

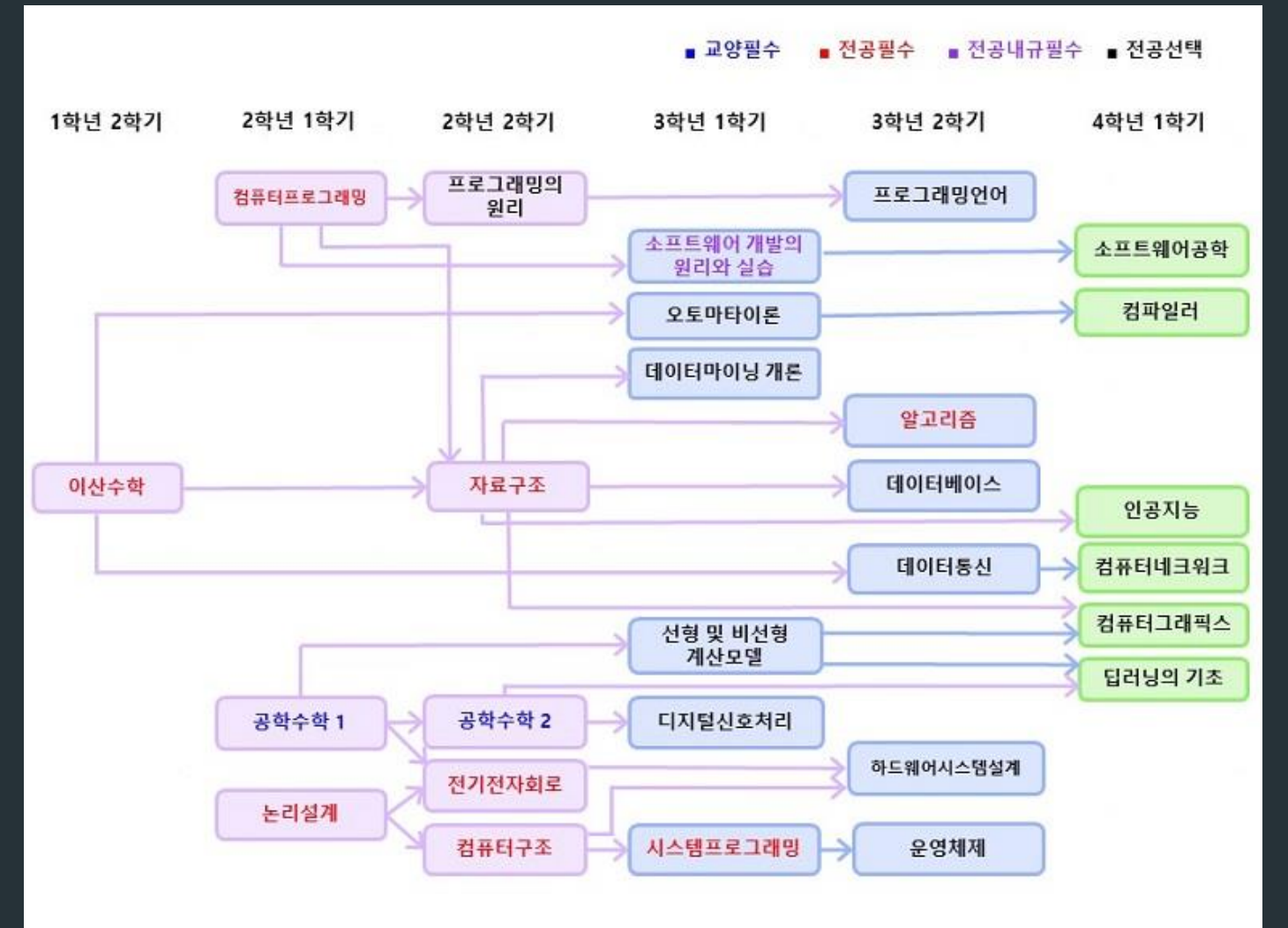
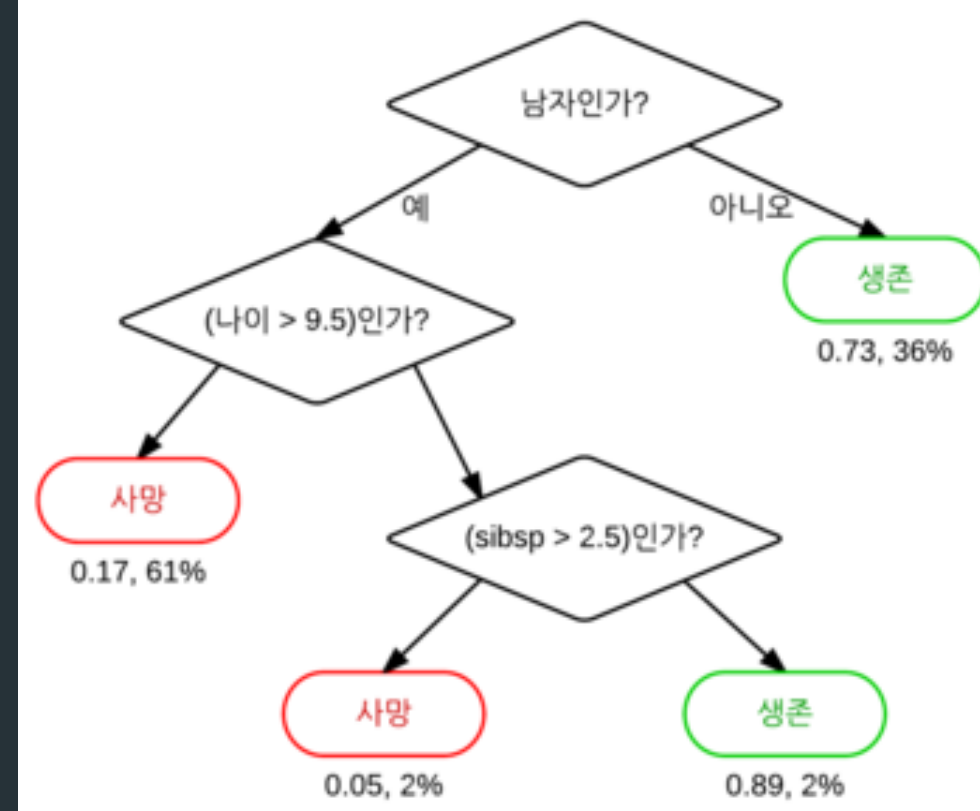
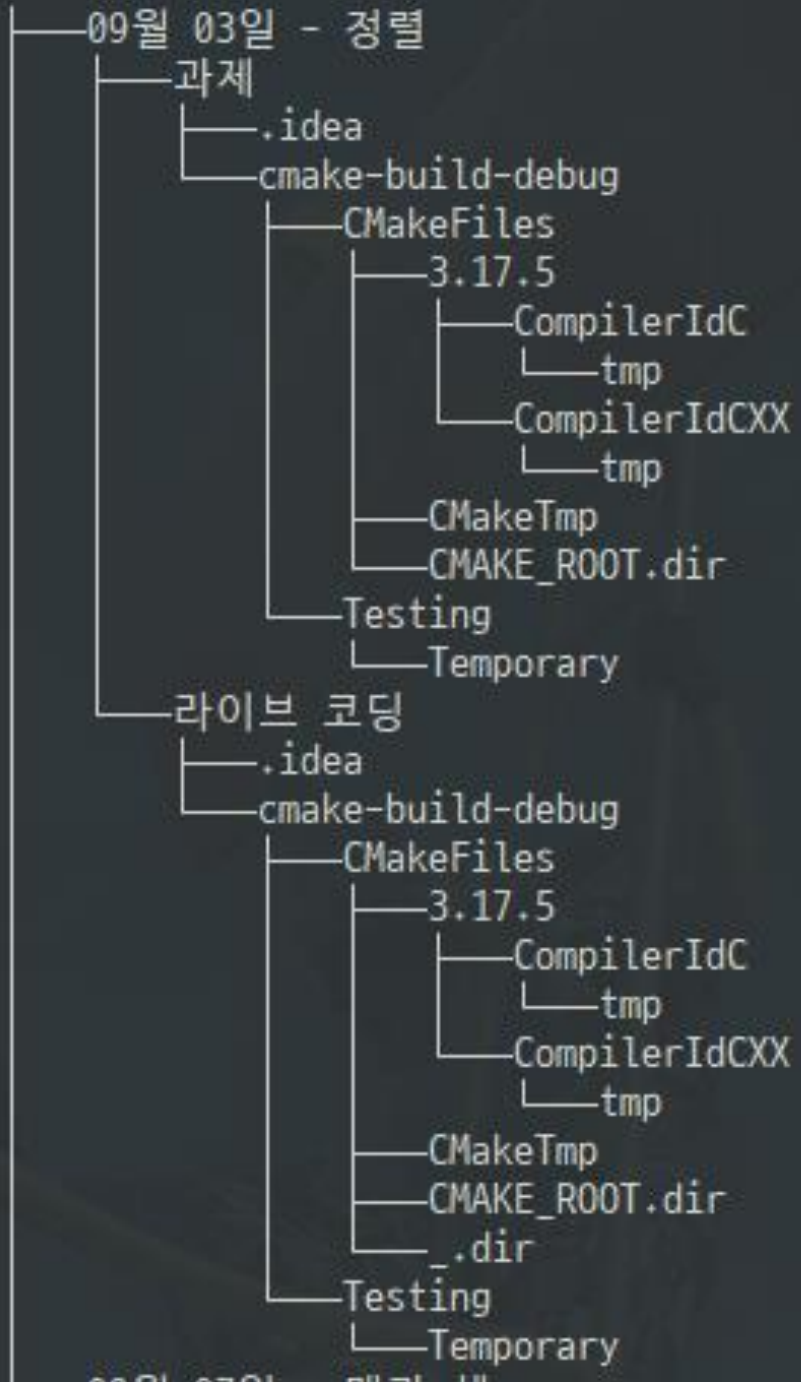
# 알튜비튜 트리

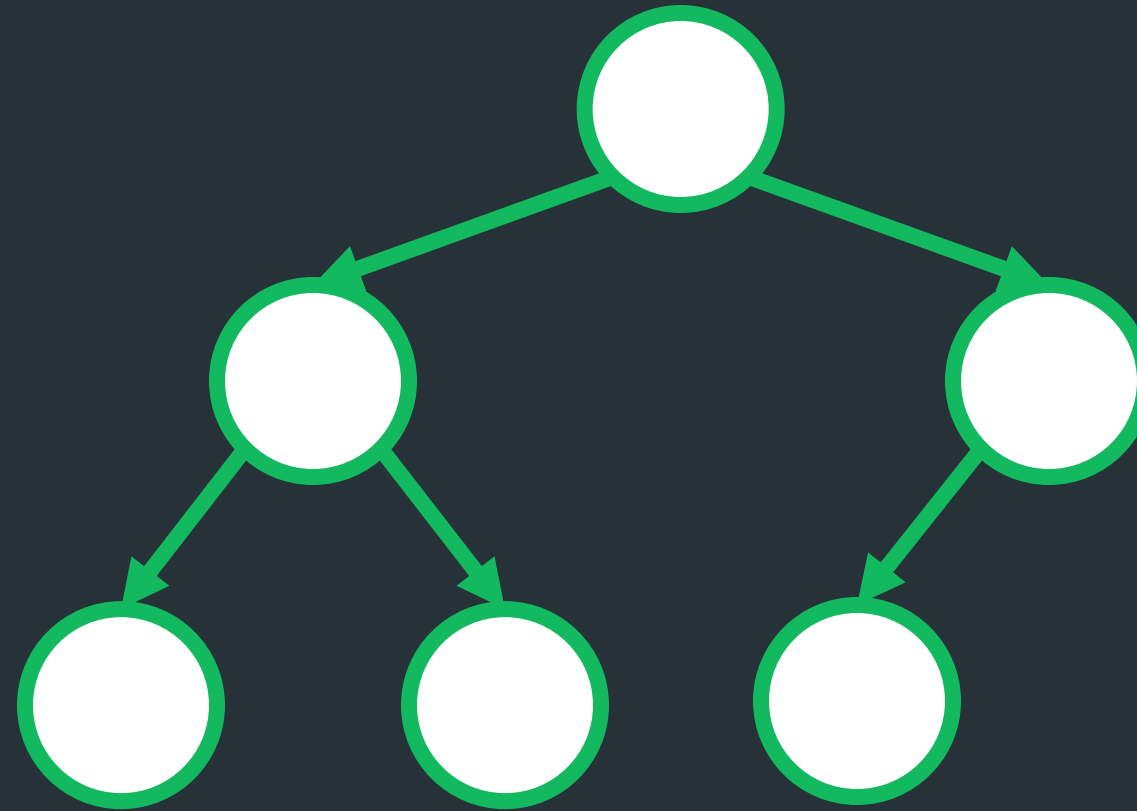


비선형 자료구조 중 하나인 트리입니다.  
그래프의 부분집합으로 계층 관계를 나타낼 때 주로 사용해요.

# 일상 속 트리

```
C:\Users\iw040\Notice (main -> origin)
λ tree
Folder PATH listing
Volume serial number is C0000100 EE98:4106
C:.
```



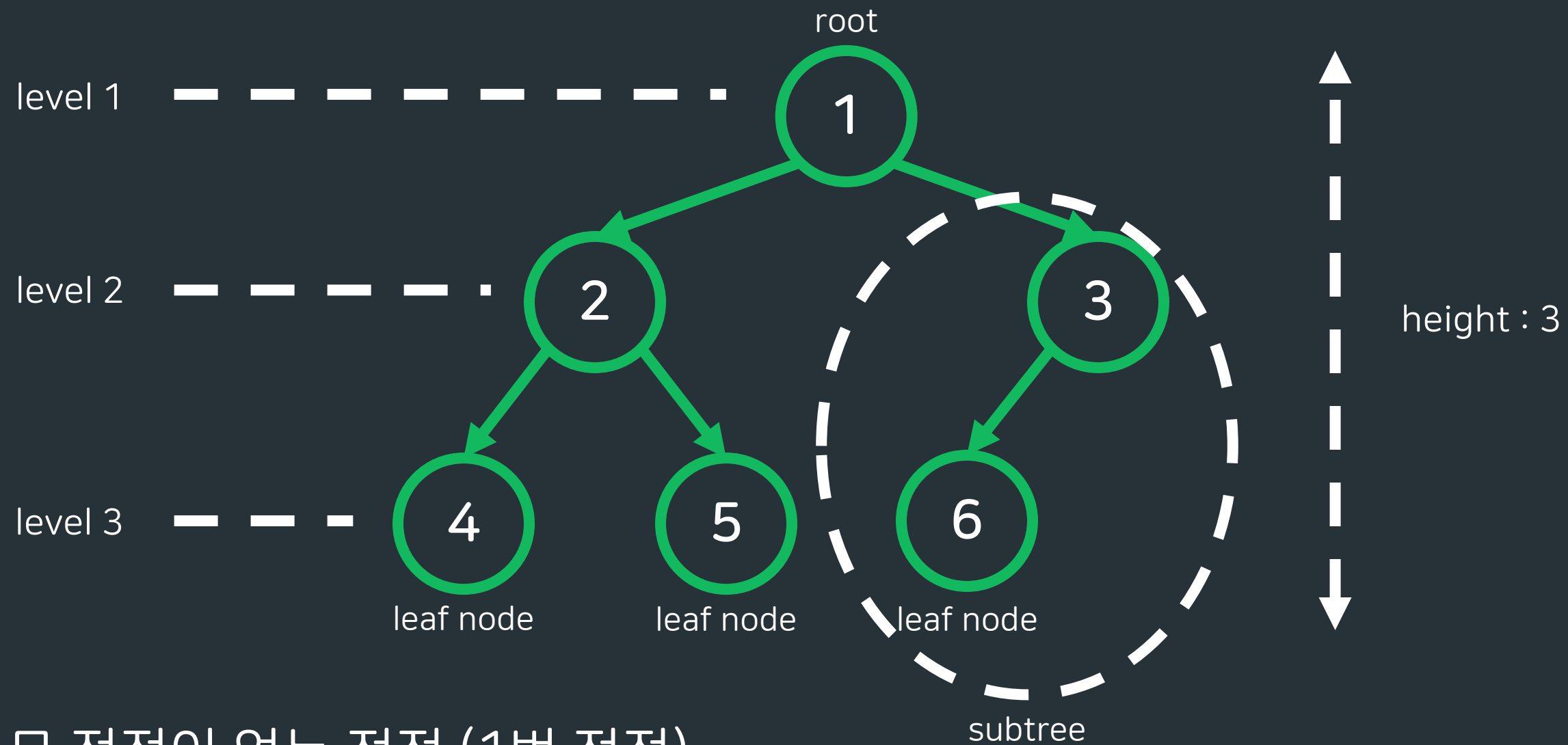


## Tree

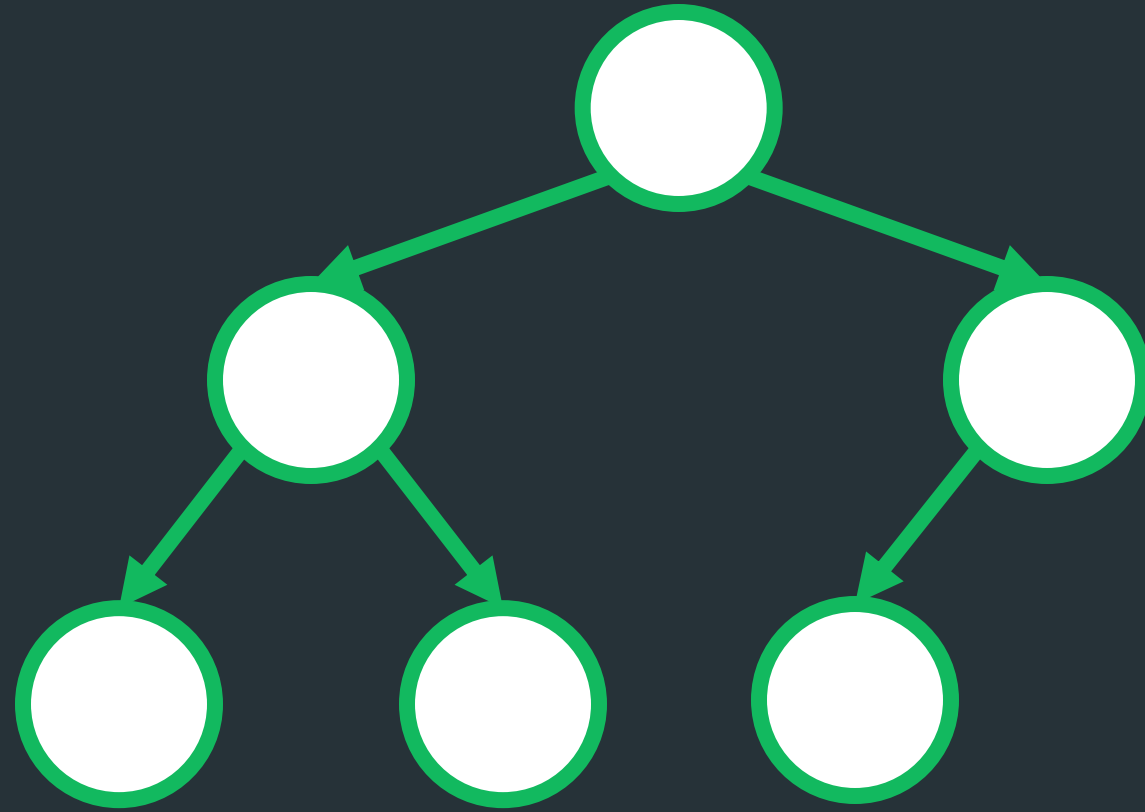
- 비선형 자료구조
- 그래프의 부분집합으로 사이클이 없고,  $V$ 개의 정점에 대해  $V-1$ 개의 간선이 있음
- 부모-자식의 계층 구조
- 트리 탐색의 시간 복잡도는  $O(h)$  ( $h$  = 트리의 높이)
- 그래프와 마찬가지로 DFS, BFS를 이용하여 탐색

	트리	그래프
간선의 수	$V-1$ 개	$V-1$ 개 이상
특정 정점 사이 경로의 수	1개	1개 이상
방향 유무	0	$\Delta$
사이클 유무	X	$\Delta$
계층 관계	0	X

## Tree

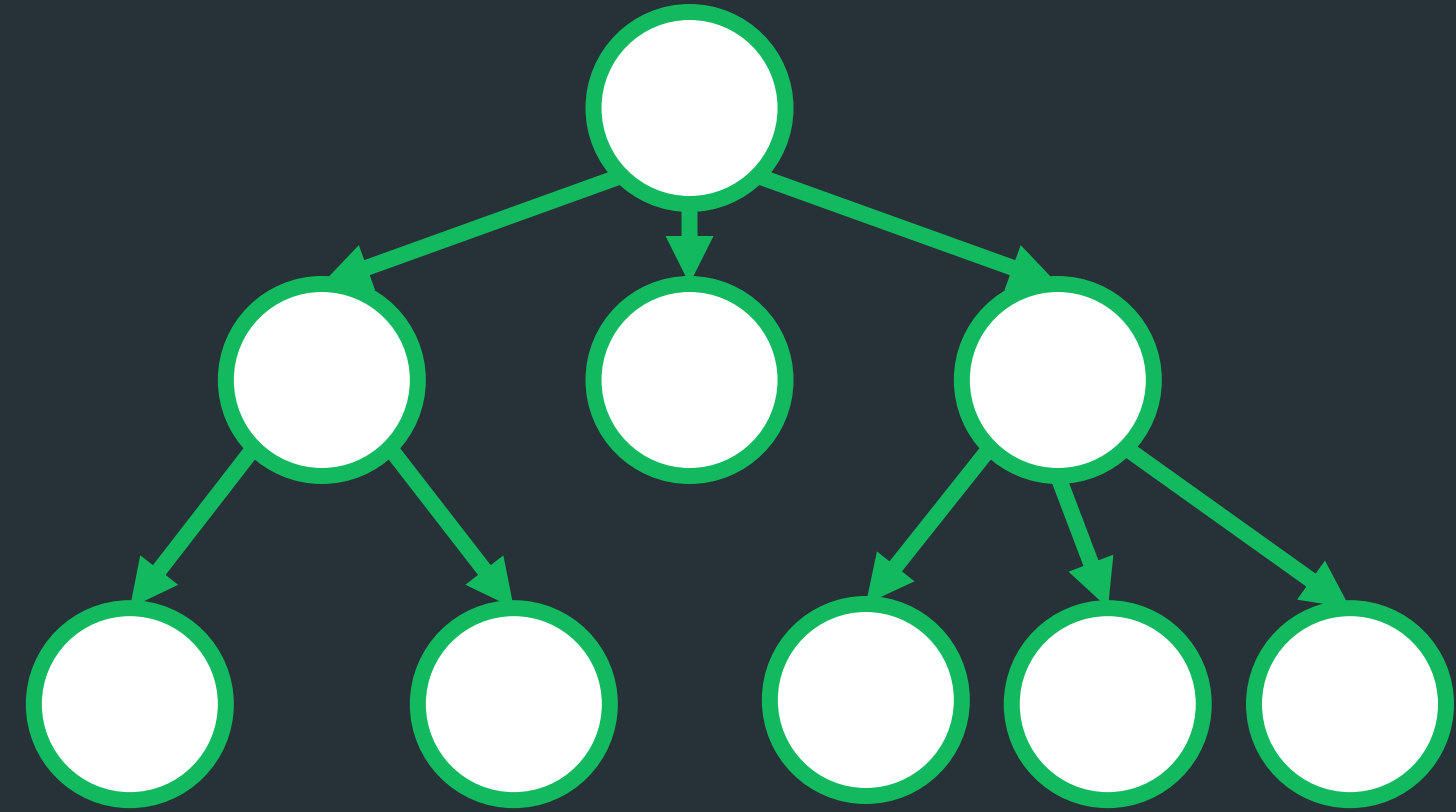


- Root : 부모 정점이 없는 정점 (1번 정점)
- Subtree : 트리의 부분 집합
- Leaf node : 자식 정점이 없는 정점 (4, 5, 6번 정점)
- Level : 트리의 각 계층
- Height : level 중 가장 큰 값



Binary Tree (이진 트리)

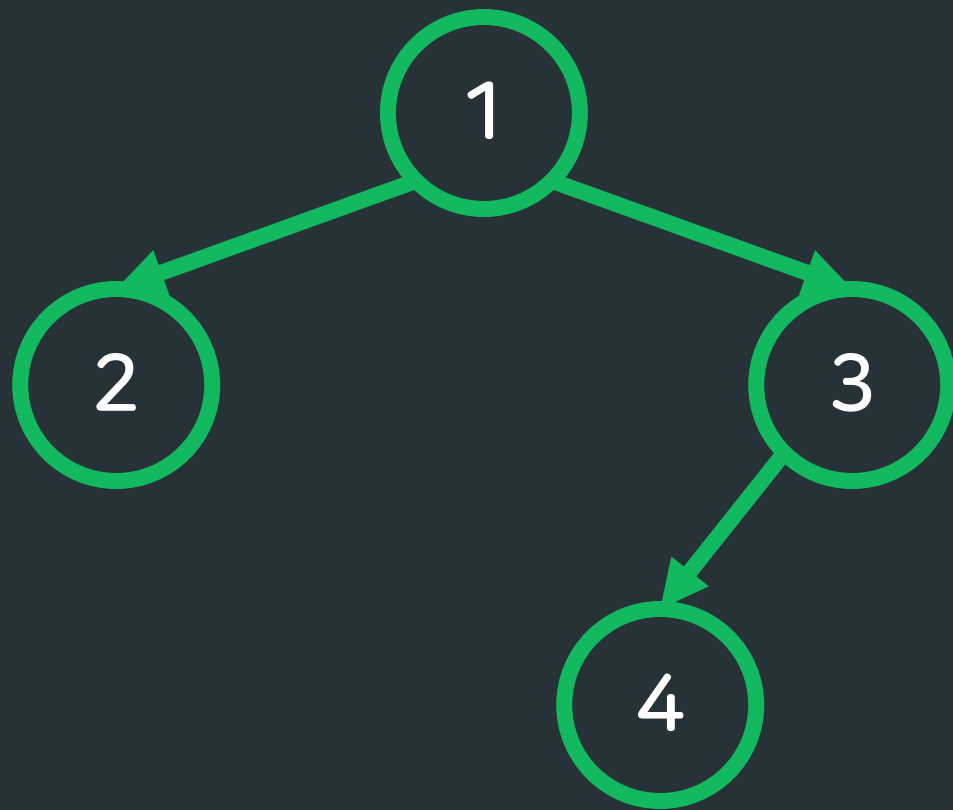
자식 정점의 수가 2개 이하



General Tree (일반 트리)

자식 정점의 수에 제한 없음

# 트리 구현 (이진 트리)



구조체 + 포인터

실시간으로 트리를 만들어야 할 때 적합



```
struct Node {  
    int data;  
    Node *left;  
    Node *right;  
};
```

```
int main() {  
    Node *n1 = new Node();  
    Node *n2 = new Node();  
    Node *n3 = new Node();  
    Node *n4 = new Node();
```

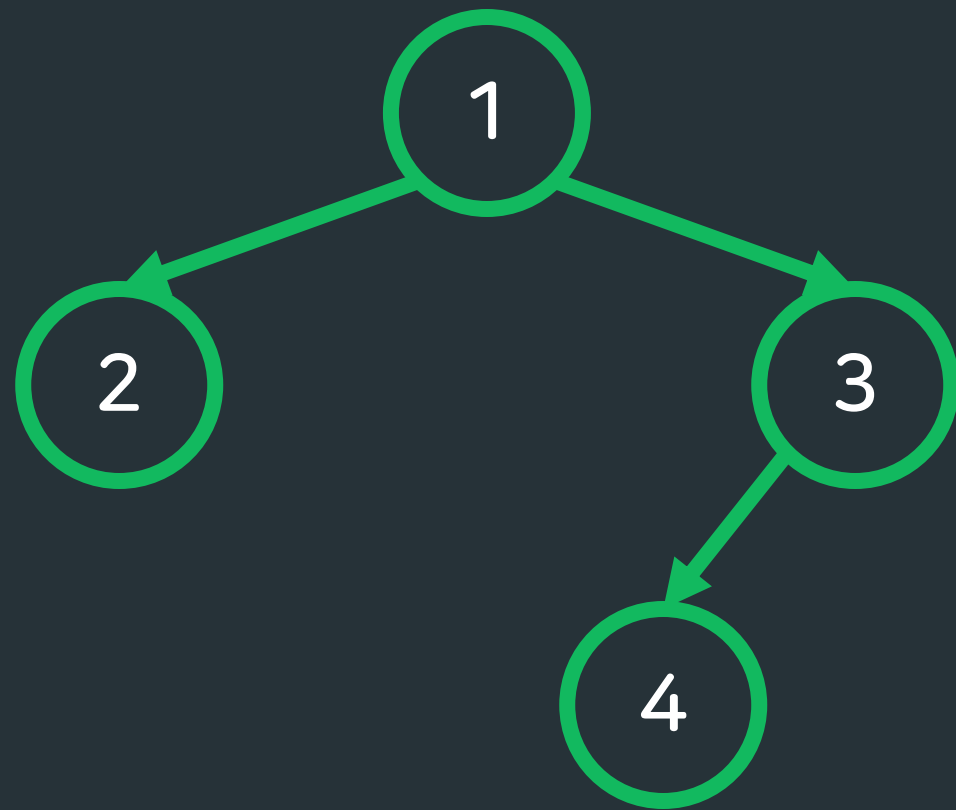
```
n1->data = 1;  
n1->left = n2;  
n1->right = n3;
```

```
n2->data = 2;  
n2->left = n2->right = NULL;
```

```
n3->data = 3;  
n3->left = n4;  
n3->right = NULL;
```

```
n4->data = 4;  
n4->left = n4->right = NULL;  
}
```

# 트리 구현 (이진 트리)



맵

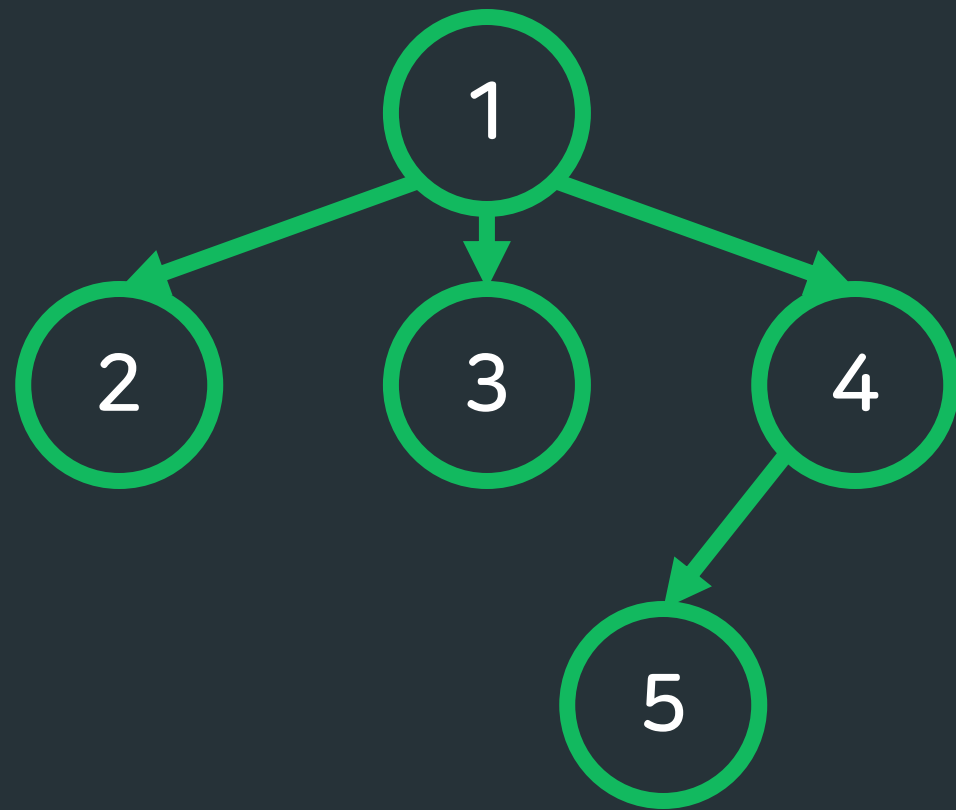
이미 트리 관계가 정의되어 있을 때 적합



```
int main() {  
    map<int, pair<int, int>> tree;  
  
    tree[1] = {2, 3};  
    tree[2] = {-1, -1};  
    tree[3] = {4, -1};  
    tree[4] = {-1, -1};  
}
```



# 트리 구현 (일반 트리)



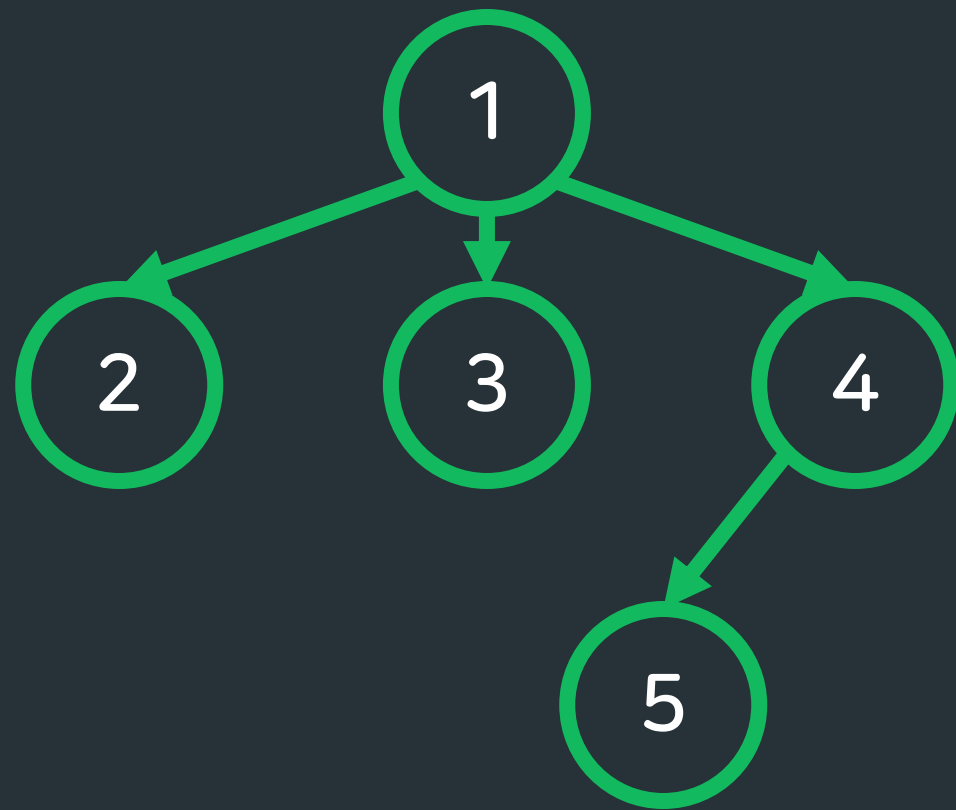
맵

정점 번호가 연속하지 않을 때 적합



```
int main() {  
    map<int, vector<int>> tree;  
  
    tree[1] = {2, 3, 4};  
    tree[4] = {5};  
}
```

# 트리 구현 (일반 트리)



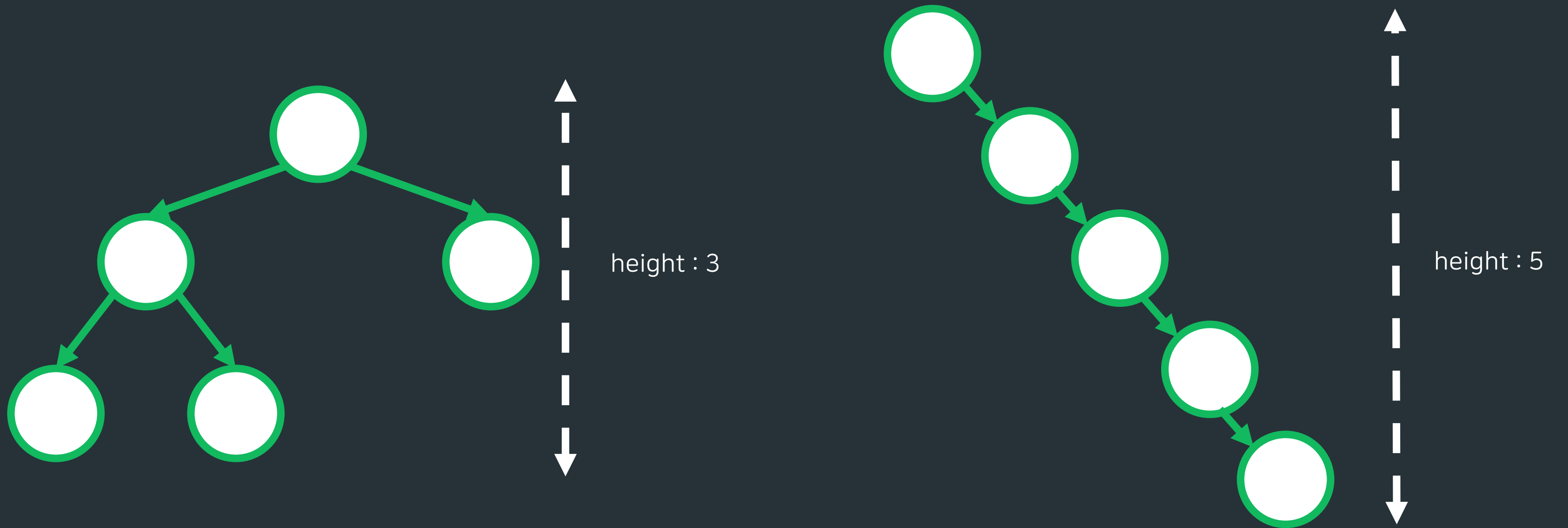
2차원 벡터

정점 번호가 연속할 때 적합

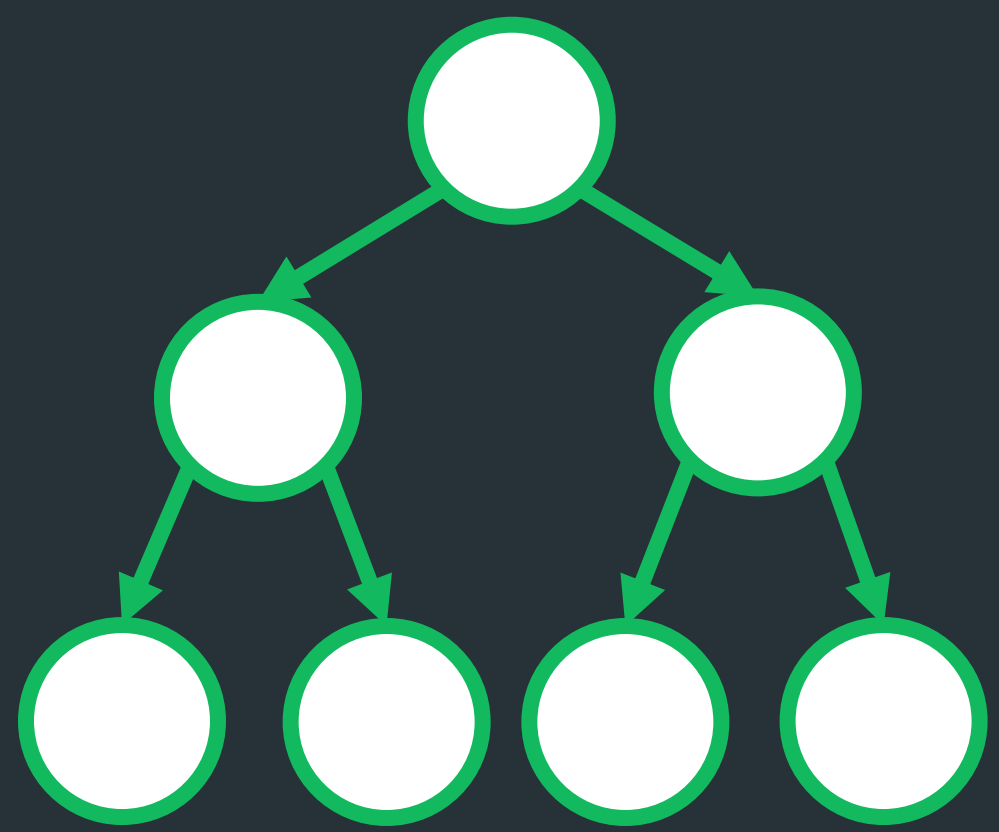


```
int main() {  
    vector<vector<int>> tree;  
  
    tree[1] = {2, 3, 4};  
    tree[4] = {5};  
}
```

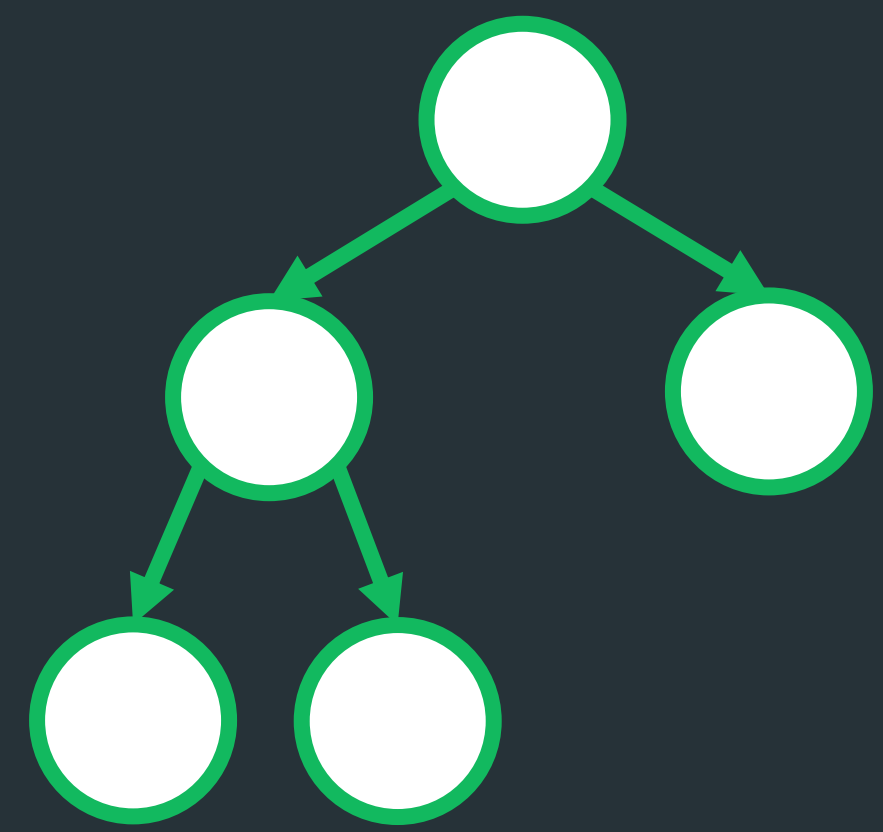
# 이진 트리의 특징



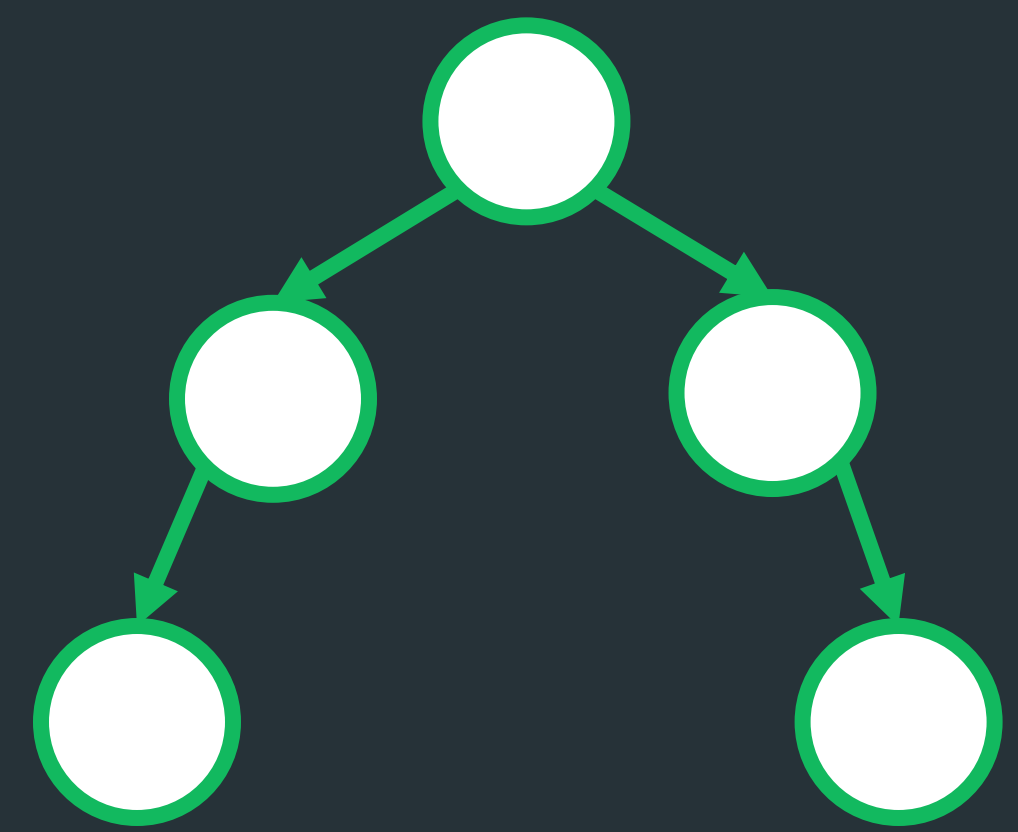
$(\text{ceil}) \log(V+1) \leq \text{이진 트리의 높이} \leq V$   
 $O((\text{ceil}) \log(V+1)) \leq \text{이진 트리의 시간 복잡도} \leq O(V)$



Full Binary Tree

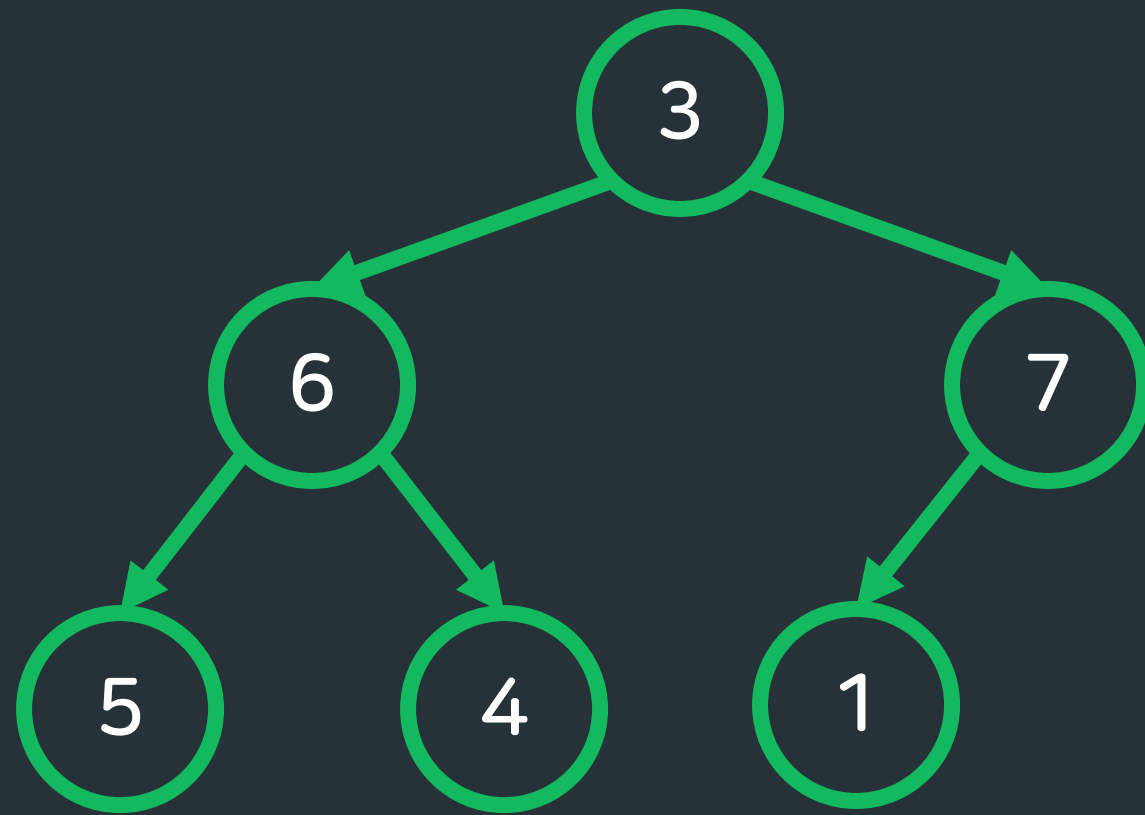


Complete Binary Tree

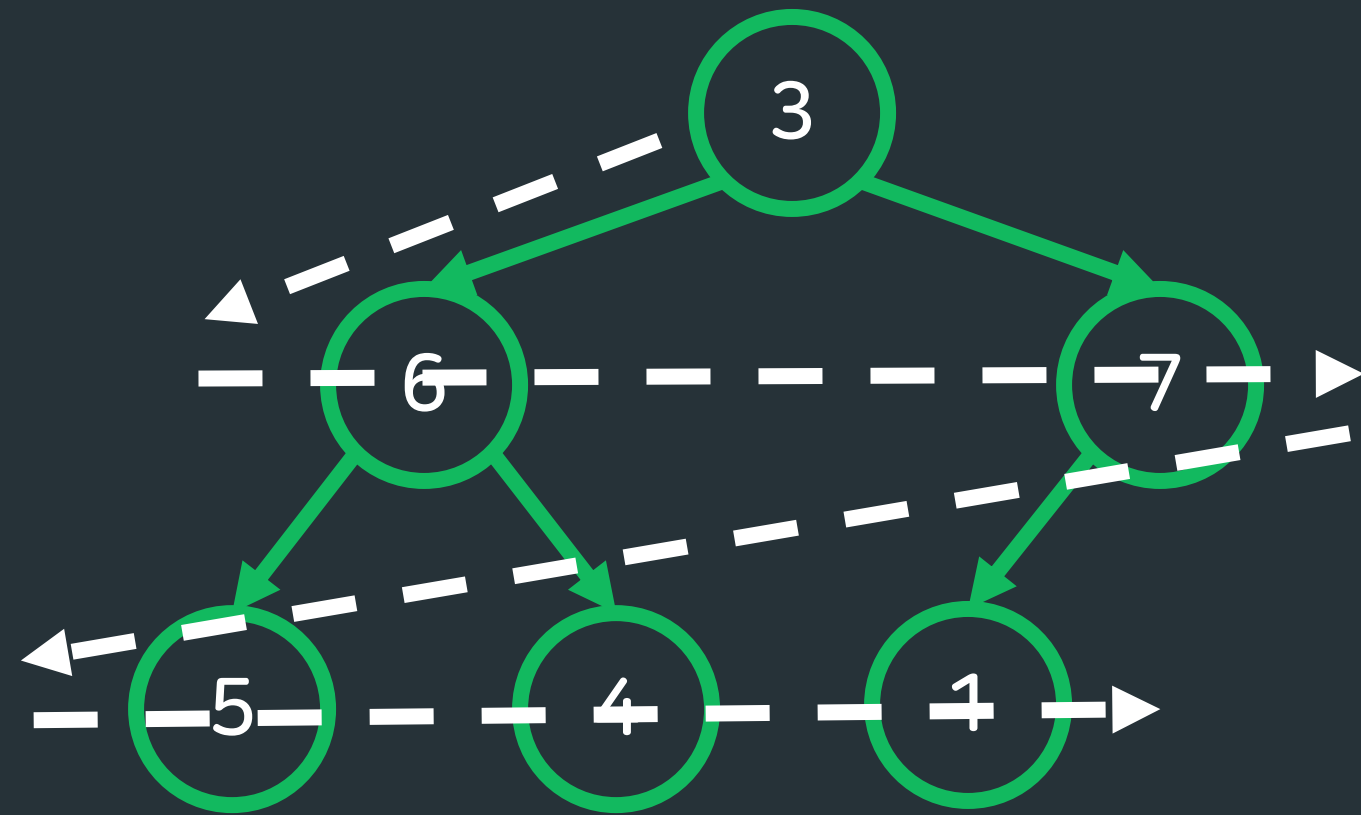


Other Binary Tree

레벨 순회 (Level traversal)  
전위 순회 (Preorder traversal)  
중위 순회 (Inorder traversal)  
후위 순회 (Postorder traversal)



```
level (root)
{
    while (!q.empty())
        v = q.front();
        if (v == null)
            continue;
        print(v);
        q.push (left(v));
        q.push (right(v));
}
```

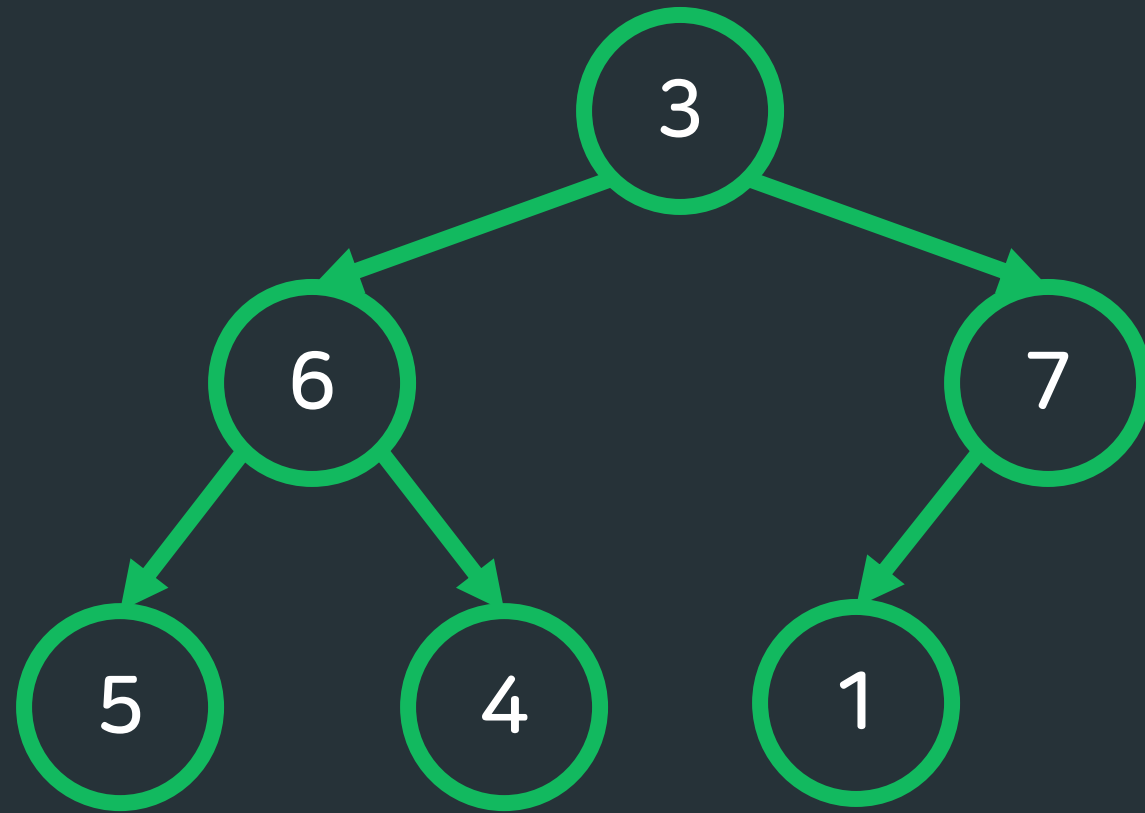


3 6 7 5 4 1

\*일반 트리도 가능



```
level (root)
{
    while (!q.empty())
        v = q.front();
        if (v == null)
            continue;
        print(v);
        q.push (left(v));
        q.push (right(v));
}
```



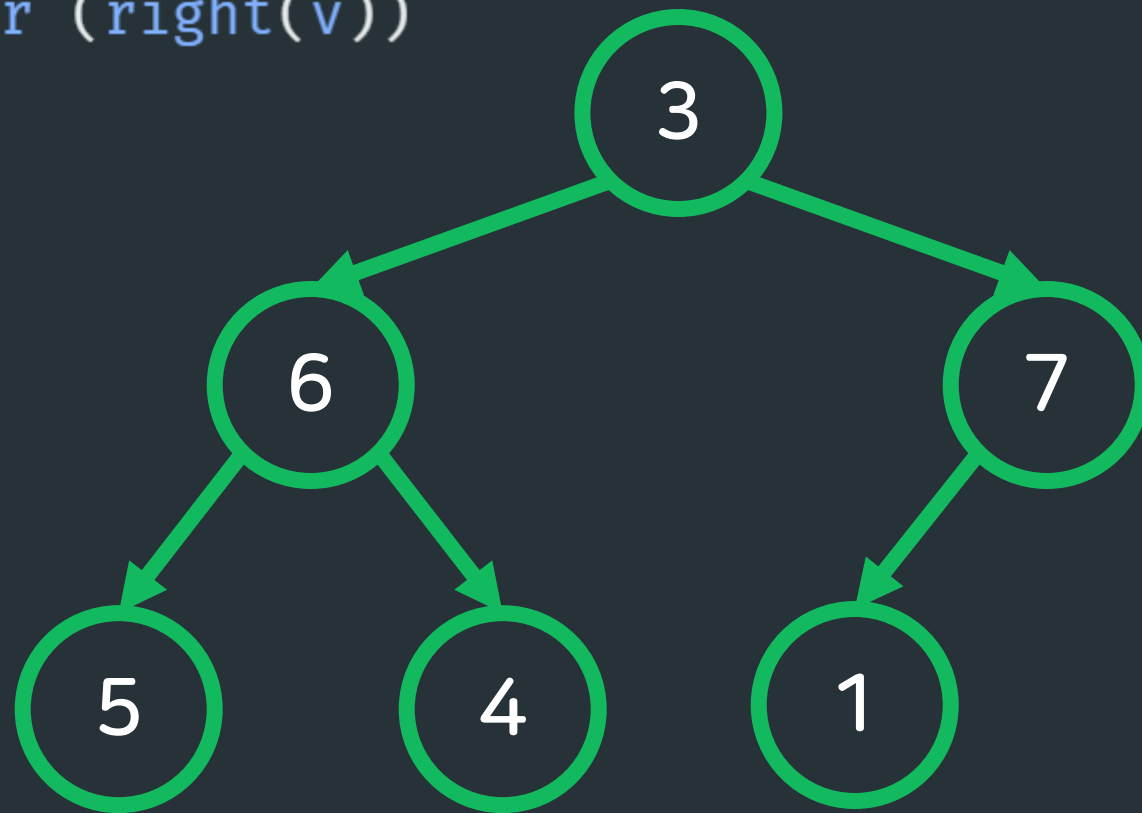
```
preorder (v)
{
    if (v ≠ null)
        print (v)
        preorder (left(v))
        preorder (right(v))
}
```



# 전위 순회

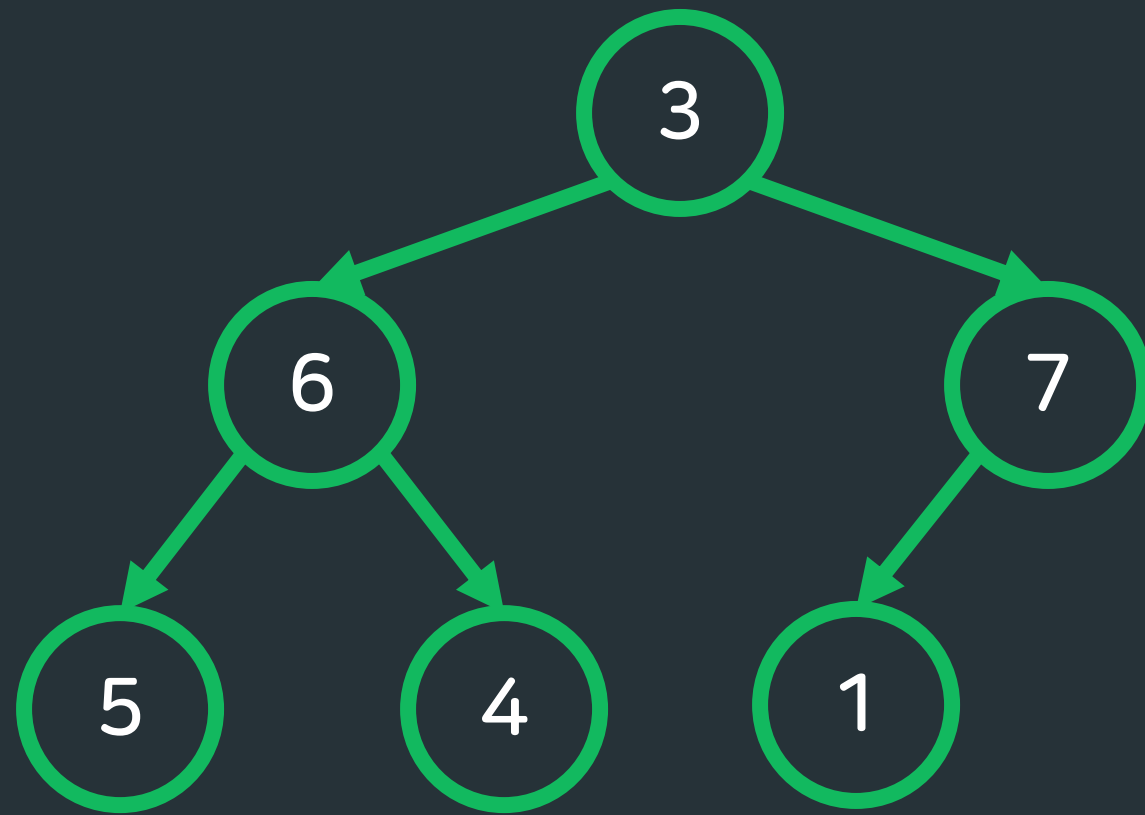


```
preorder (v)
{
    if (v ≠ null)
        print (v)
        preorder (left(v))
        preorder (right(v))
}
```



3 6 5 4 7 1

```
preorder(3)
  print(3)
  preorder(3 -> left : 6)
    print(6)
    preorder(6 -> left : 5)
      print(5)
      preorder(5 -> left : null)
      preorder(5 -> right : null)
    preorder(6 -> right : 4)
      print(4)
      preorder(4 -> left : null)
      preorder(4 -> right : null)
  preorder(3 -> right : 7)
    print(7)
    preorder(7 -> left : 1)
      print(1)
      preorder(1 -> left : null)
      preorder(1 -> right : null)
    preorder(7 -> right : null)
```

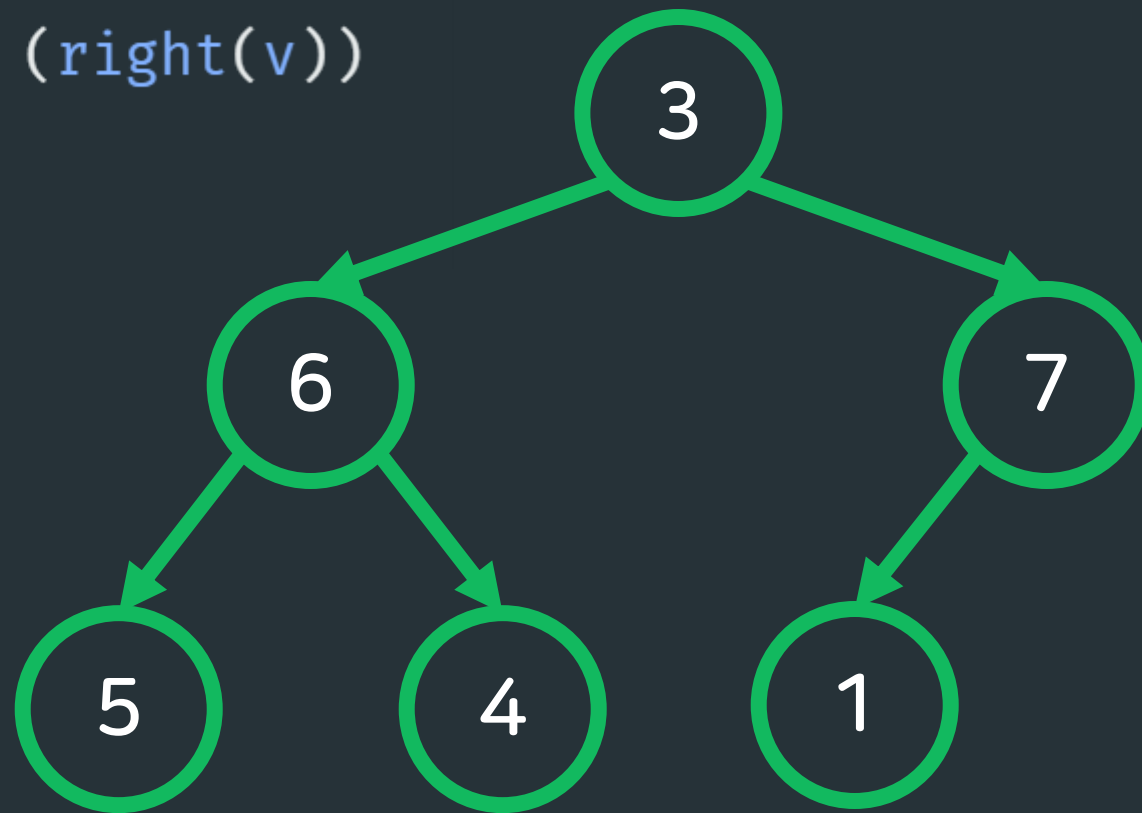


```
inorder (v)
{
    if (v ≠ null)
        inorder (left(v))
        print (v)
        inorder (right(v))
}
```

# 중위 순회

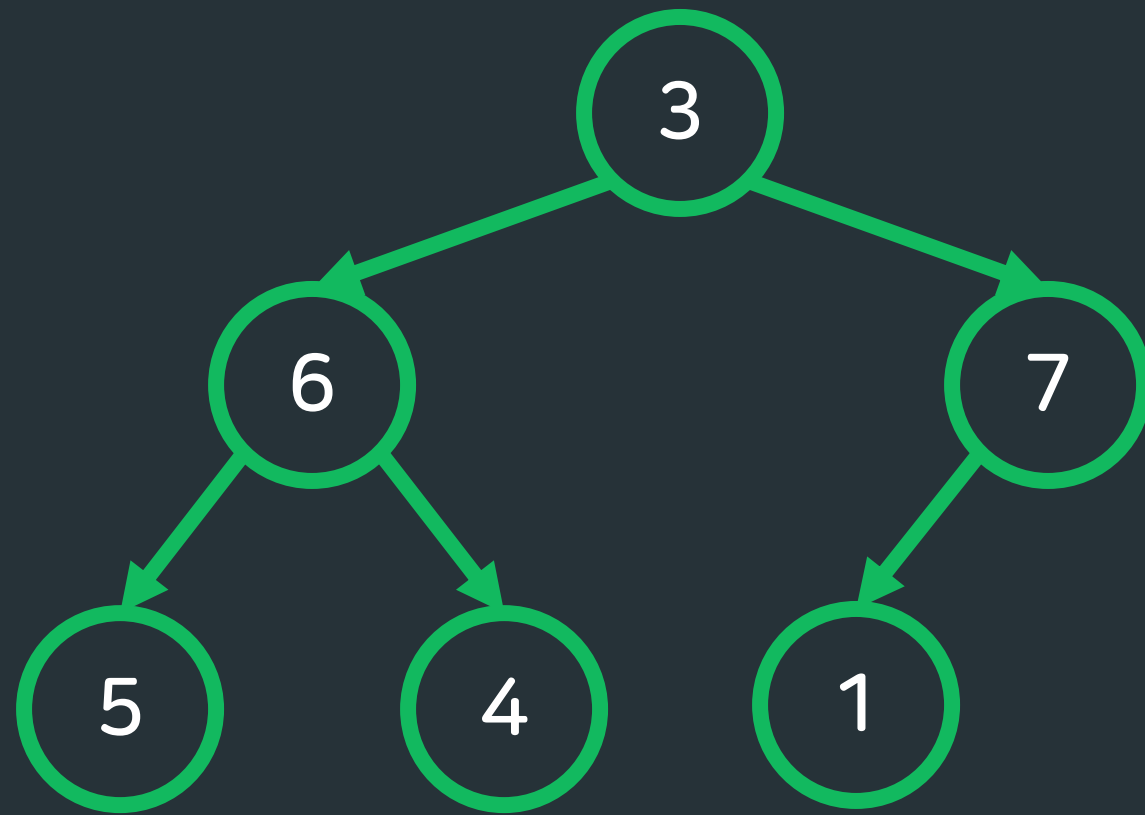


```
inorder (v)
{
    if (v ≠ null)
        inorder (left(v))
        print (v)
        inorder (right(v))
}
```



5 6 4 3 1 7

```
inorder(3)
  inorder(3 -> left : 6)
    inorder(6 -> left : 5)
      inorder(5 -> left : null)
      print(5)
      inorder(5 -> right : null)
    print(6)
    inorder(6 -> right : 4)
      inorder(4 -> left : null)
      print(4)
      inorder(4 -> right : null)
  print(3)
  inorder(3 -> right : 7)
    inorder(7 -> left : 1)
      inorder(1 -> left : null)
      print(1)
      inorder(1 -> right : null)
    print(7)
    inorder(7 -> right : null)
```

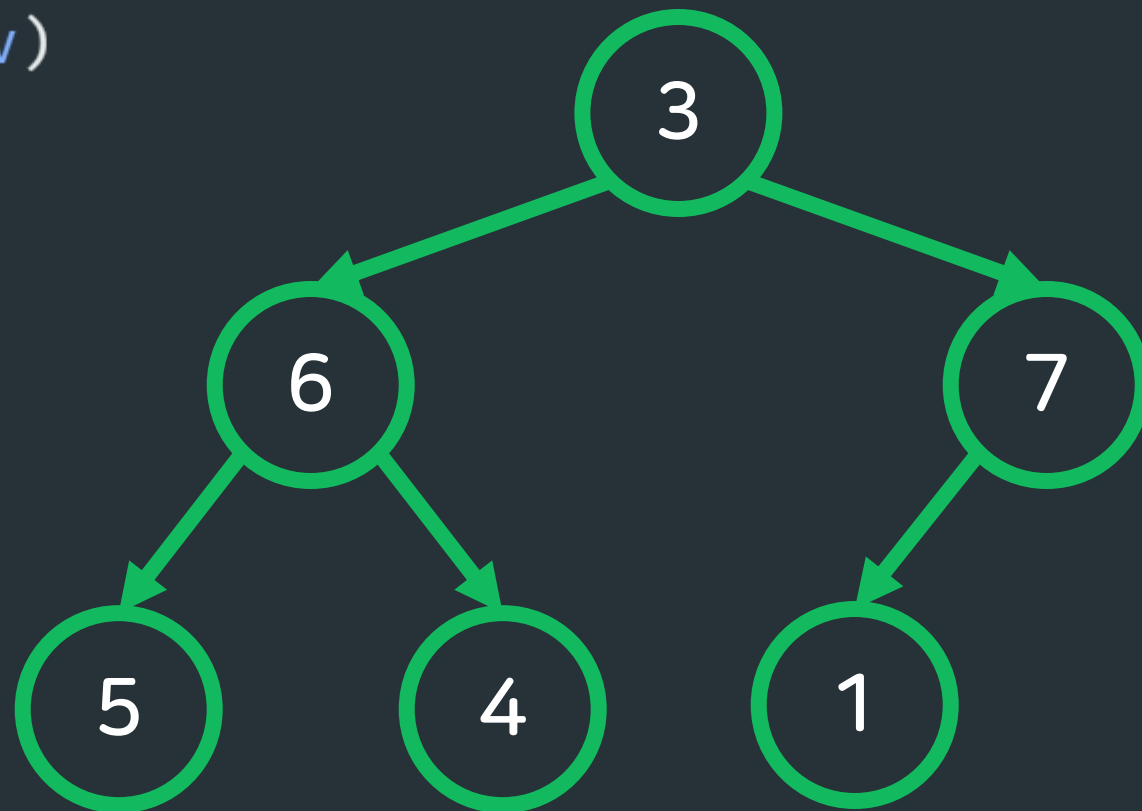


```
postorder (v)
{
    if (v ≠ null)
        postorder (left(v))
        postorder (right(v))
        print (v)
}
```

# 후위 순회



```
postorder (v)
{
    if (v ≠ null)
        postorder (left(v))
        postorder (right(v))
        print (v)
}
```

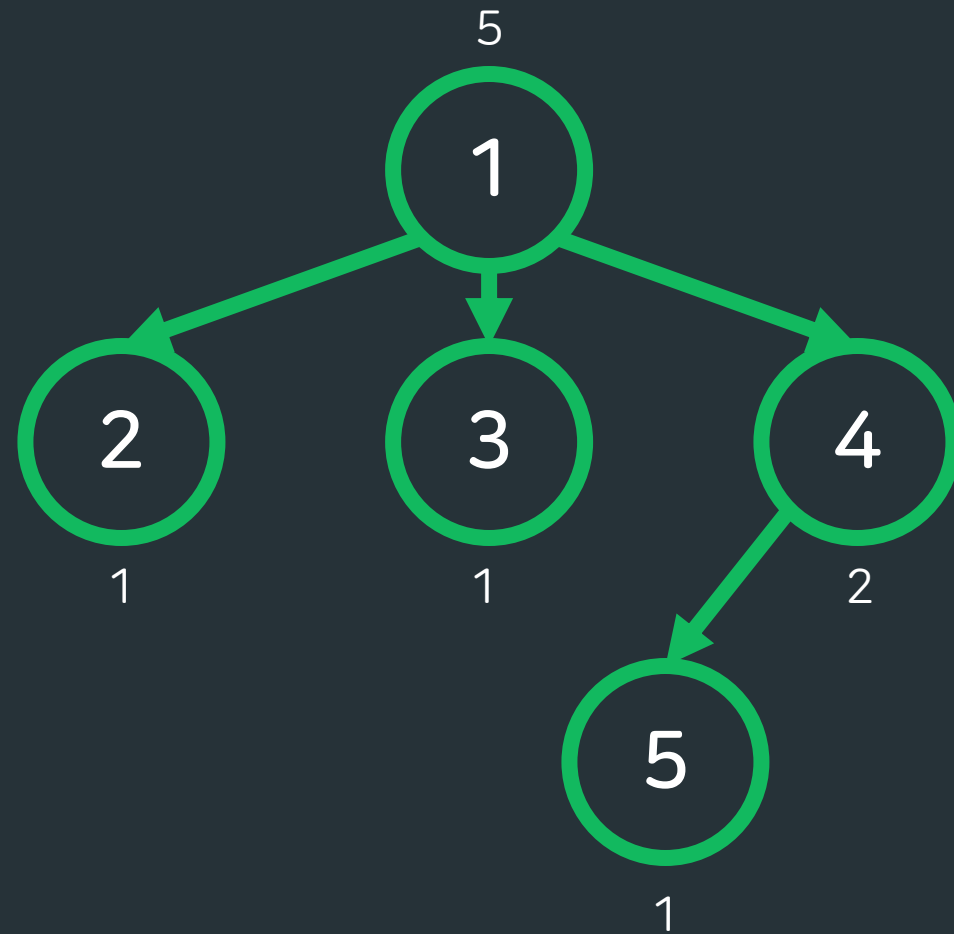


5 4 6 1 7 3

```
postorder(3)
  postorder(3 -> left : 6)
    postorder(6 -> left : 5)
      postorder(5 -> left : null)
      postorder(5 -> right : null)
      print(5)
    postorder(6 -> right : 4)
      postorder(4 -> left : null)
      postorder(4 -> right : null)
      print(4)
    print(6)
  postorder(3 -> right : 7)
    postorder(7 -> left : 1)
      postorder(1 -> left : null)
      postorder(1 -> right : null)
      print(1)
    postorder(7 -> right : null)
    print(7)
  print(3)
```

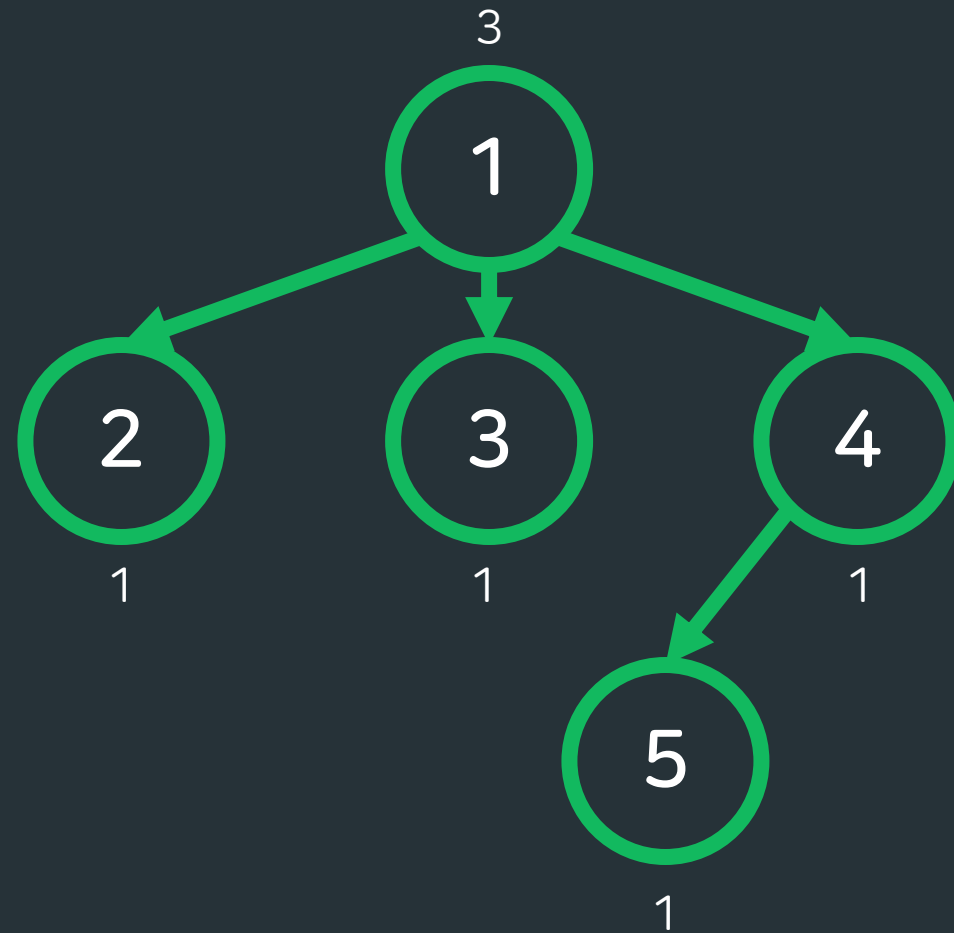
트리의 정점 수 구하기  
리프 노드의 수 구하기  
트리의 높이 구하기

# 트리의 정점 수 구하기



```
int nodeCnt(int node){  
    int cnt = 1;  
    for(int child : tree[node])  
        cnt += nodeCnt(child);  
    return cnt;  
}
```

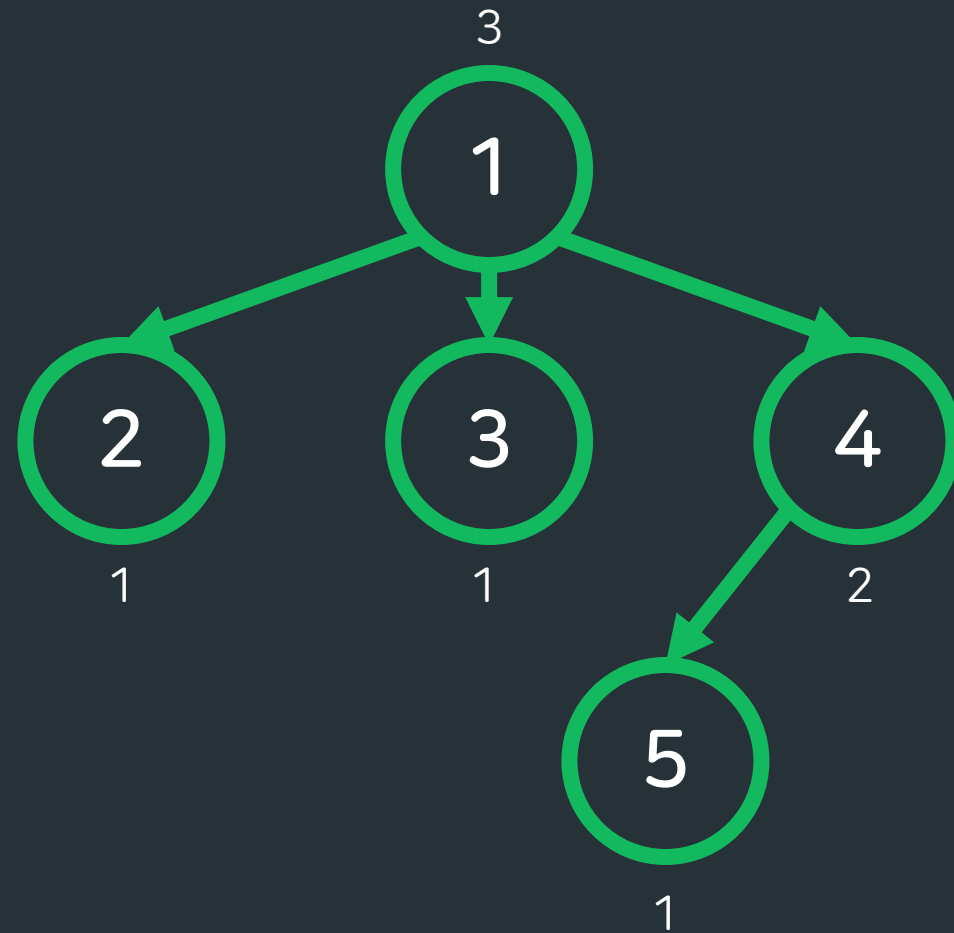
# 리프 노드의 수 구하기



```
int leafCnt(int node){  
    if(tree[node].empty())  
        return 1;  
  
    int cnt = 0;  
    for(int child : tree[node])  
        cnt += leafCnt(child);  
    return cnt;  
}
```



# 트리의 높이 구하기



```
int treeHeight(int node){  
    int height = 0;  
    for(int child : tree[node])  
        height = max(height, treeHeight(child));  
    return height + 1;  
}
```

## /<> 1991번 : 트리 순회 - Silver 1

### 문제

- 이진 트리의 전위 순회, 중위 순회, 후위 순회 결과를 출력하라

### 제한 사항

- 정점의 개수  $N$ 은  $1 \leq N \leq 26$
- 모든 정점은 알파벳 대문자
- 루트 정점은 항상 A

## 예제 입력 1

```
7
A B C
B D .
C E F
E ..
F . G
D ..
G ..
```

## 예제 출력 1

```
ABDCEFG
DBAECFG
DBEGFCA
```

## /<> 4803번 : 트리 - Gold 4

### 문제

- 그래프가 주어질 때, **트리의 개수**를 출력하라

### 제한 사항

- 정점의 개수  $n$ 은  $0 \leq n \leq 500$
- 간선의 개수  $m$ 은  $0 \leq m \leq n(n-1)/2$
- 입력은 **무방향 그래프**

## 예제 입력 1

```
6 3
1 2 2 3 3 4
6 5
1 2 2 3 3 4 4 5 5 6
6 6
1 2 2 3 1 3 4 5 5 6 6 4
0 0
```

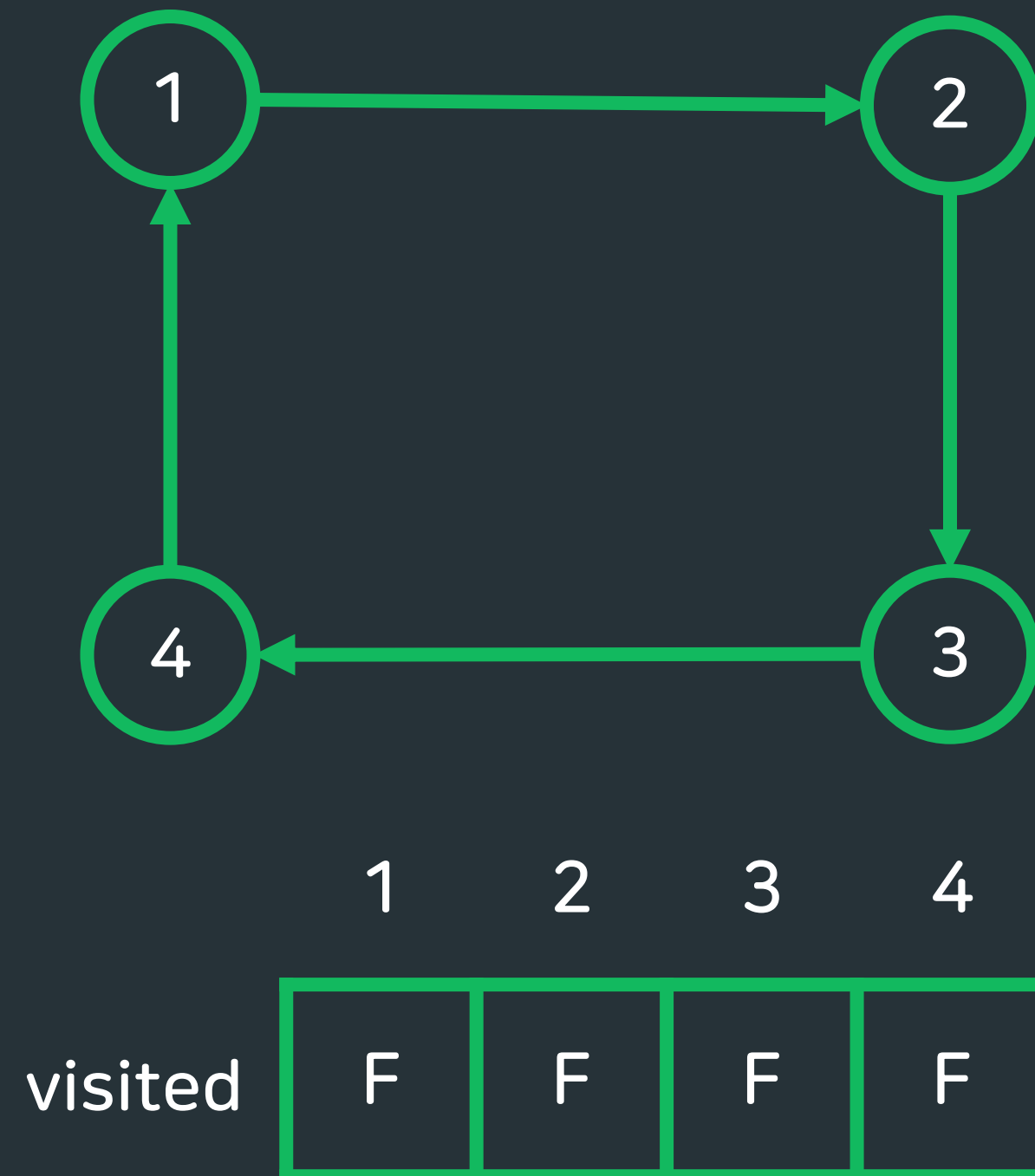
## 예제 출력 1

Case 1: A forest of 3 trees.  
Case 2: There is one tree.  
Case 3: No trees.

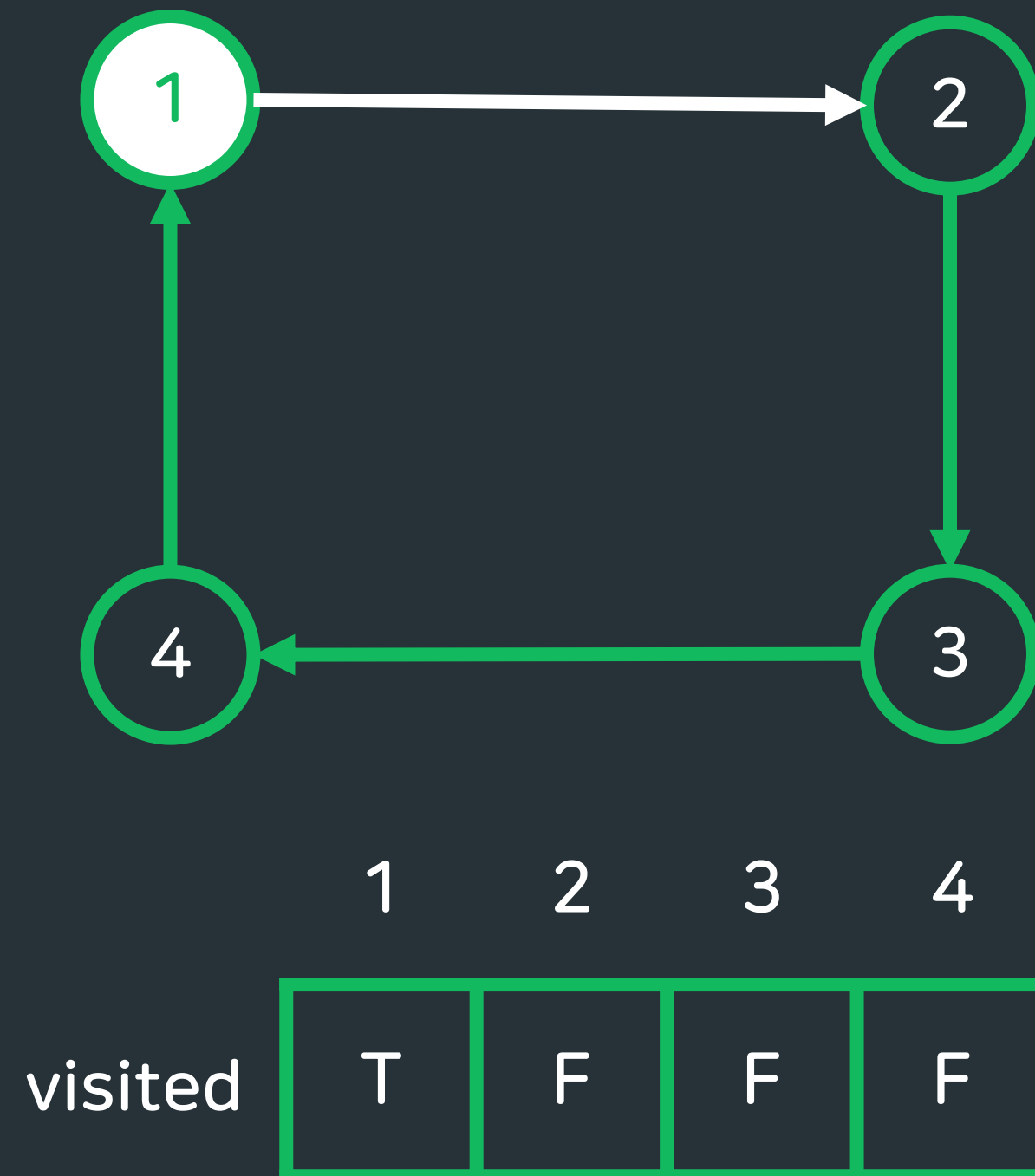
## Hint

1. 사이클 여부를 판단하는 게 중요해요.
2. 사이클이 생성되는 순간을 그려보세요.
3. 트리 순회에는 DFS와 BFS를 사용한다고 했어요. 여기서 뭘 써야 할까요?

# 사이클이 생성되는 순간

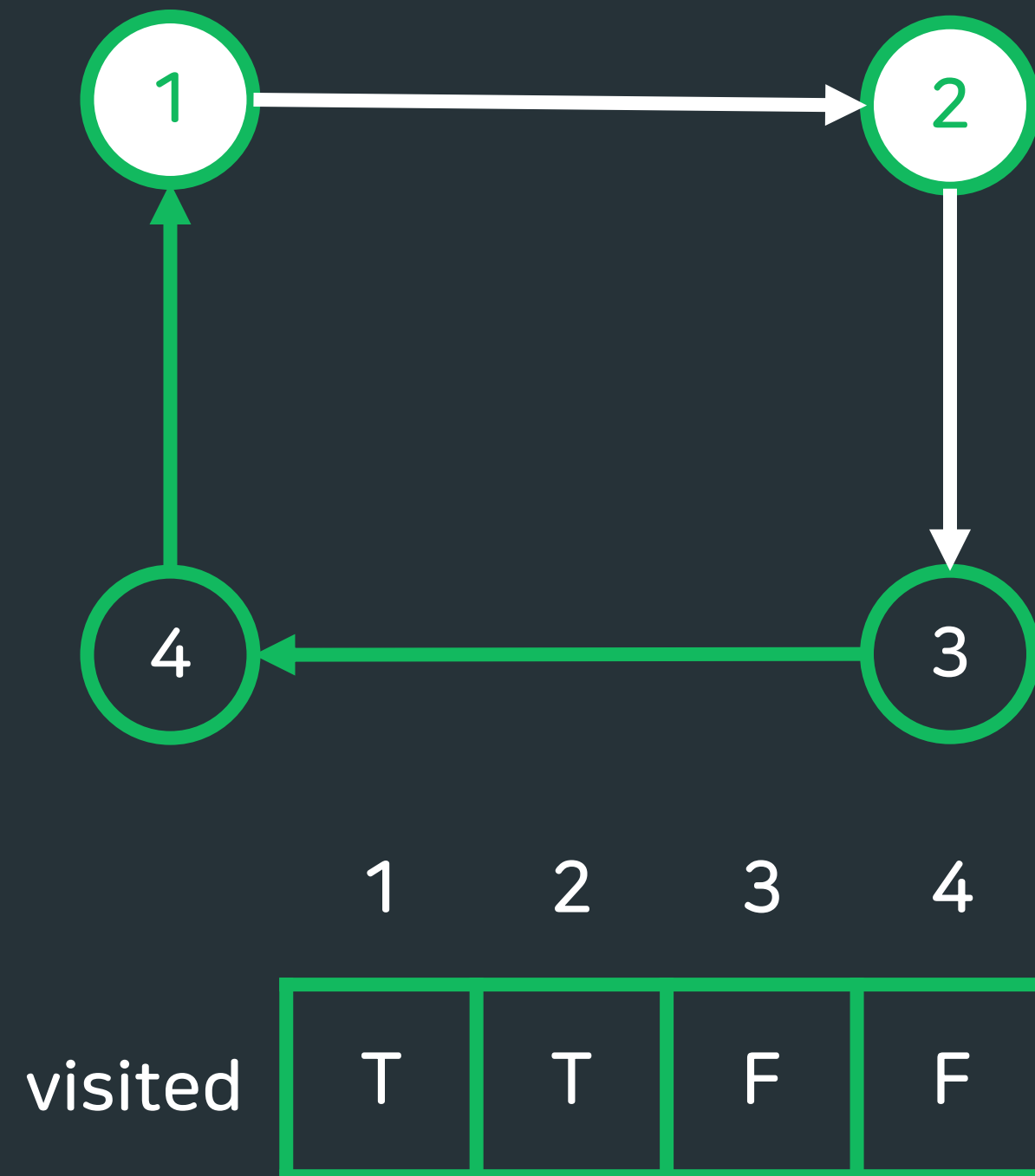


# 사이클이 생성되는 순간

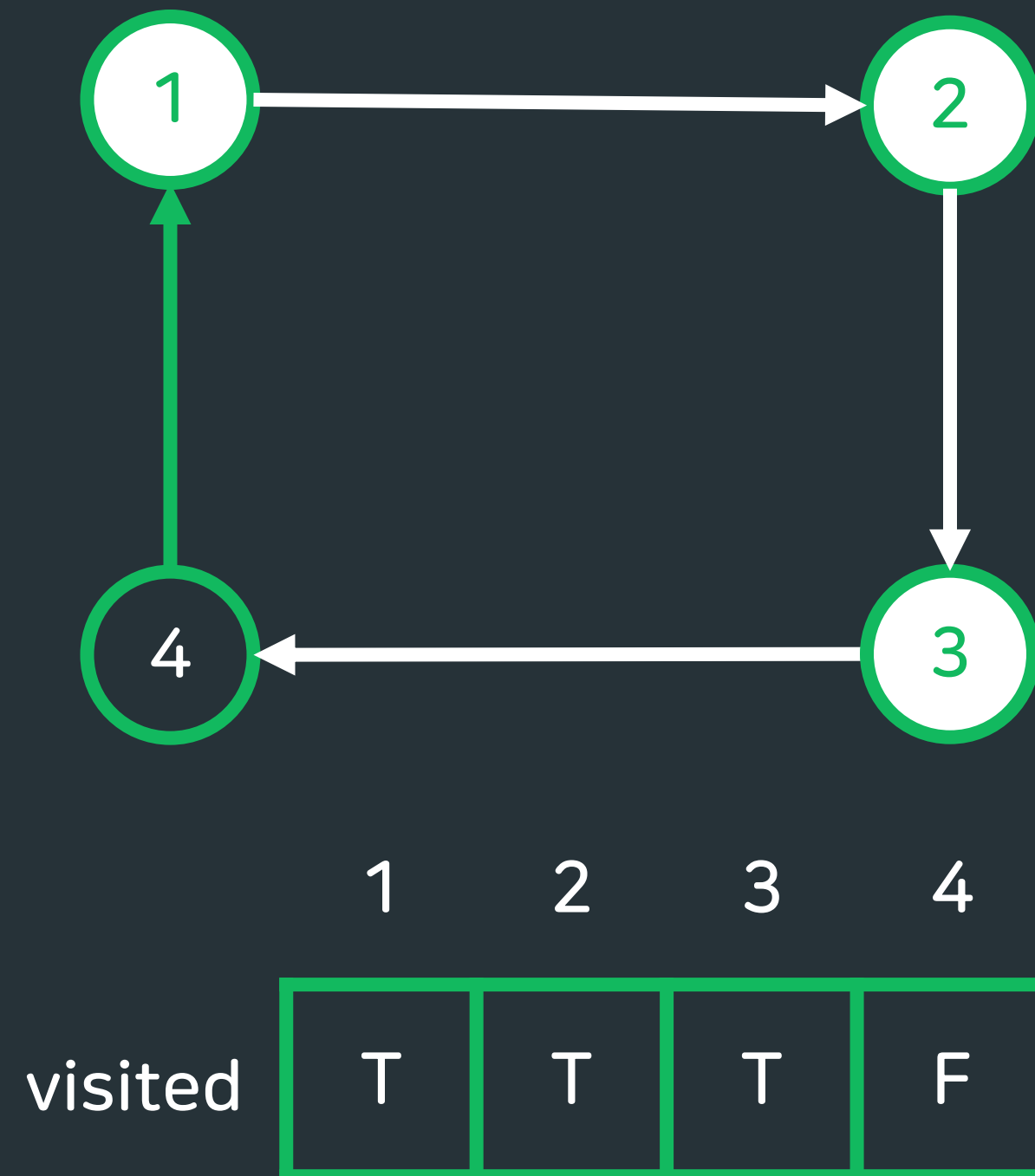




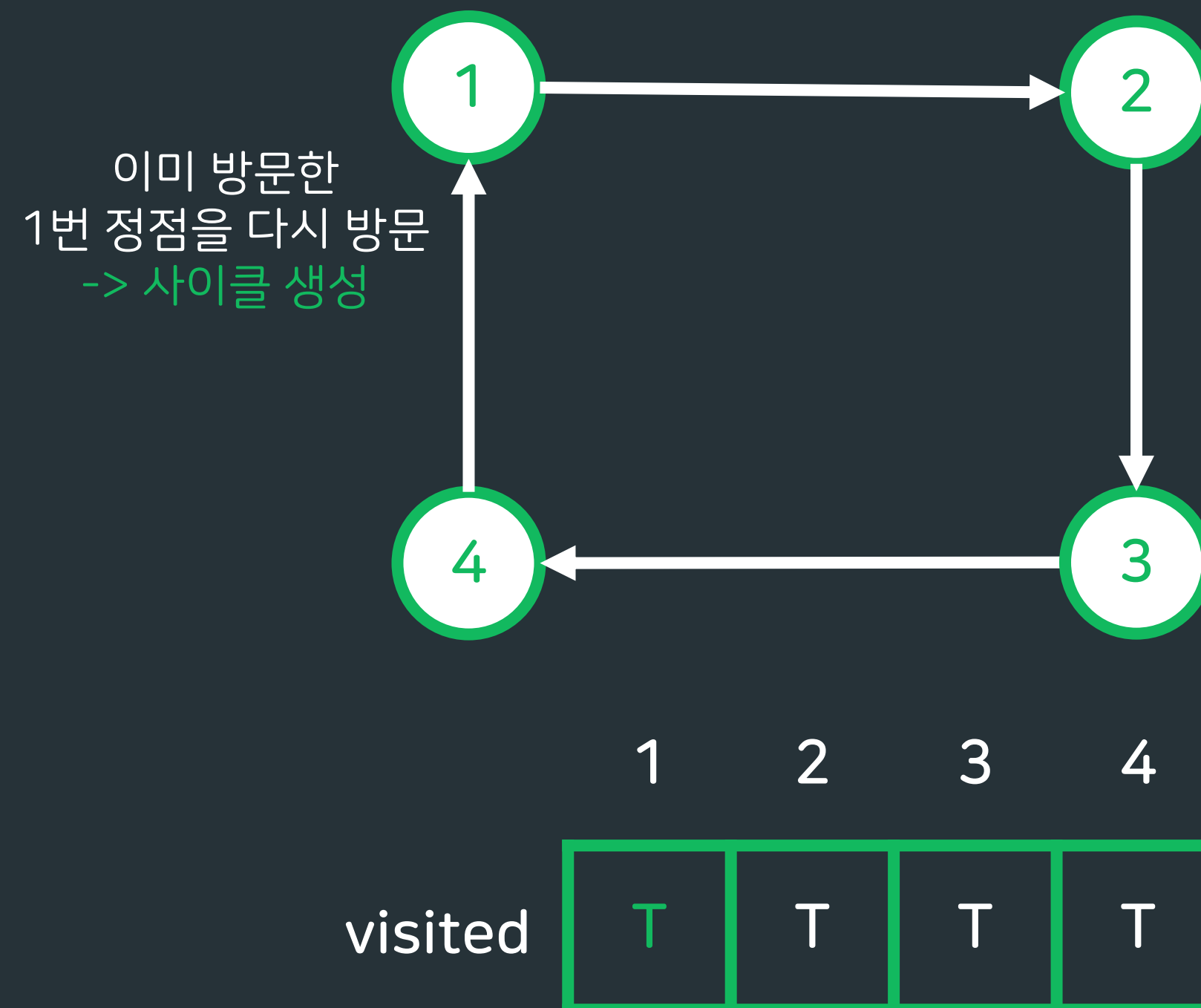
# 사이클이 생성되는 순간



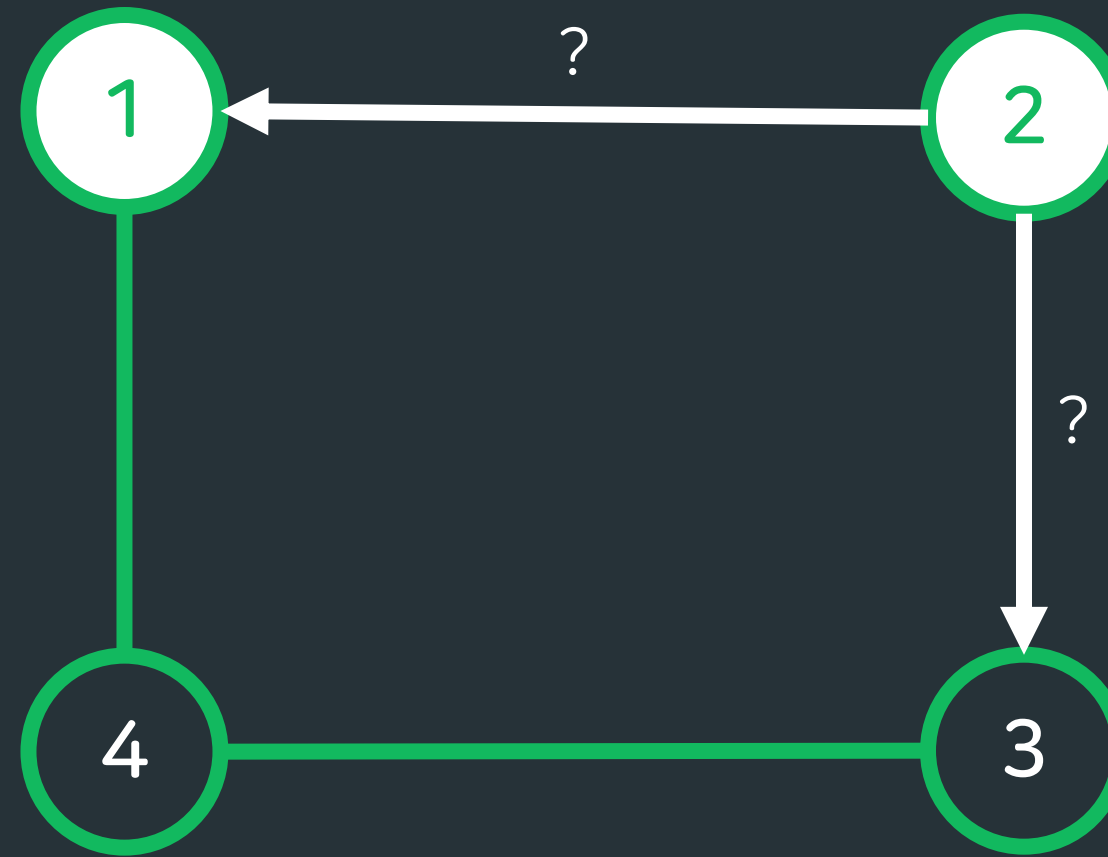
# 사이클이 생성되는 순간



# 사이클이 생성되는 순간



## 입력은 무방향 그래프인데...



바로 직전에 탐색한 정점(부모 정점)을 기억해두면  
직전 정점을 다시 탐색해 사이클로 오해하는 일이 없음!  
-> BFS는 직전에 탐색한 정점이 부모 정점이란 보장이 없으므로 불가능

## 2021 KAKAO BLIND RECRUITMENT : 매출 하락 최소화 - Level 4

### 문제

- 회사 조직도가 주어진다.
- 2개의 팀에 소속된 직원은 한 팀에선 팀장, 한 팀에선 팀원이다.
- 각 팀(서브 트리)에서 최소 1명 이상의 직원이 워크숍에 참가해야 할 때, 워크숍에 참가하는 직원들의 하루평균 매출액의 합이 최소가 되게 하라.

### 제한 사항

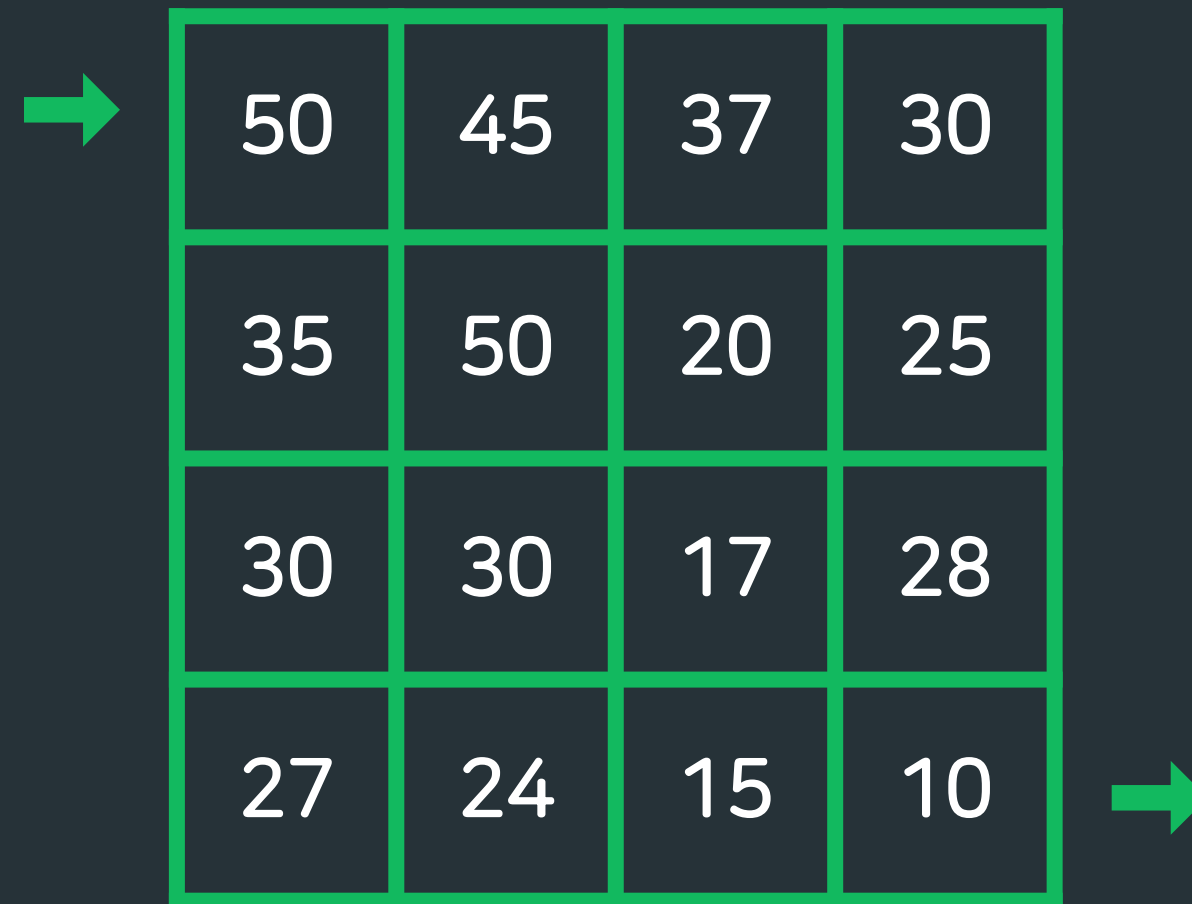
- 직원의 수를  $V$ 라고 할 때,  $V$ 는  $2 \leq V \leq 300,000$
- 1번 직원은 항상 CEO (루트 정점)

## Hint

1. CEO를 제외하곤 모두 팀장이 있어요.
2. 만약 팀장이 워크숍에 참가한다면 팀원들은 워크숍에 참가해도 되고 안해도 돼요.
3. 만약 팀장이 워크숍에 불참한다면 팀원들 중 1명 이상이 워크숍에 참가해야 해요.
4. 모든 사람들은 참가하거나, 불참하거나 둘 중 하나의 상태예요.
5. 모든 경우를 기억하는 방법은 어떤 것이었나요?

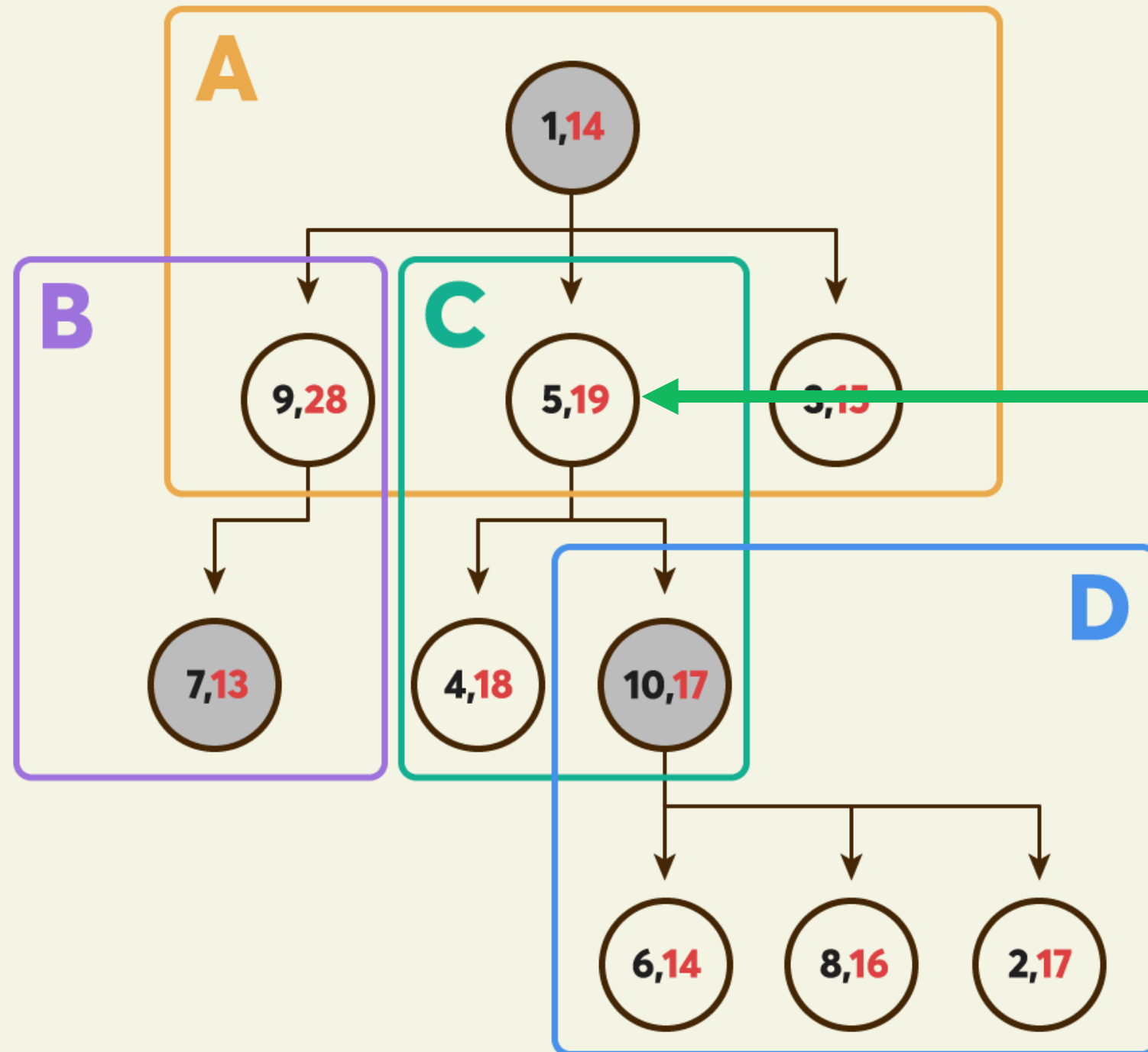
## 이 문제가 기억나시나요?

- $N \times M$ 으로 표현되는 지도가 있고 각 칸마다 지점의 **높이**가 쓰여있다.
- (1,1)에서 시작해서 (N,M)까지 항상 **높이가 더 낮은 지점으로만 이동**할 때, 가능한 **이동 경로의 개수**를 구하여라



50	45	37	30
35	50	20	25
30	30	17	28
27	24	15	10

- BFS로 가능한 이동 경로를 계속 구해 나가면, 완전탐색에 가까우므로 **시간초과**
- 사실 DFS로 풀어도 **시간초과**



만약 5번 직원이 **참가**한다면  
4, 10번 직원은 참가해도 되고 안해도 됨

만약 5번 직원이 **불참**한다면  
4, 10번 직원 중 1명 이상은 반드시 참가해야 함

dp 배열에 각 직원의 가능한 모든 상태의 결과를 저장



## 정리

- 그래프의 부분집합인 트리
- 그래프와 트리의 차이점을 잘 기억해두기
- 이진 트리와 일반 트리로 나눌 수 있고, 이진 트리는 전위 & 중위 & 후위 순회 가능
- 기본적으로 그래프의 한 종류라서 DFS, BFS 탐색 가능

## 이것도 알아보세요

- 일반 트리를 이진 트리로 바꿀 수도 있습니다. 어떻게 하면 될까요?
- 우리에게 익숙한 BST에서 파생된 여러 트리들에 대해 알아보세요.  
AVL Tree, Red-Black Tree, B Tree

## 필수

- /<> 14503번 : 로봇 청소기 - Gold 5
- /<> 2011번 : 암호코드 - Silver 1

## 3문제 이상 선택

- /<> 1967번 : 트리의 지름 - Gold 4
- /<> 2250번 : 트리의 높이와 너비 - Gold 2
- /<> 5639번 : 이진 검색 트리 - Silver 1
- /<> 14675번 : 단절점과 단절선 - Gold 5
- /<> 15681번 : 트리와 쿼리 - Gold 5
- /<> 20924번 : 트리의 기둥과 가지 - Gold 4