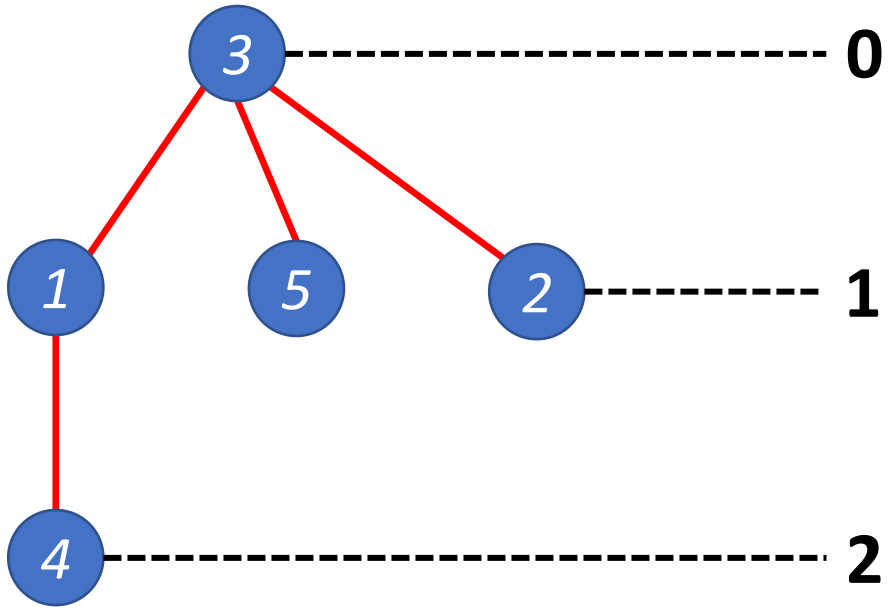


I Rooted Tree 란?

1. Node
2. Root
3. **Depth**
4. Parent, Child, Ancestor, Sibling
5. Leaf Node

Rooted Tree

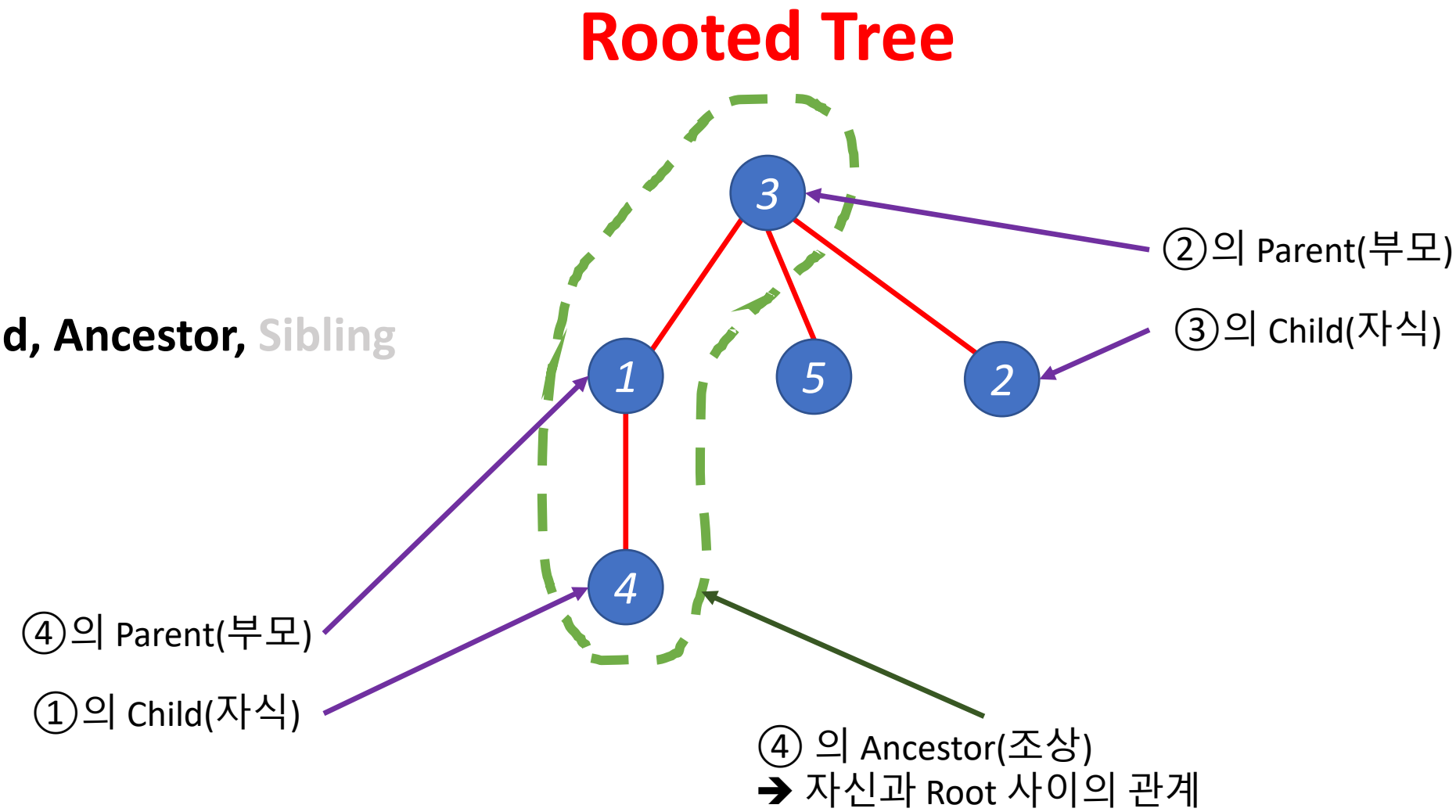
상대적인 값이므로,
1부터 시작해도 상관없다.



Depth == Root로부터의 거리

I Rooted Tree 란?

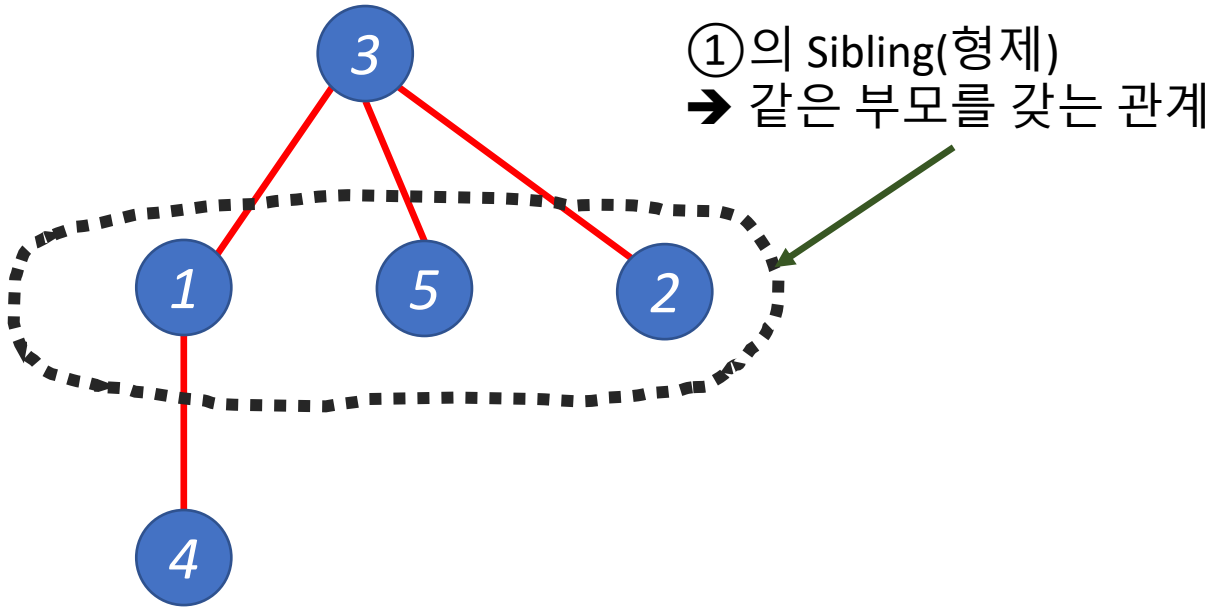
1. Node
2. Root
3. Depth
4. **Parent, Child, Ancestor,** Sibling
5. Leaf Node



I Rooted Tree 란?

1. Node
2. Root
3. Depth
4. Parent, Child, Ancestor, **Sibling**
5. Leaf Node

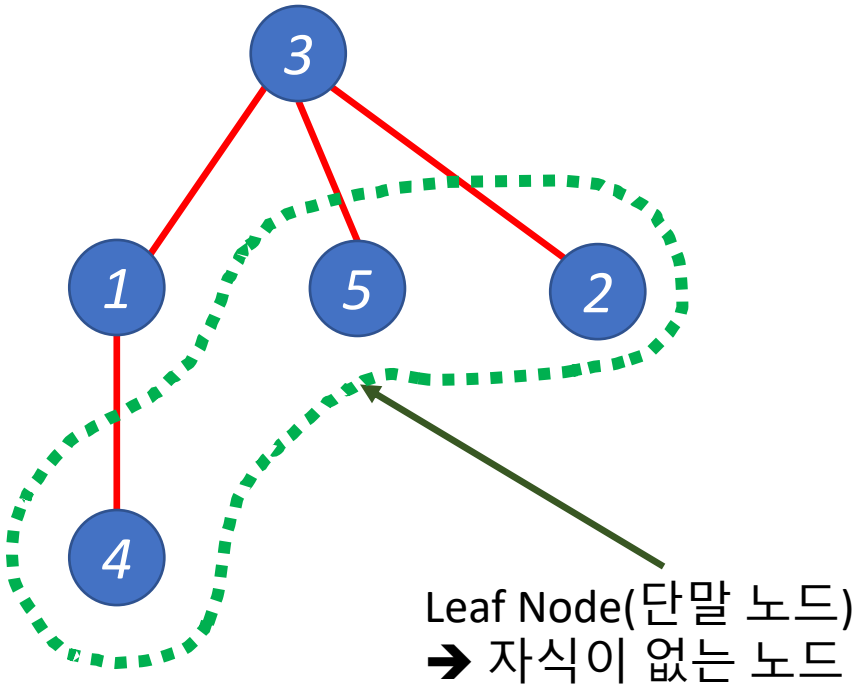
Rooted Tree



I Rooted Tree 란?

1. Node
2. Root
3. Depth
4. Parent, Child, Ancestor, Sibling
5. **Leaf Node**

Rooted Tree



I 트리(Tree) 문제의 **Keyword**

- 즉 모든 두 정점 사이에 이들을 잇는 경로가 존재하면서 사이클이 존재하지 않는 경우만 고려한다.
- 즉 마을과 마을 사이를 직접 잇는 N-1개의 길이 있으며, 모든 마을은 연결되어 있다.
- 문제는 일반적인 그래프가 아니라 트리(연결되어 있고 사이클이 없는 그래프)

I 트리(Tree)를 저장하는 방법

그래프를 저장하는 대표적인 두 가지 방법

~~1. 인접 행렬 (Adjacency Matrix)~~

2. 인접 리스트 (Adjacency List)

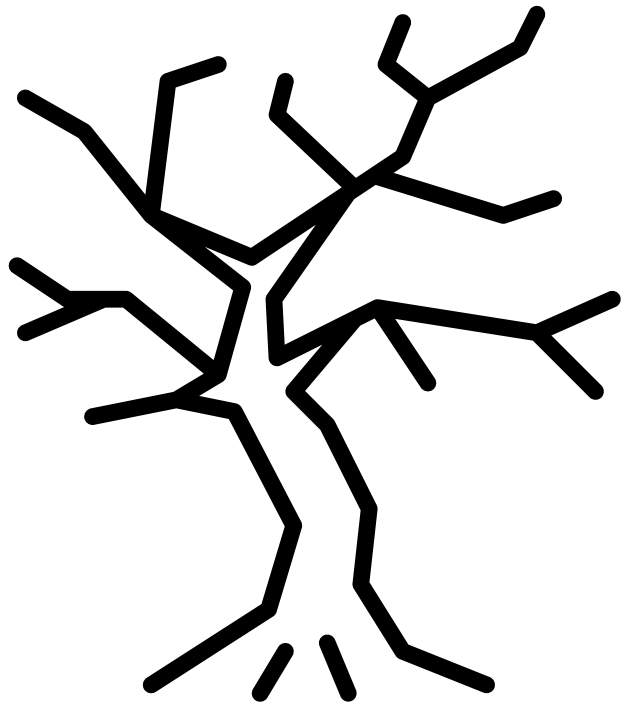
트리(Tree)는? (대부분) 인접 리스트! 이유는?

I 그래프(Graph)를 저장하는 방법 – REMIND

	인접 행렬	인접 리스트
*A*와 *B*를 잇는 간선 존재 여부 확인	$O(1)$	$O(\min(\deg(A), \deg(B))$
*A*와 연결된 모든 정점 확인	$O(\|V\|)$	$O(\deg(A))$
공간 복잡도	$O(\|V\|^2)$	$O(\|E\|)$

I 트리(Graph) 문제의 **핵심**!

- **정점(Vertex)** & **간선(Edge)** 에 대한 정확한 정의
- 트리의 요소와 문제의 요구 사항 매치!

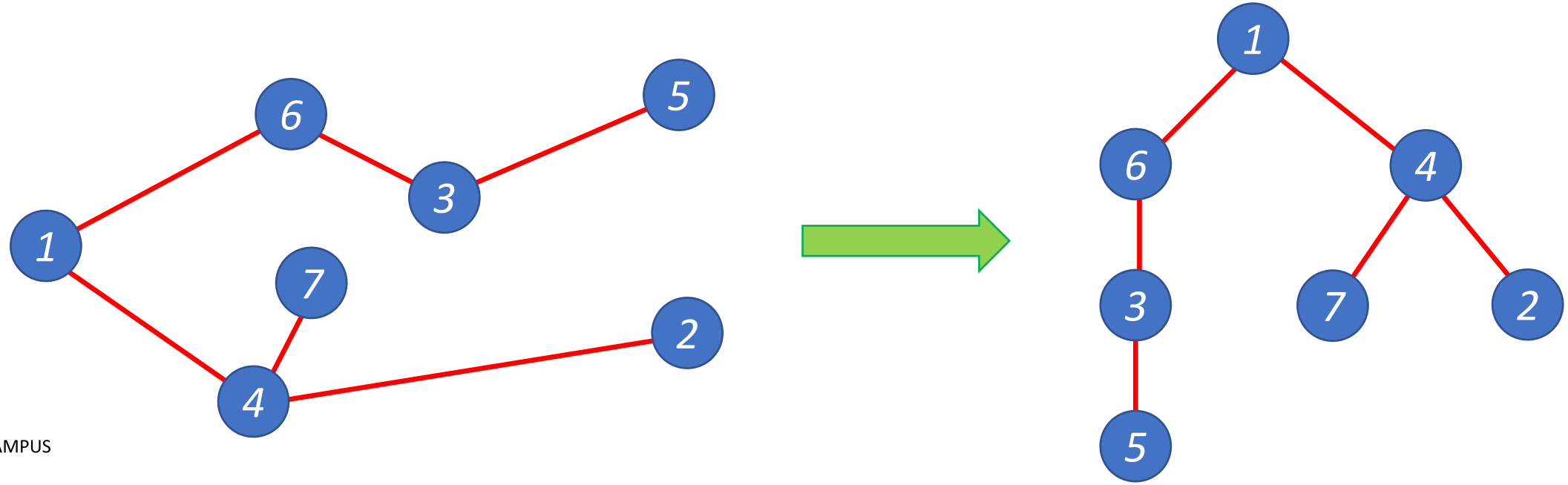


BOJ 11725 – 트리의 부모 찾기

난이도: 2

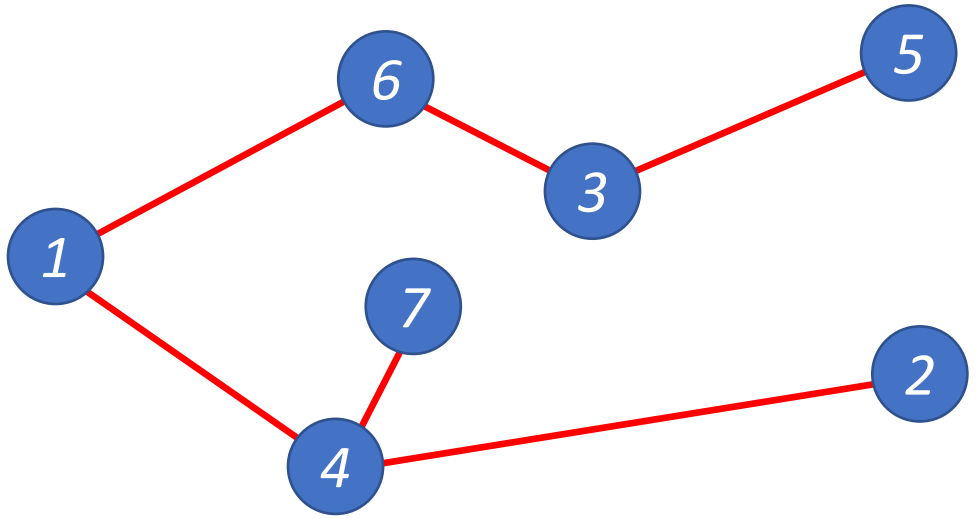
$1 \leq$ 정점 개수, $N \leq 100{,}000$

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



I 접근 – Non-Rooted Tree ➔ Rooted Tree

1. 인접 리스트로 저장하기



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

I 접근 – Non-Rooted Tree ➔ Rooted Tree

1. 인접 리스트로 저장하기
2. ROOT 말고는 아무것도 정답을 구하지 못한 상태로 시작.

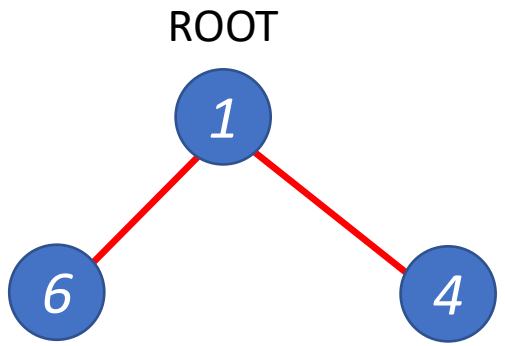
I 접근 – Non-Rooted Tree ➔ Rooted Tree

1. 인접 리스트로 저장하기
2. ROOT 말고는 아무것도 정답을 구하지 못한 상태로 시작.
3. 정점 X가 Parent를 안다면, 자신의 자식 Children 을 찾을 수 있다.

1과 연결된 정점들	6	4

I 접근 – Non-Rooted Tree ➔ Rooted Tree

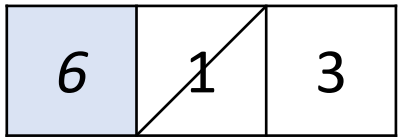
1. 인접 리스트로 저장하기
2. ROOT 말고는 아무것도 정답을 구하지 못한 상태로 시작.
3. 정점 X가 Parent를 안다면, 자신의 자식 Children 을 찾을 수 있다.
    1. 어떻게? 연결된 것들 중 Parent를 제외한 모든 것들!

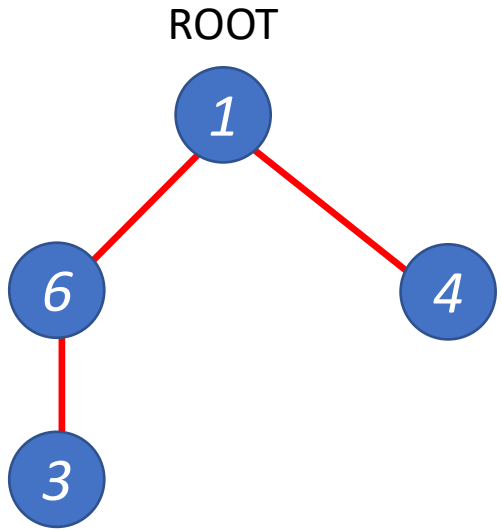


1과 연결된 정점들	6	4

I 접근 – Non-Rooted Tree ➔ Rooted Tree

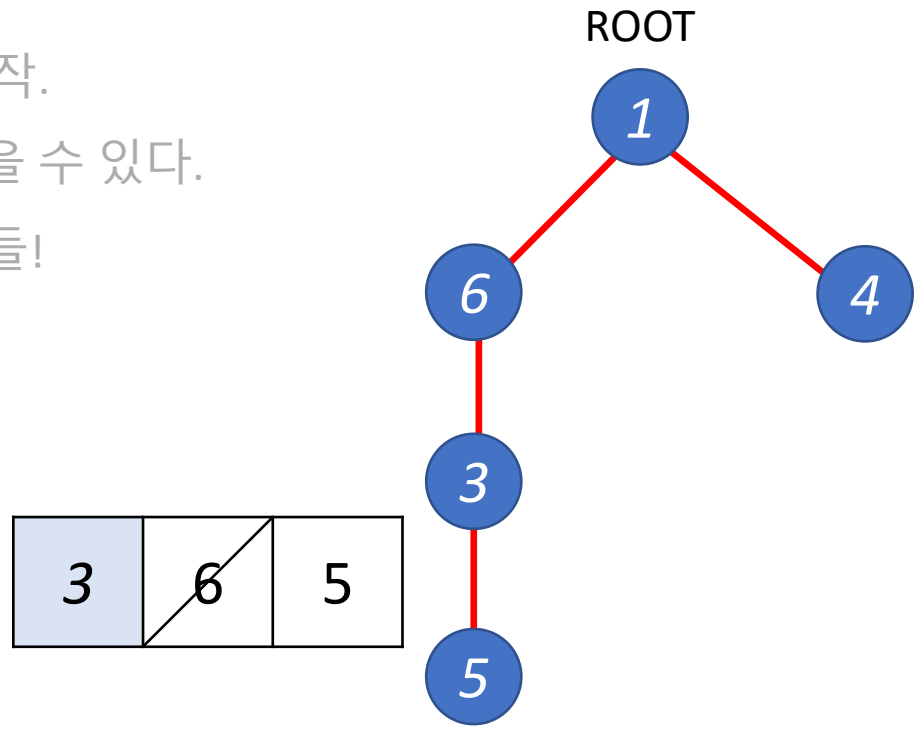
1. 인접 리스트로 저장하기
2. ROOT 말고는 아무것도 정답을 구하지 못한 상태로 시작.
3. 정점 X가 Parent를 안다면, 자신의 자식 Children 을 찾을 수 있다.
    1. 어떻게? 연결된 것들 중 Parent를 제외한 모든 것들!
4. ROOT부터 차례대로 문제를 해결해보자!





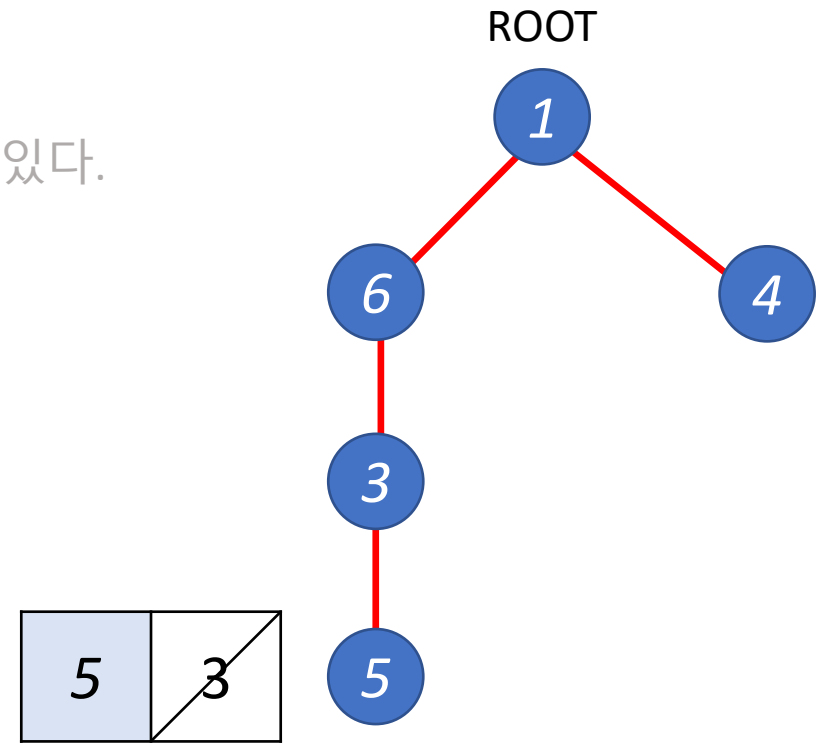
I 접근 – Non-Rooted Tree ➔ Rooted Tree

1. 인접 리스트로 저장하기
2. ROOT 말고는 아무것도 정답을 구하지 못한 상태로 시작.
3. 정점 X가 Parent를 안다면, 자신의 자식 Children 을 찾을 수 있다.
    1. 어떻게? 연결된 것들 중 Parent를 제외한 모든 것들!
4. ROOT부터 차례대로 문제를 해결해보자!



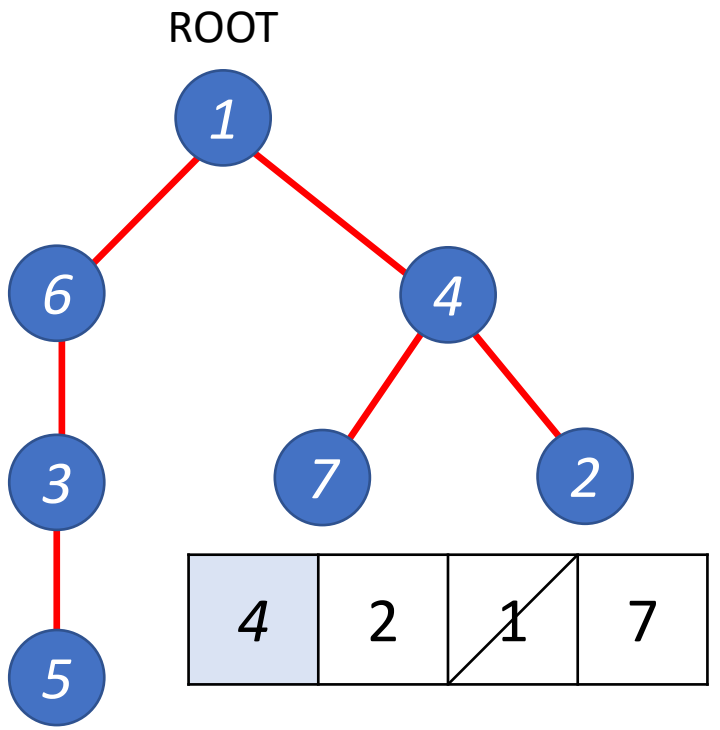
I 접근 – Non-Rooted Tree ➔ Rooted Tree

1. 인접 리스트로 저장하기
2. ROOT 말고는 아무것도 정답을 구하지 못한 상태로 시작.
3. 정점 X가 Parent를 안다면, 자신의 자식 Children 을 찾을 수 있다.
    1. 어떻게? 연결된 것들 중 Parent를 제외한 모든 것들!
4. ROOT부터 차례대로 문제를 해결해보자!



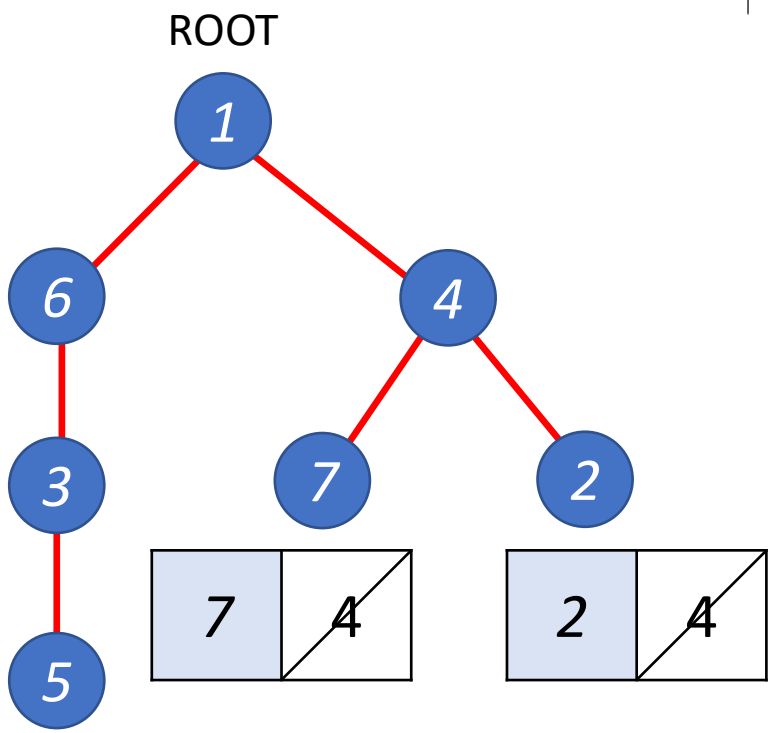
I 접근 – Non-Rooted Tree ➔ Rooted Tree

1. 인접 리스트로 저장하기
2. ROOT 말고는 아무것도 정답을 구하지 못한 상태로 시작.
3. 정점 X가 Parent를 안다면, 자신의 자식 Children 을 찾을 수 있다.
    1. 어떻게? 연결된 것들 중 Parent를 제외한 모든 것들!
4. ROOT부터 차례대로 문제를 해결해보자!



I 접근 – Non-Rooted Tree ➔ Rooted Tree

1. 인접 리스트로 저장하기
2. ROOT 말고는 아무것도 정답을 구하지 못한 상태로 시작.
3. 정점 X가 Parent를 안다면, 자신의 자식 Children 을 찾을 수 있다.
    1. 어떻게? 연결된 것들 중 Parent를 제외한 모든 것들!
4. ROOT부터 차례대로 문제를 해결해보자!



I 시간, 공간 복잡도 계산하기

"Root"를 시작으로 하는 그래프 탐색 문제

탐색 알고리즘: BFS or DFS
➔ 인접 리스트를 쓴다면 O(V + E)

➔ DFS 가 매우 쉽게 구현할 수 있다!

I 구현

```java
static void input() {
    n = scan.nextInt();

    // 인접 리스트 구성하기
    /* TODO */
}

// dfs(x, par) := 정점 x 의 부모가 par 였고, x의 children들을 찾아주는 함수
static void dfs(int x, int par) {
    /* TODO */
}

static void pro() {
    // 1 번 정점이 ROOT 이므로, 여기서 시작해서 Tree의 구조를 파악하자.
    /* TODO */

    // 정답 출력하기
    /* TODO */
}

public static void main(String[] args) {
    input();
    pro();
}
```

I 연습 문제

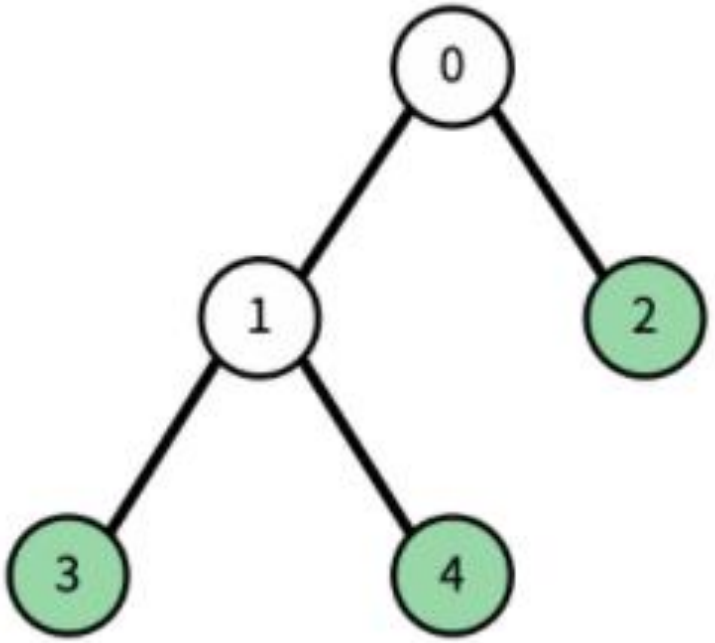
- BOJ 1991 – 트리 순회
- BOJ 5639 – 이진 검색 트리
- BOJ 15900 – 나무 탈출
- BOJ 20364 – 부동산 다툼
- BOJ 3584 – 가장 가까운 공통 조상
- BOJ 1240 – 노드 사이의 거리
- BOJ 9489 – 사촌

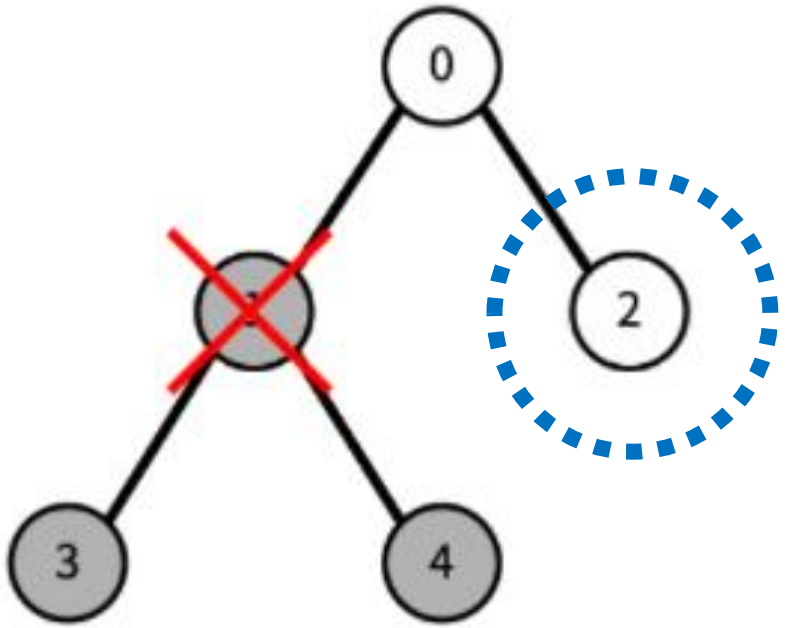
이외의 추천 문제가 추가되면 Github 자료에 코드 업로드

BOJ 1068 – 트리

난이도: 2

$1 \leq$ 정점 개수, $N \leq 50$





정답: 1

FAST CAMPUS
ONLINE
류호석 강사.

I 접근 – 정답의 최대치

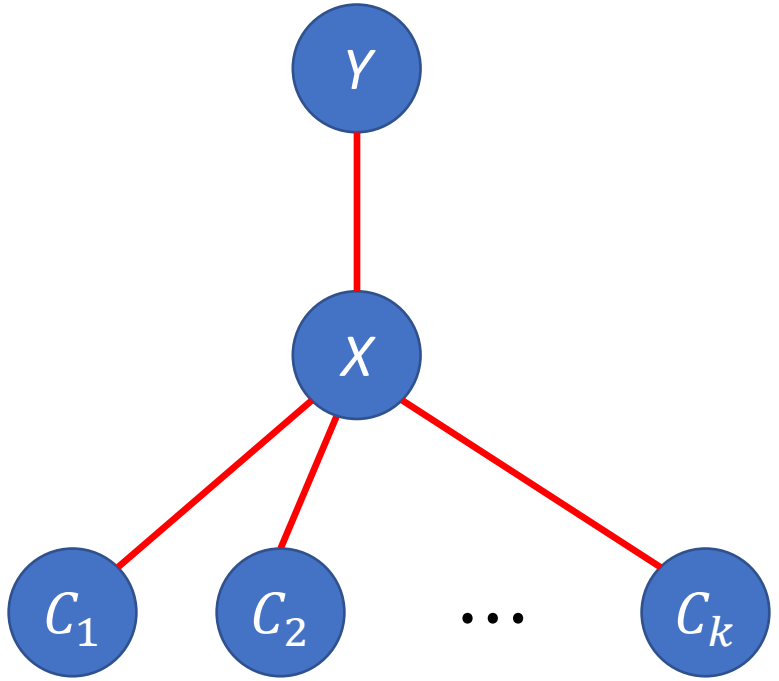
단말 노드의 개수 $\leq$ 전체 정점의 개수 $\leq N$

Integer 범위 안에 들어온다.

I 접근 – 1. 정점 제거 방법

1. 정점 x 가 지워진다.

➔ 그래프에서 정점이 사라진다?



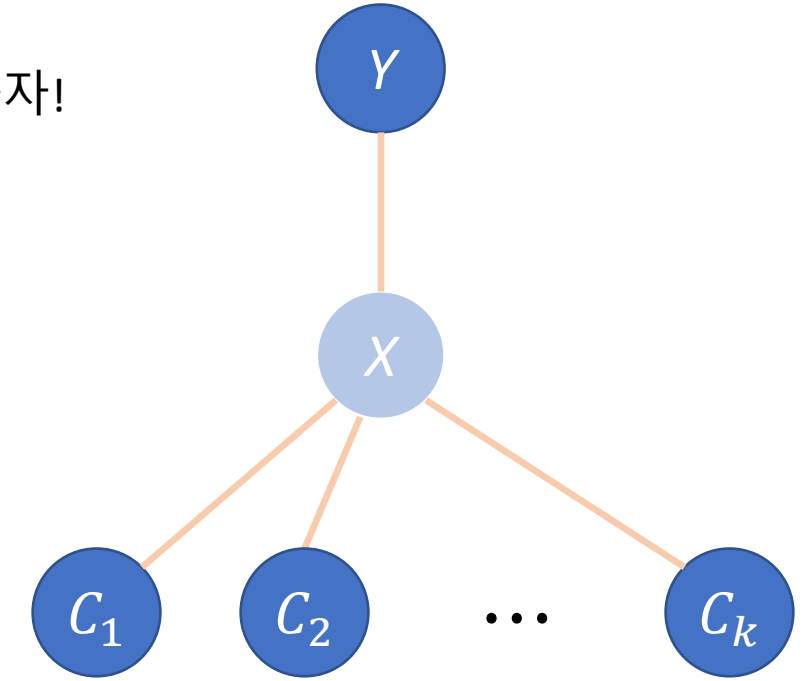
I 접근 – 1. 정점 제거 방법

1. 정점 x 가 지워진다.

➔ 그래프에서 정점이 사라진다?

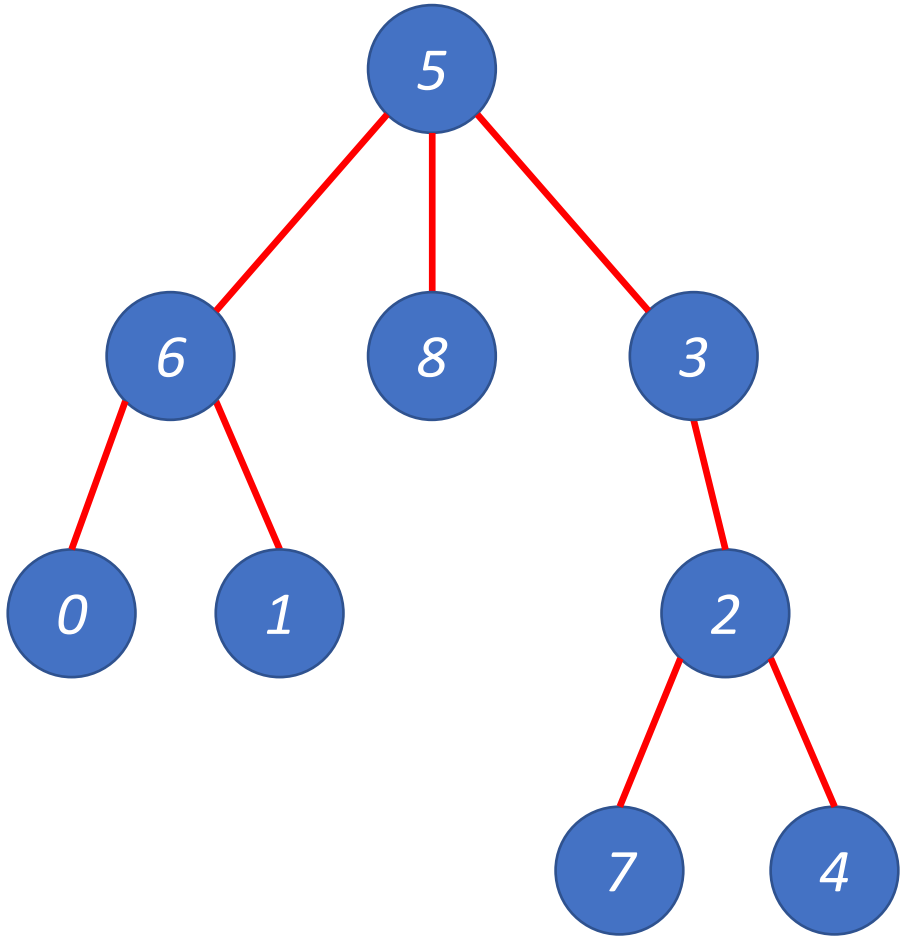
➔ 정점과 이어진 간선들이 모두 사라진다.

➔ 정점 x 의 부모에서 x 로 가는 간선을 ***삭제 or 무시*** 하자!



I 접근 – 2. 트리의 단말 노드 개수를 세는 방법

2. 트리의 단말 노드의 개수?

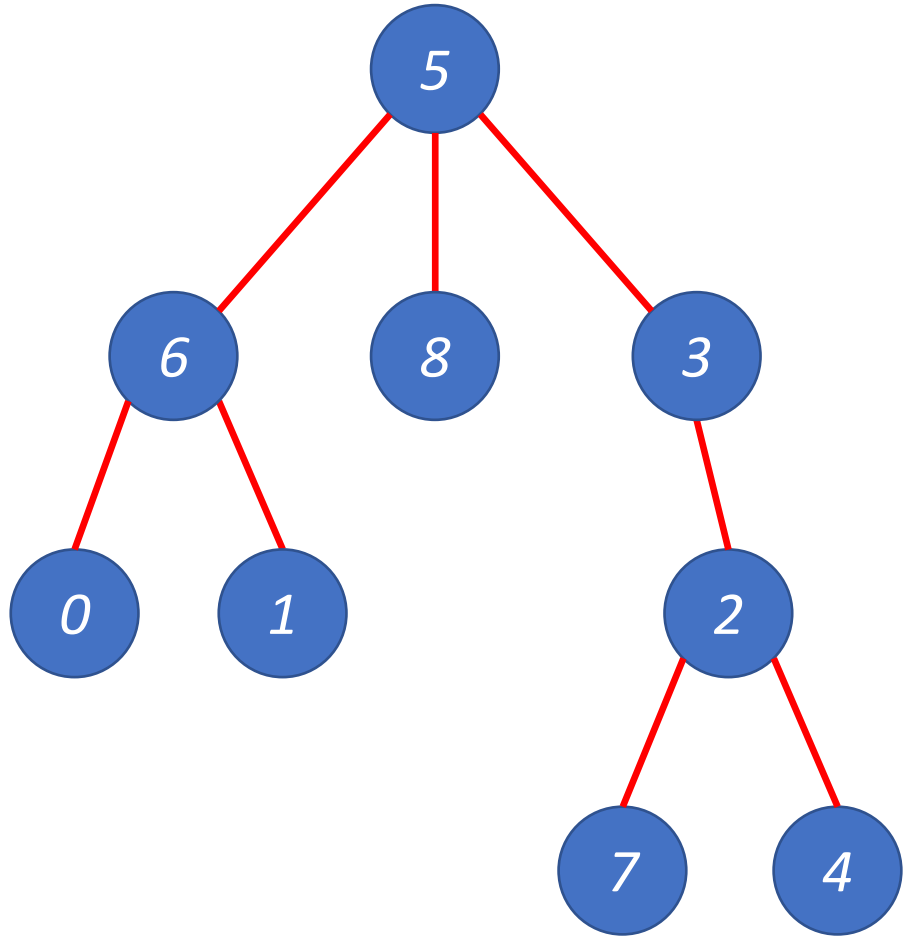


FAST CAMPUS
ONLINE
류호석 강사.

I 접근 – 2. 트리의 단말 노드 개수를 세는 방법

2. 트리의 단말 노드의 개수?

➔ 트리 := Root 노드와 연결된 정점들의 그래프



I 접근 – 2. 트리의 단말 노드 개수를 세는 방법

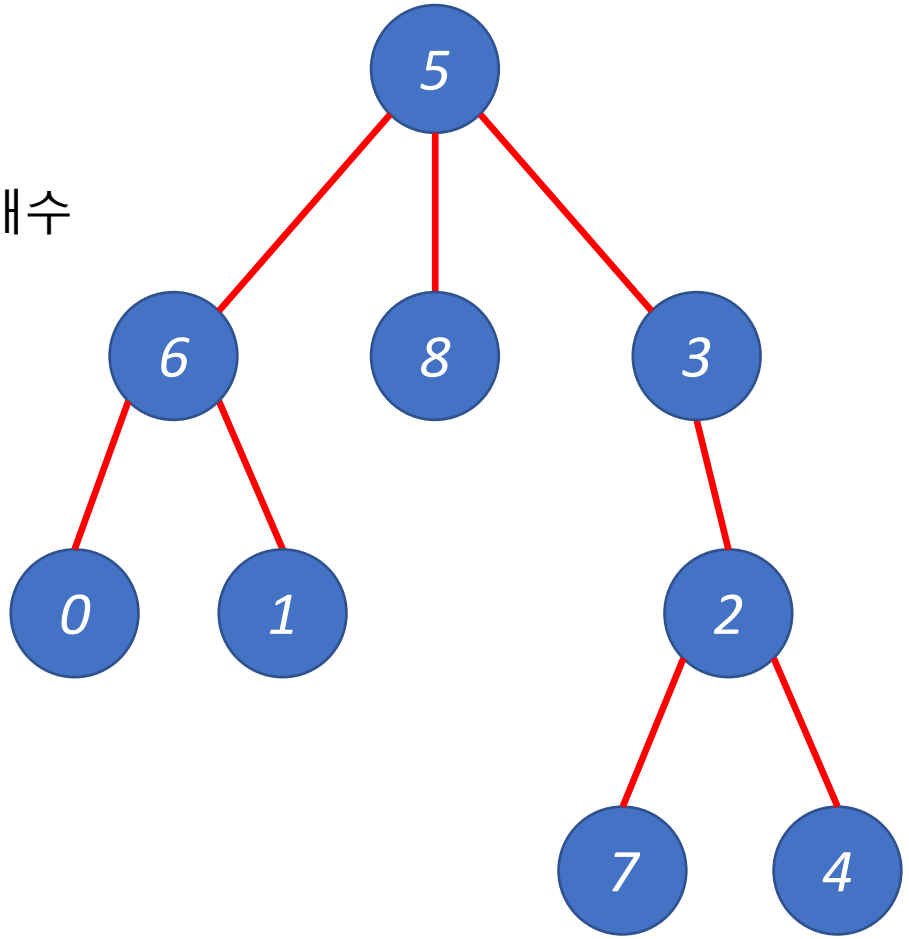
2. 트리의 단말 노드의 개수?

➔ 트리 := Root 노드와 연결된 정점들의 그래프

➔ 트리의 단말 노드 := Root 노드에서 탐색할 수 있는 단말 노드의 개수

➔ Root를 시작점으로 하는 그래프 탐색 알고리즘! BFS or DFS

➔ 이렇게 문제를 풀어도 된다.

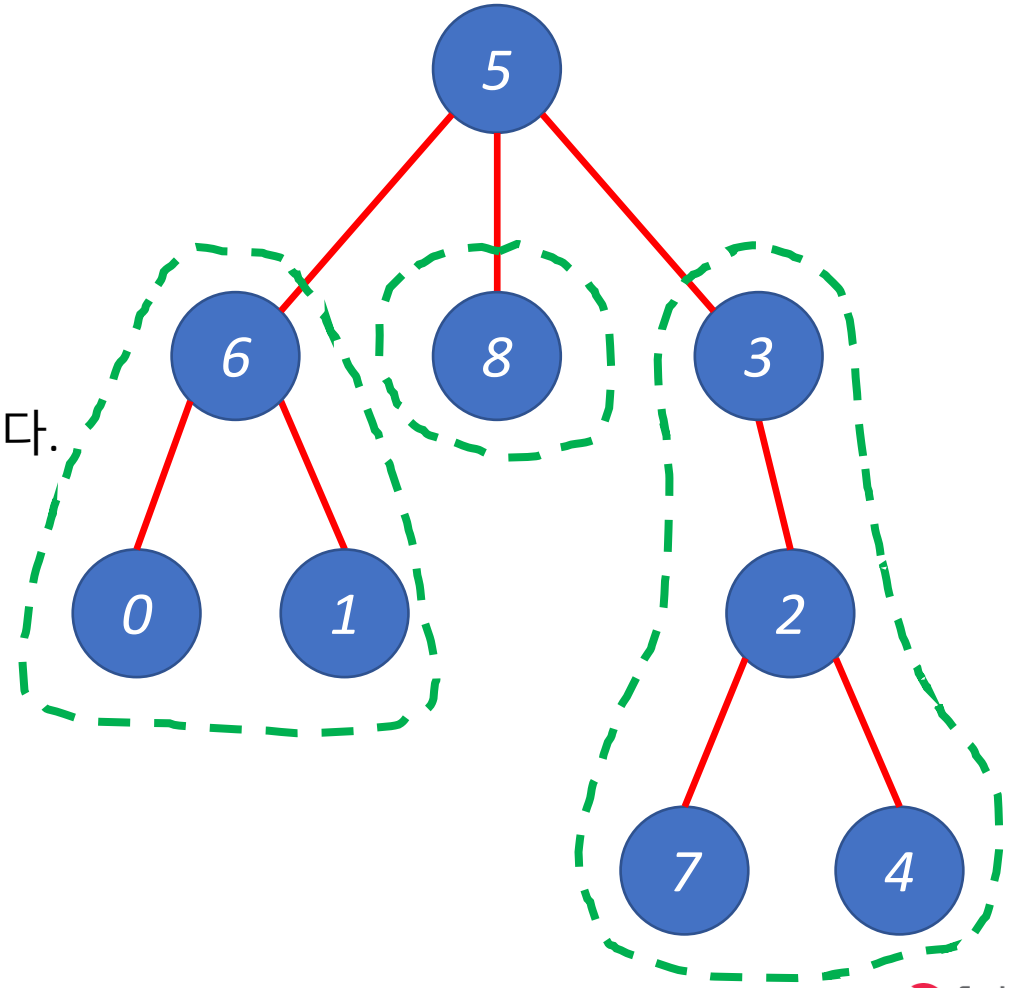


I 접근 – 트리에서의 큰 문제와 작은 문제

새로운 용어: Subtree

Rooted tree에서 어떤 정점 X 의 **Subtree**란?

X와 그의 모든 자손들을 포함하는 트리! ➔ X가 새로운 Root가 된다.



FAST CAMPUS
ONLINE
류호석 강사.

I 접근 – 트리에서의 큰 문제와 작은 문제

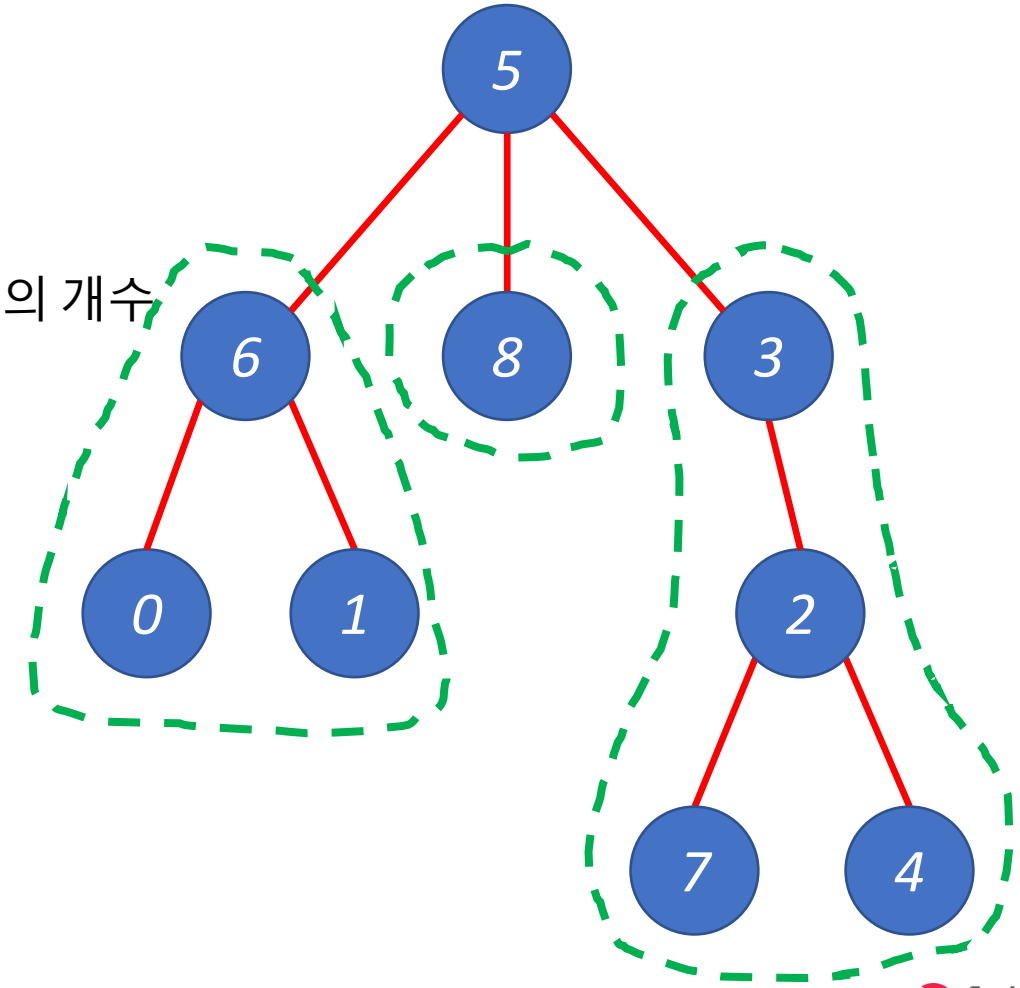
큰 문제와 **작은 문제**

큰 문제 := 트리 안에 있는 단말 노드의 개수

작은 문제 := Root의 자식 노드들의 subtree 안에 있는 단말 노드의 개수

<포인트>

큰 문제의 정답을 **작은 문제**의 정답을 이용해서 구하자!

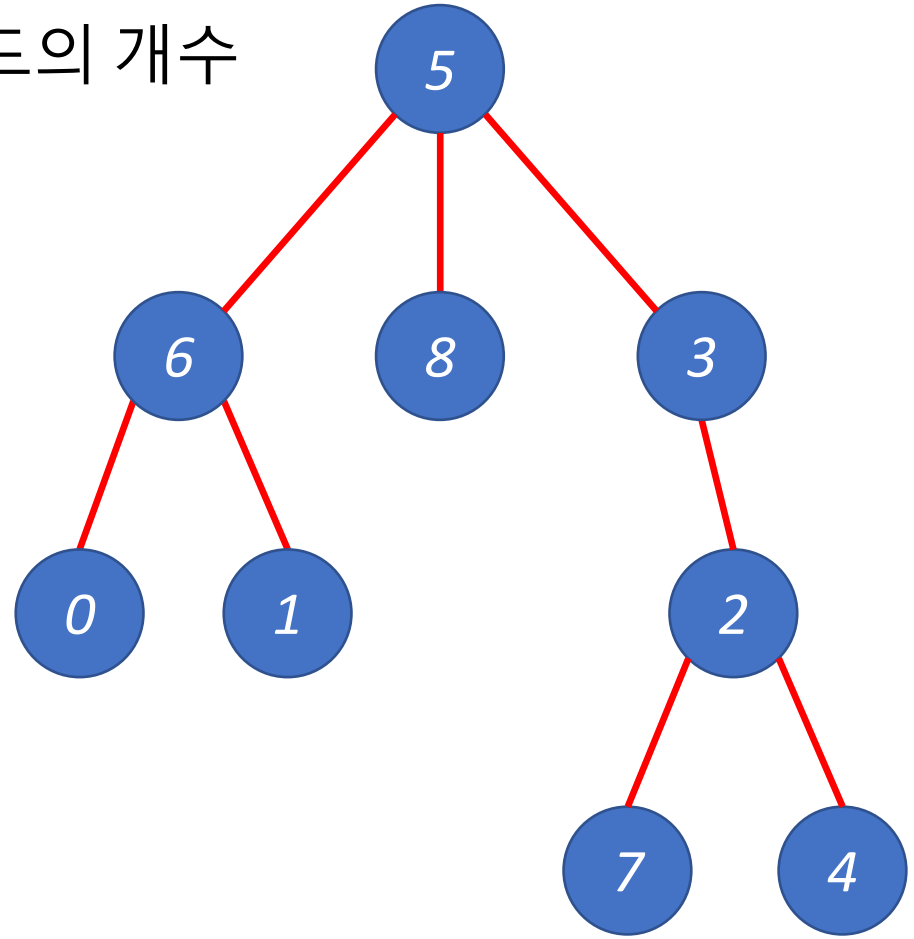


I 접근 – 3. 단말 노드의 개수를 세는 법

$leaf[x]$:= x를 root로 하는 subtree에 있는 단말 노드의 개수

- x 가 단말 노드인 경우 ➔ $leaf[x] = 1$
- 아닌 경우 ➔ x의 자식들에 대해 $leaf$ 를 먼저 계산한다면?

  ➔ $leaf[x] = \sum leaf[x$의 자식 노드들$]$



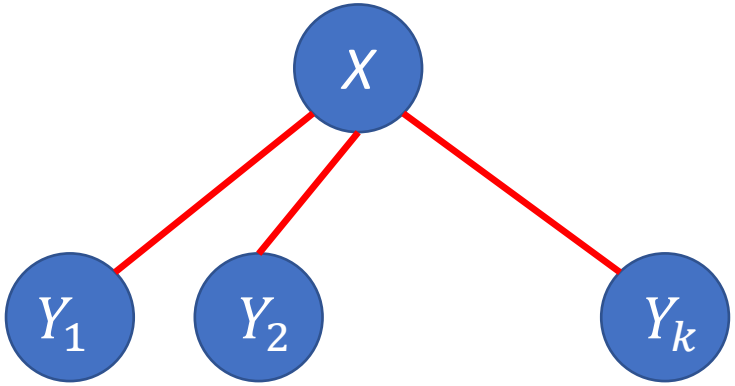
I 접근 – 3. 단말 노드의 개수를 세는 법

< $leaf[x]$ 를 계산하는 방법>

Root 에서 DFS를 한다면?

어떤 노드 X에서 자식 노드 Y에 대한 탐색을 끝내고 돌아오면

leaf[Y] 값이 계산되어 왔을 테니, leaf[X] 에 leaf[Y] 를 누적해주면 된다.



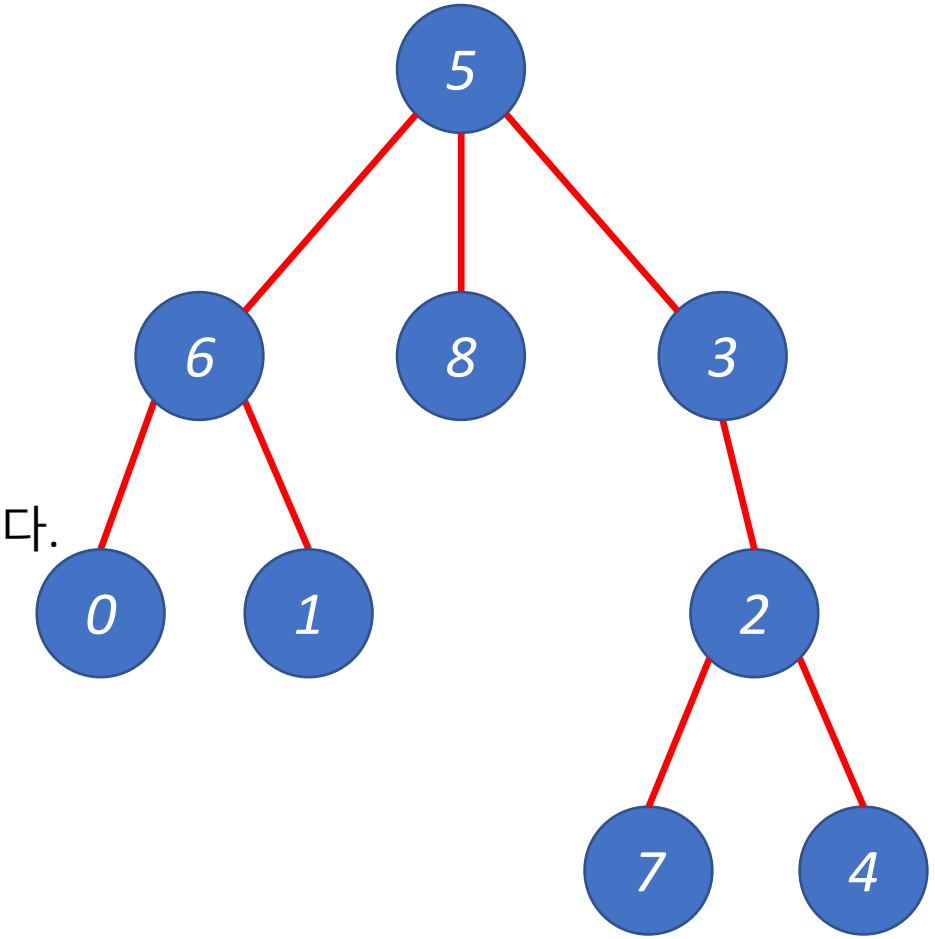
I 접근 – 3. 단말 노드의 개수를 세는 법

< $leaf[x]$ 를 계산하는 방법>

Root 에서 DFS를 한다면?

어떤 노드 X에서 자식 노드 Y에 대한 탐색을 끝내고 돌아오면

leaf[Y] 값이 계산되어 왔을 테니, leaf[X] 에 leaf[Y] 를 누적해주면 된다.



FAST CAMPUS
ONLINE
류호석 강사.

I 시간, 공간 복잡도 계산하기

"Root"를 시작으로 하는 그래프 탐색 문제

탐색 알고리즘: BFS or DFS
➔ 인접 리스트를 쓴다면 O(V + E)

➔ DFS 가 매우 쉽게 구현할 수 있다!

I 구현

```java
static void input() {
    n = scan.nextInt();
    /* TODO */
}

// dfs(x, par) := 정점 x 의 부모가 par 였고, Subtree(x) 의 leaf 개수를 세주는 함수
static void dfs(int x, int par) {
    /* TODO */
}

static void pro() {
    // erased와 그의 부모 사이의 연결을 끊어주기
    /* TODO */

    // erased 가 root 인 예외 처리하기
    /* TODO */

    // 정답 출력하기
    /* TODO */
}
```

I 연습 문제

- BOJ 15681 – 트리와 쿼리
- BOJ 14267 – 회사 문화 1

이외의 추천 문제가 추가되면 Github 자료에 코드 업로드