2021 겨울 신촌 연합 알고리즘 캠프

트리(Tree)

초급 알고리즘

HI-ARC 김기선

트리(Tree)

그래프의 일종

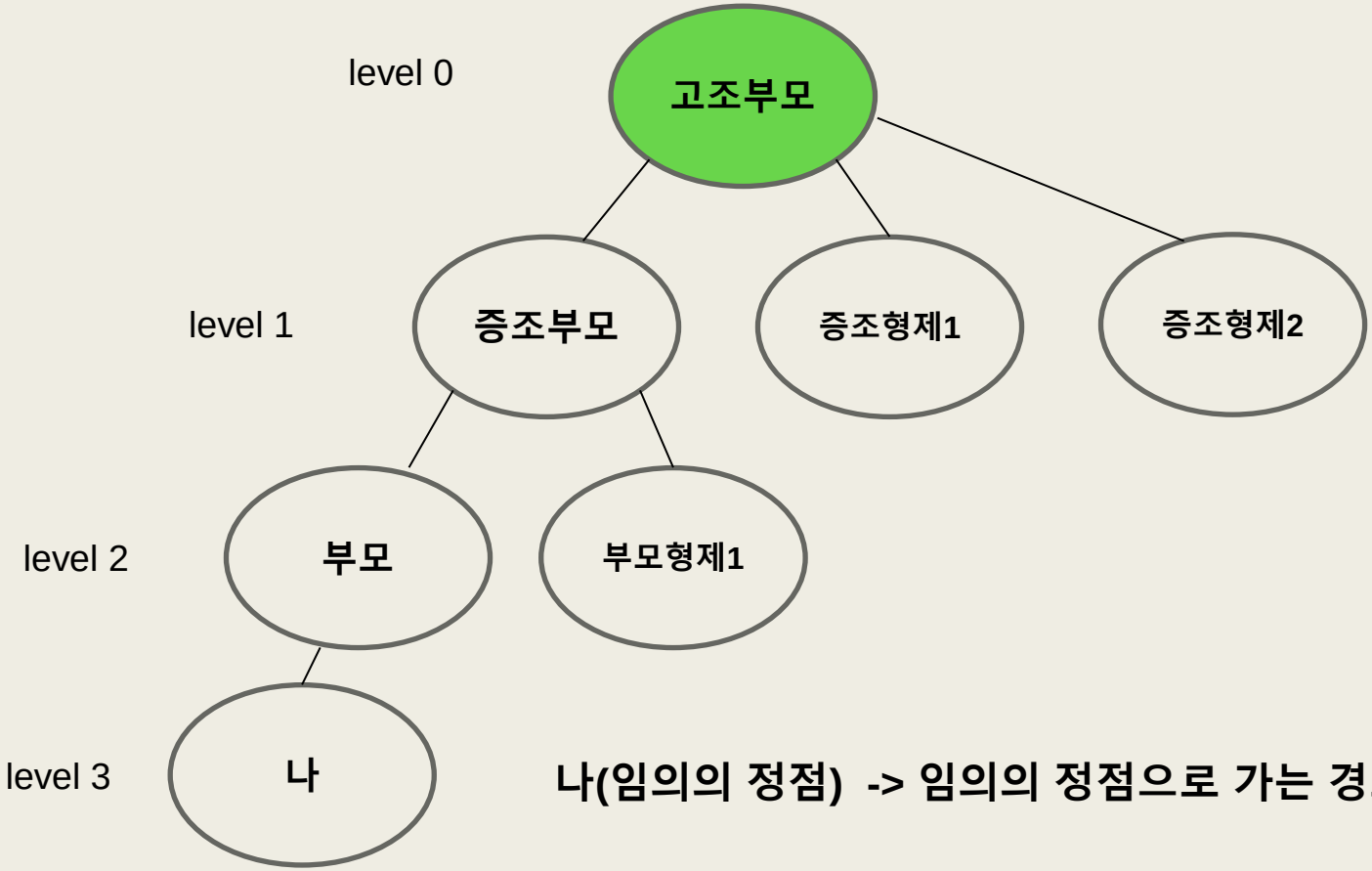
- 모든 정점끼리 연결되어 있어야 한다.
- 방향을 무시했을 때 사이클이 존재하지 않아야 한다.

위 두가지를 만족하는 정점 N개의 그래프는 항상 N-1개의 간선을 갖는다.

-> 트리의 간선 개수는 정점-1개이다.

트리의 특징

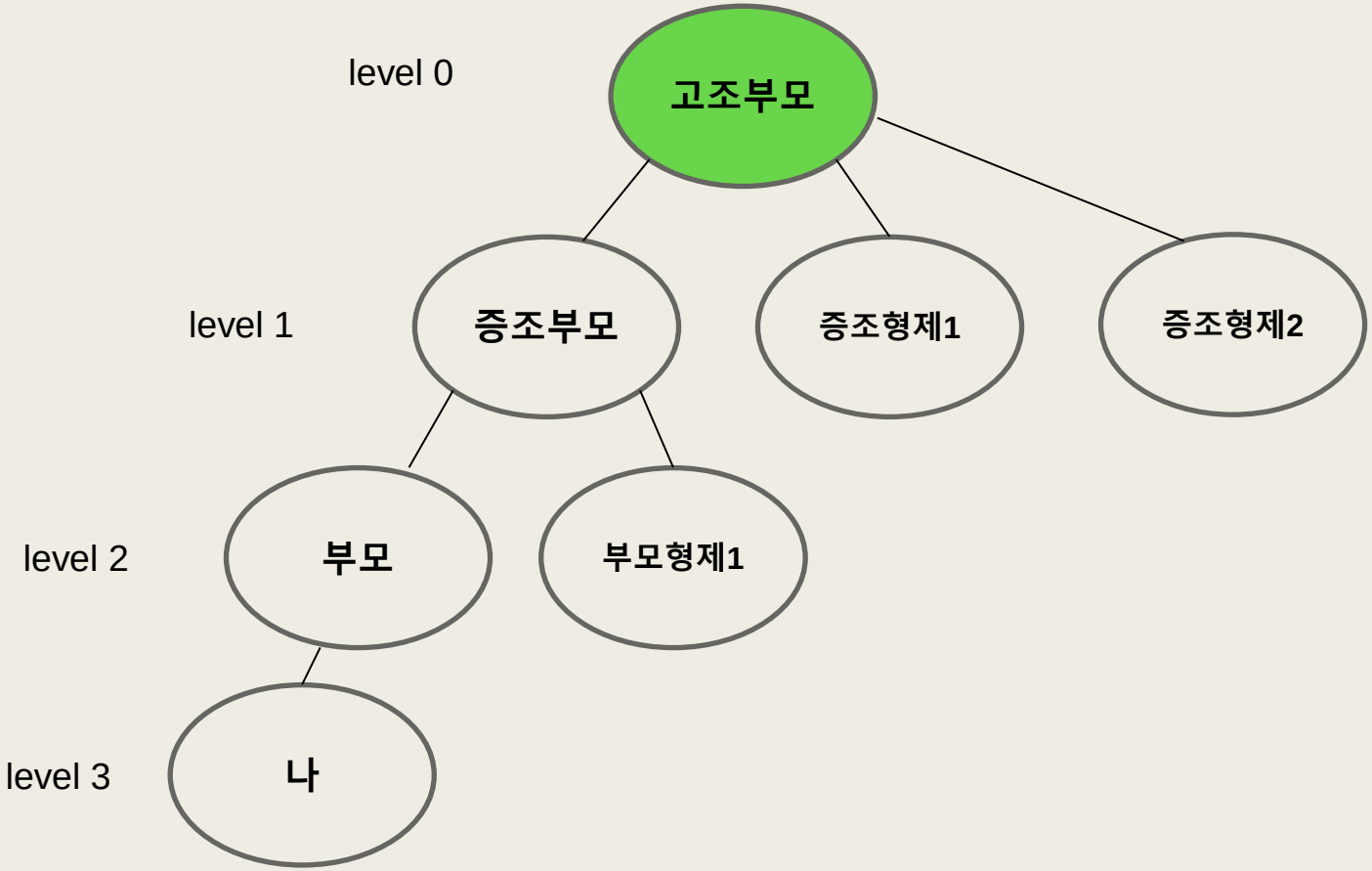
계층 구조를 이루는 자료를 표현하는데 유용



나(임의의 정점) -> 임의의 정점으로 가는 경로는 유일하다.

트리의 특징

계층 구조를 이루는 자료를 표현하는데 유용



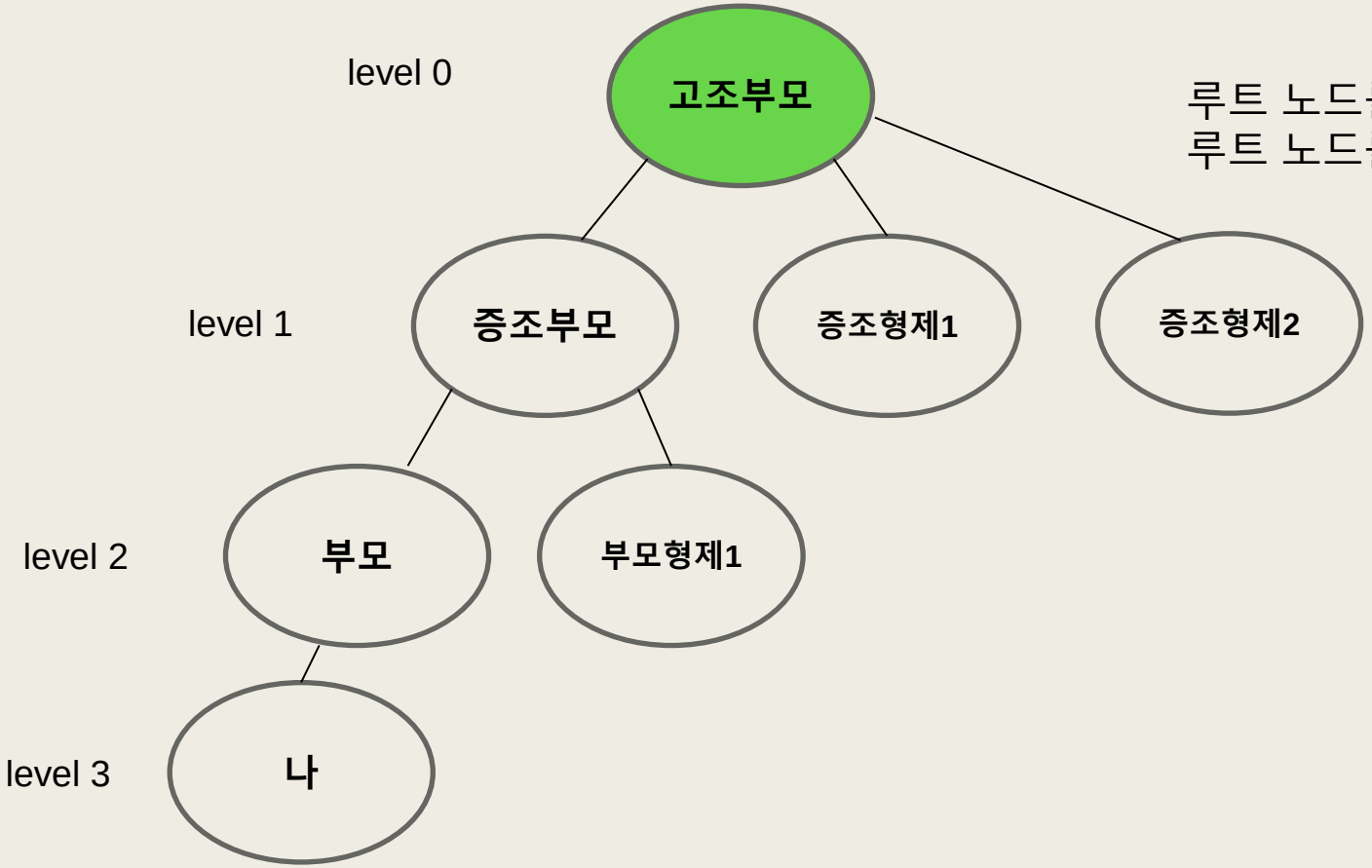
루트 노드 : 가장 상위의 노드 (level 0 or 1)
깊이(depth, or level) : 루트 노드와의 거리
높이 : 가장 깊은 정점의 깊이 or 깊이 + 1

부모 노드 : 연결관계의 두 노드 중 상위 노드
자식 노드 : 연결관계의 두 노드 중 하위 노드
리프 노드 : 자식이 없는 노드

차수(degree) : 자식 노드의 수
서브 트리(subtree) : 자식 노드를 루트로 하는 트리

트리의 특징

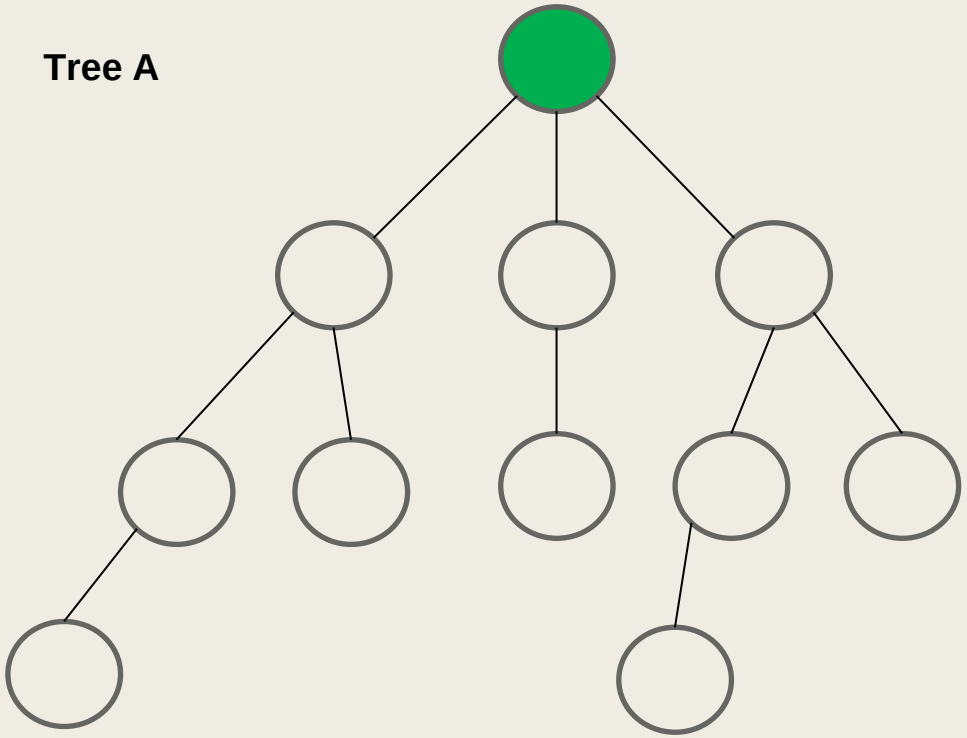
계층 구조를 이루는 자료를 표현하는데 유용



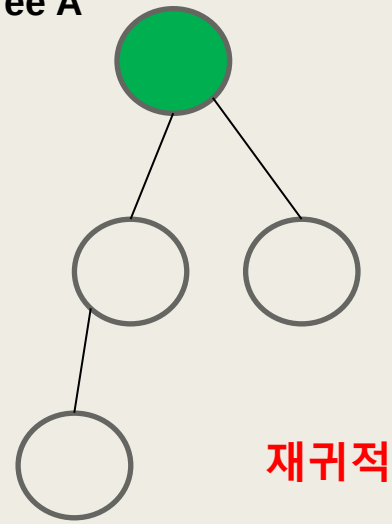
루트 노드를 제외한 모든 노드는 하나의 부모를 갖는다.
루트 노드를 제외한 모든 노드의 수 = N-1 = 간선의 개수

트리의 특징

Tree A



Subtree of Tree A



재귀적 구조를 갖고 있다!

트리의 특징

- 계층 구조를 표현하는데 용이하다.
- 임의의 점 u, v에 대해서 path(u, v) 가 유일하다.
- 루트 노드를 제외한 모든 노드는 하나의 부모를 갖는다.
- 재귀적 구조를 이루고 있다.

트리 구현

- 노드의 값
- 해당 노드의 부모
- 해당 노드의 자식들

```
struct TREENODE {
    int parent = -1;
    vector<int> children;
};

struct TREENODE tree[MAX];
```

트리 구현

- 노드의 값
- 해당 노드의 부모
- 해당 노드의 자식들

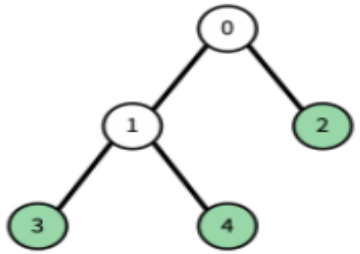
```
for (int i = 0; i < N-1; i++) {
    int par, child;
    cin >> par >> child;
    tree[child].parent = par;
    tree[par].children.push_back(child);
}
```

백준 1068 트리

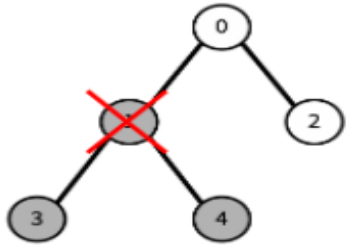
트리에서 리프 노드란, 자식의 개수가 0인 노드를 말한다.

트리가 주어졌을 때, 노드 하나를 지울 것이다. 그 때, 남은 트리에서 리프 노드의 개수를 구하는 프로그램을 작성하시오. 노드를 지우면 그 노드와 노드의 모든 자손이 트리에서 제거된다.

예를 들어, 다음과 같은 트리가 있다고 하자.



현재 리프 노드의 개수는 3개이다. (초록색 색칠된 노드) 이때, 1번을 지우면, 다음과 같이 변한다. 검정색으로 색칠된 노드가 트리에서 제거된 노드이다.



이제 리프 노드의 개수는 1개이다.

입력

첫째 줄에 트리의 노드의 개수 N이 주어진다. N은 50보다 작거나 같은 자연수이다. 둘째 줄에는 0번 노드부터 N-1번 노드까지, 각 노드의 부모가 주어진다. 만약 부모가 없다면 (루트) -1이 주어진다. 셋째 줄에는 지울 노드의 번호가 주어진다.

출력

첫째 줄에 입력으로 주어진 트리에서 입력으로 주어진 노드를 지웠을 때, 리프 노드의 개수를 출력한다.

백준 1068 트리

1. 특정 노드 이하를 제거한 트리의 자식이 0인 노드의 개수를 구한다.
2. 만약 지워진 노드의 형제가 자신밖에 없으면 (부모의 자식의 수가 1이면) 1. – 2. 값 +1 아니면 1. – 2.
3. 그 값을 출력

백준 11725 트리의 부모 찾기

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	256 MB	17939	7565	5625	43.373%

문제

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

입력

첫째 줄에 노드의 개수 N ($2 \le N \le 100,000$)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.

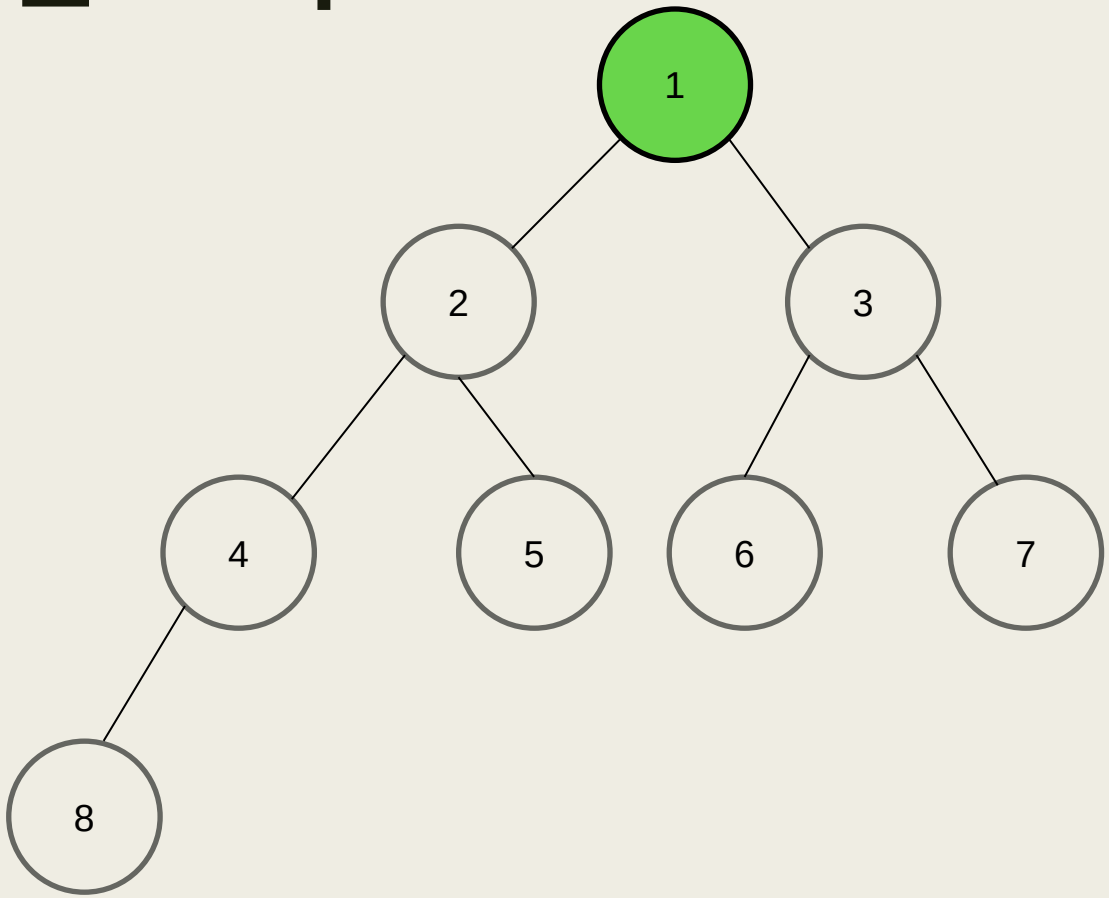
백준 11725 트리의 부모 찾기

1. 1번 노드부터 탐색 시작
2. 만약 인접한 노드가 부모가 아니면 자식 노드
3. 재귀함수를 돌면서 그 부모의 값을 저장

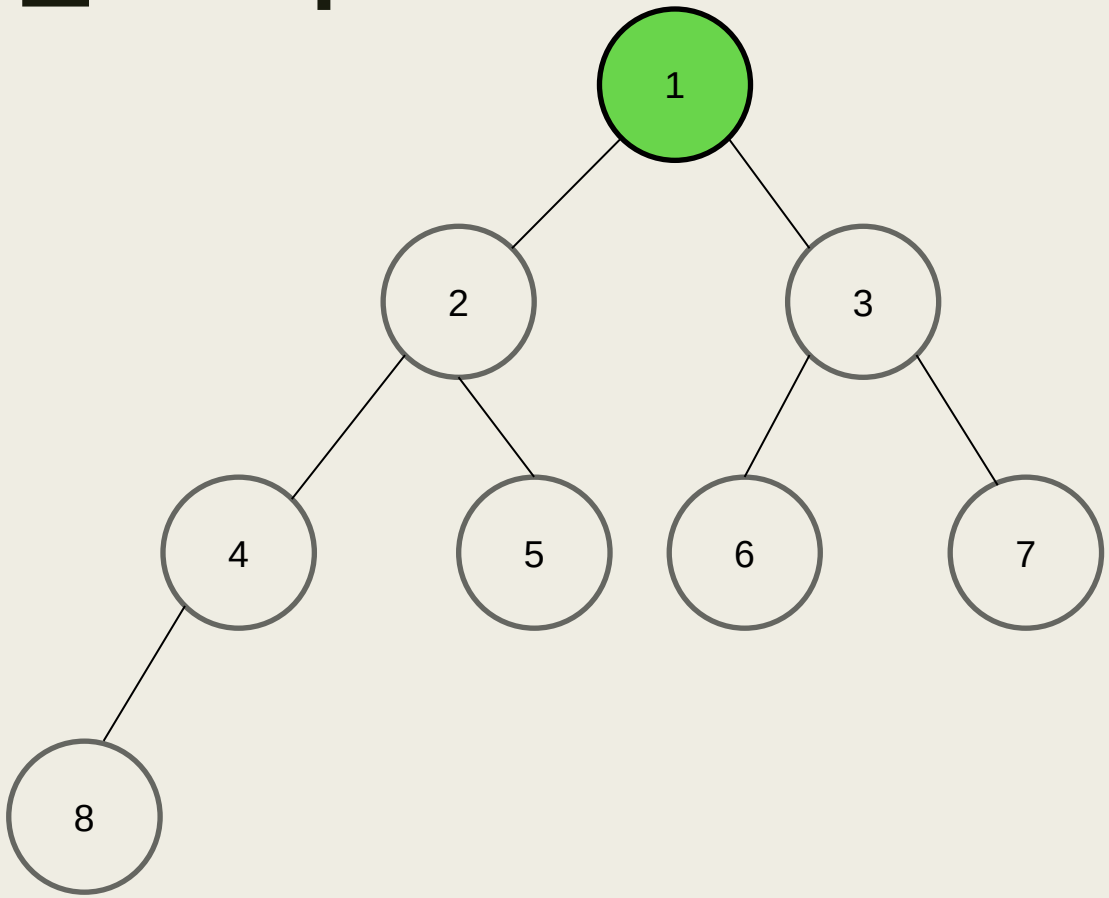
이진 트리

자식 노드의 수가 최대 2개인 트리 구조

이진 트리

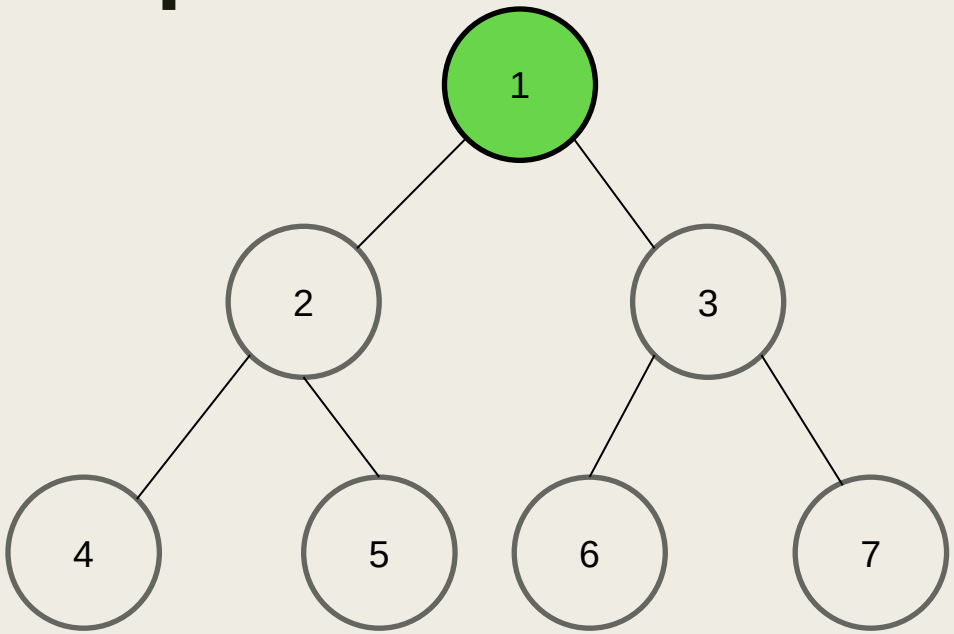


이진 트리



완전 이진 트리

이진 트리



포화 이진 트리

이진 트리 구현

- 노드의 값
- 해당 노드의 부모
- 해당 노드의 왼쪽 자식, 오른쪽 자식

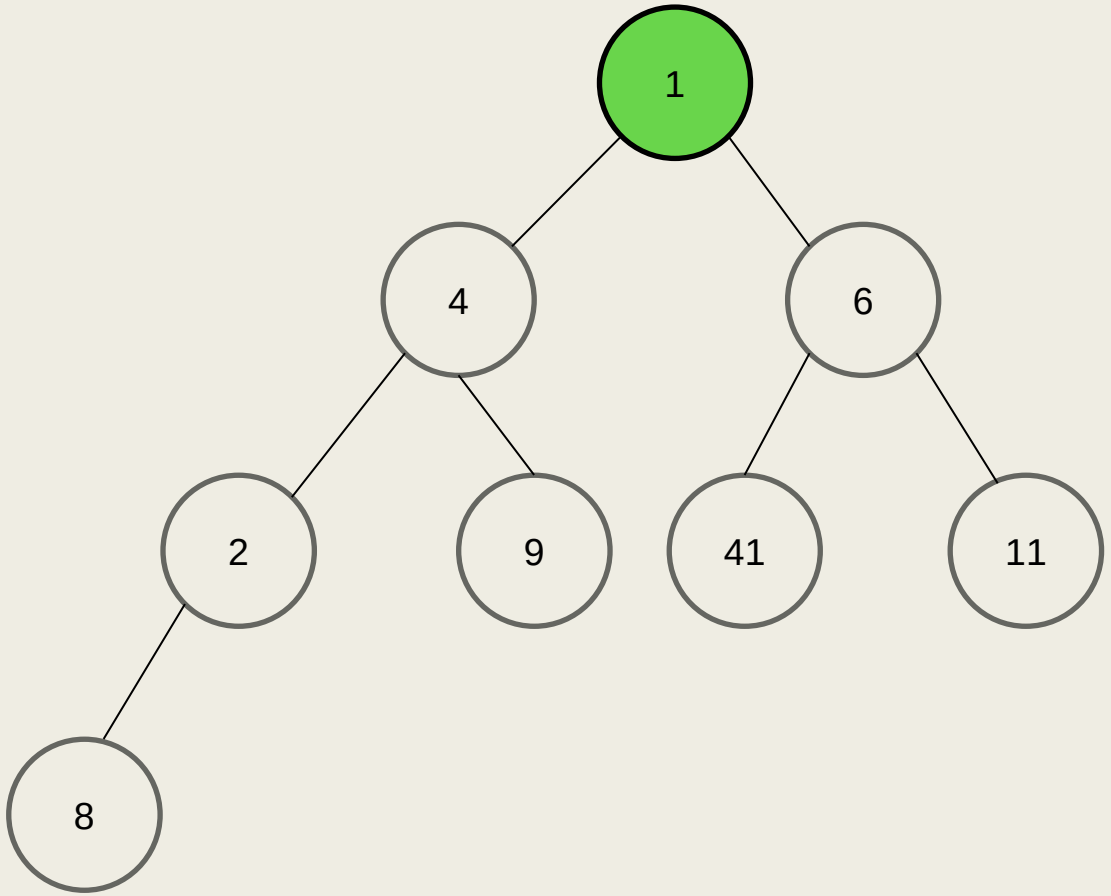
```
struct TREENODE {
    int parent = -1;
    int left_child = -1, right_child = -1;
};

struct TREENODE tree[MAX];
```

이진 트리 구현

- 노드의 값
- 해당 노드의 부모
- 해당 노드의 왼쪽 자식, 오른쪽 자식

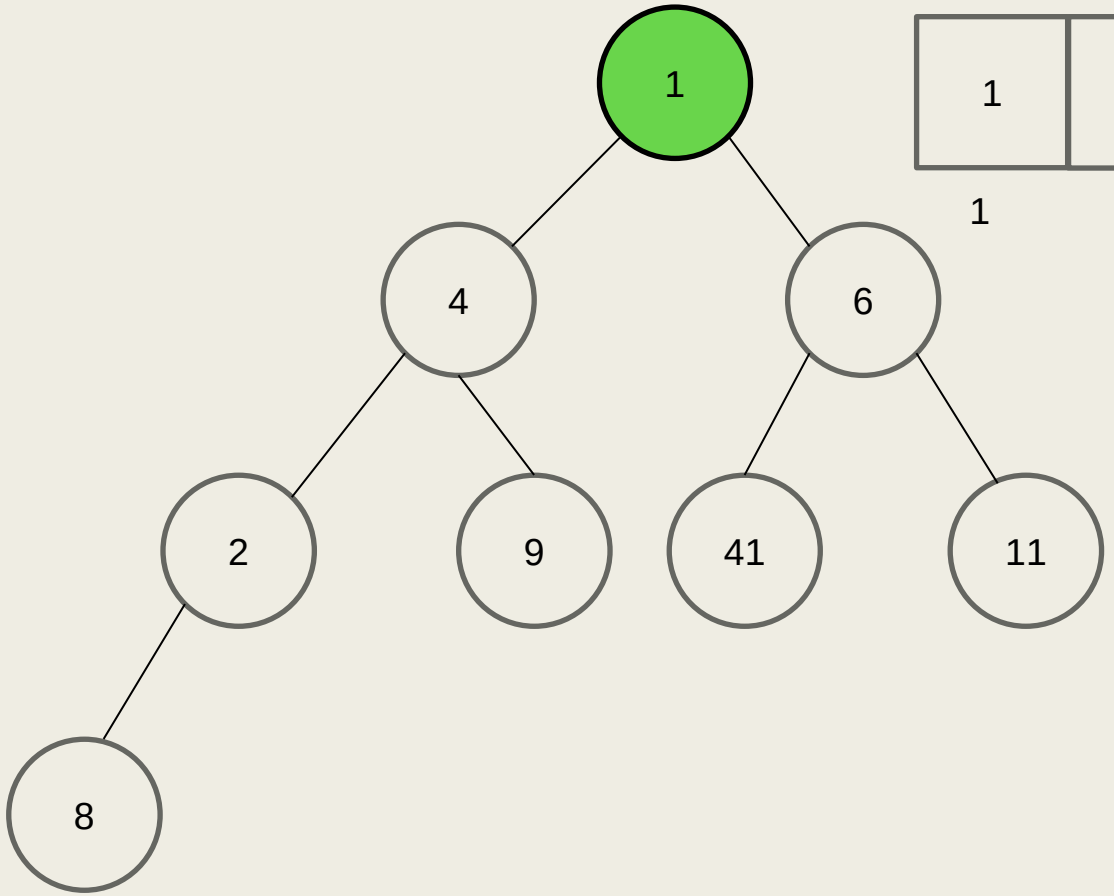
```
for (int i = 0; i < N-1; i++) {
    int par, child;
    cin >> par >> child;
    tree[child].parent = par;
    if (tree[par].left_child == -1)tree[par].left_child = child;
    else tree[par].right_child = child;
}
```

이진 트리 구현(배열)



따라서 배열로 구현이 가능하다!

이진 트리 구현(배열)

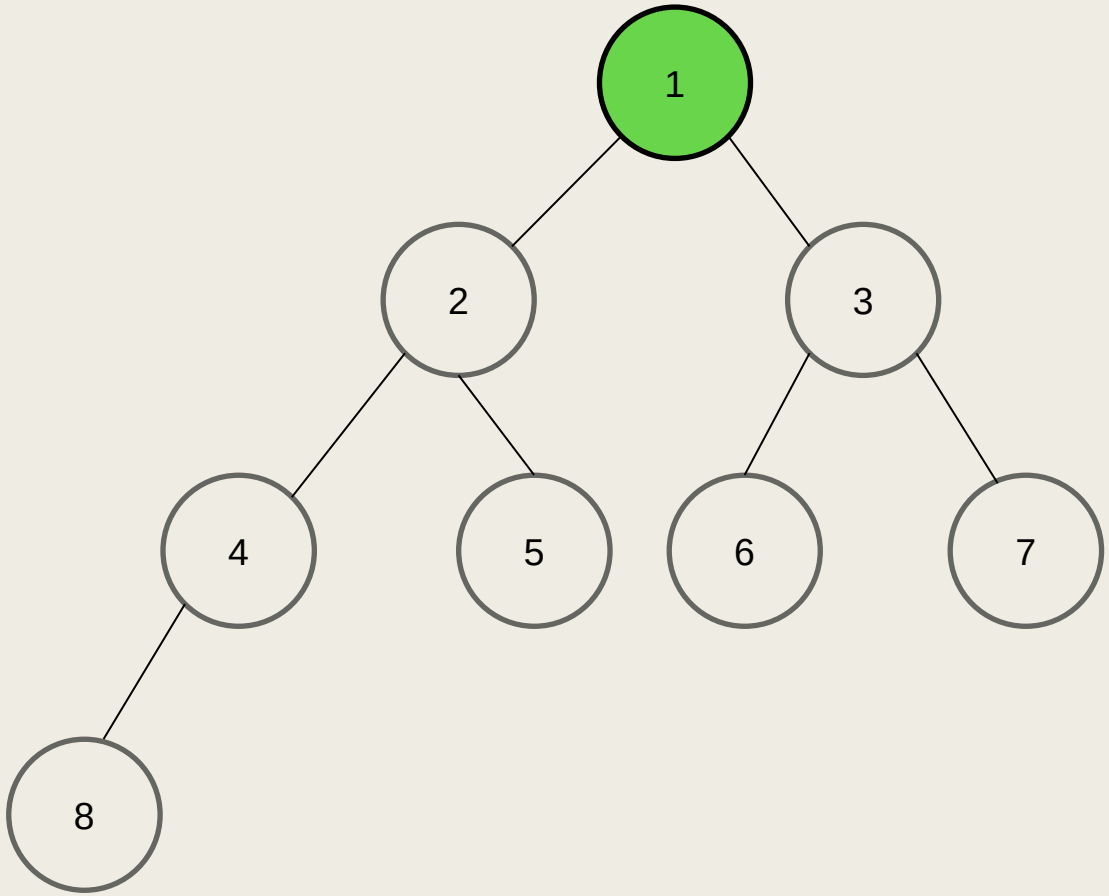


1	4	6	2	9	41	11	8
1	2	3	4	5	6	7	8

트리의 순회

- 전위 순회 : 자기 자신을 방문 후에 자식들을 방문
- 중위 순회 : 왼쪽 자식 방문 -> 자신 방문 -> 오른쪽 자식 방문
- 후위 순회 : 자식들을 방문 후에 자신 방문

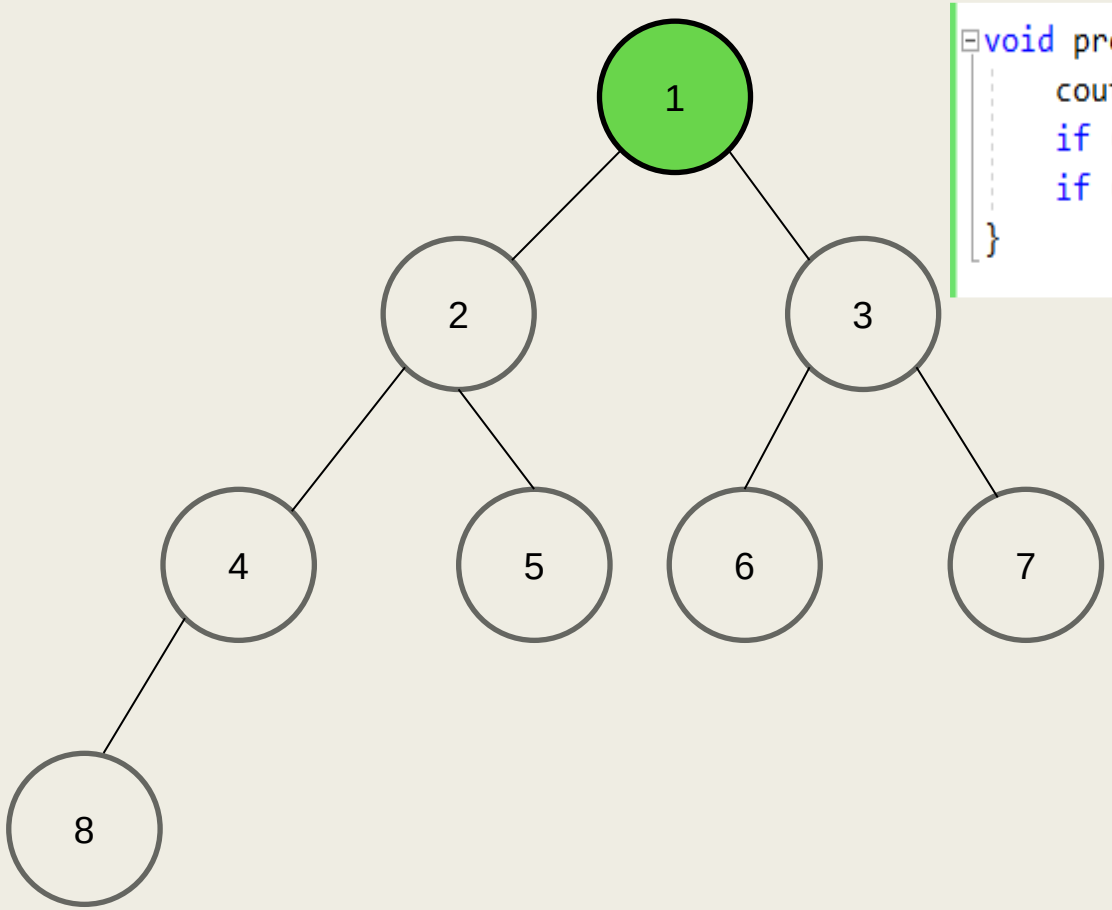
트리의 순회(전위 순회)



전위 순회 (VLR)

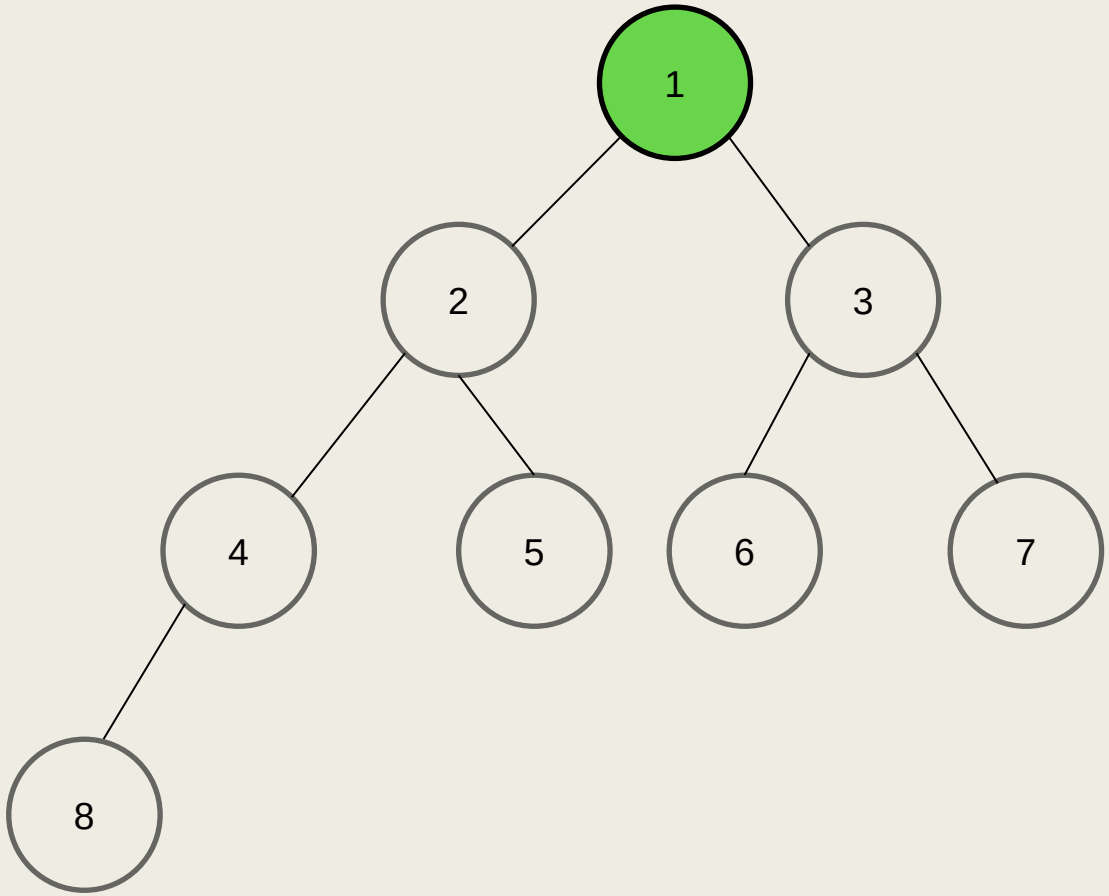
1 -> 2 -> 4 -> 8 -> 5 - > 3 -> 6 -> 7

트리의 순회(전위 순회)




```
void pre_order(int root) {
    cout << root << " ";
    if (tree[root].left_child != -1)pre_order(tree[root].left_child);
    if (tree[root].right_child != -1)pre_order(tree[root].right_child);
}
```

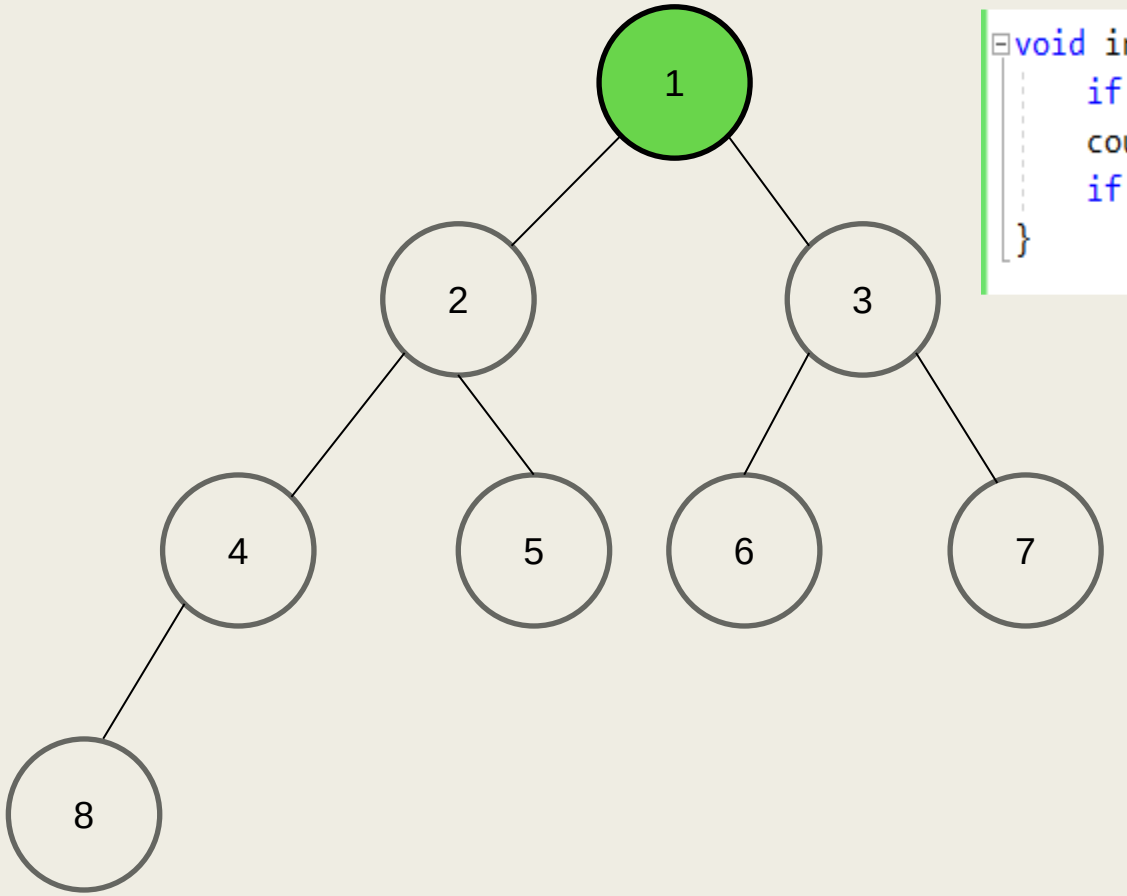
트리의 순회(중위 순회)



중위 순회 (LVR)

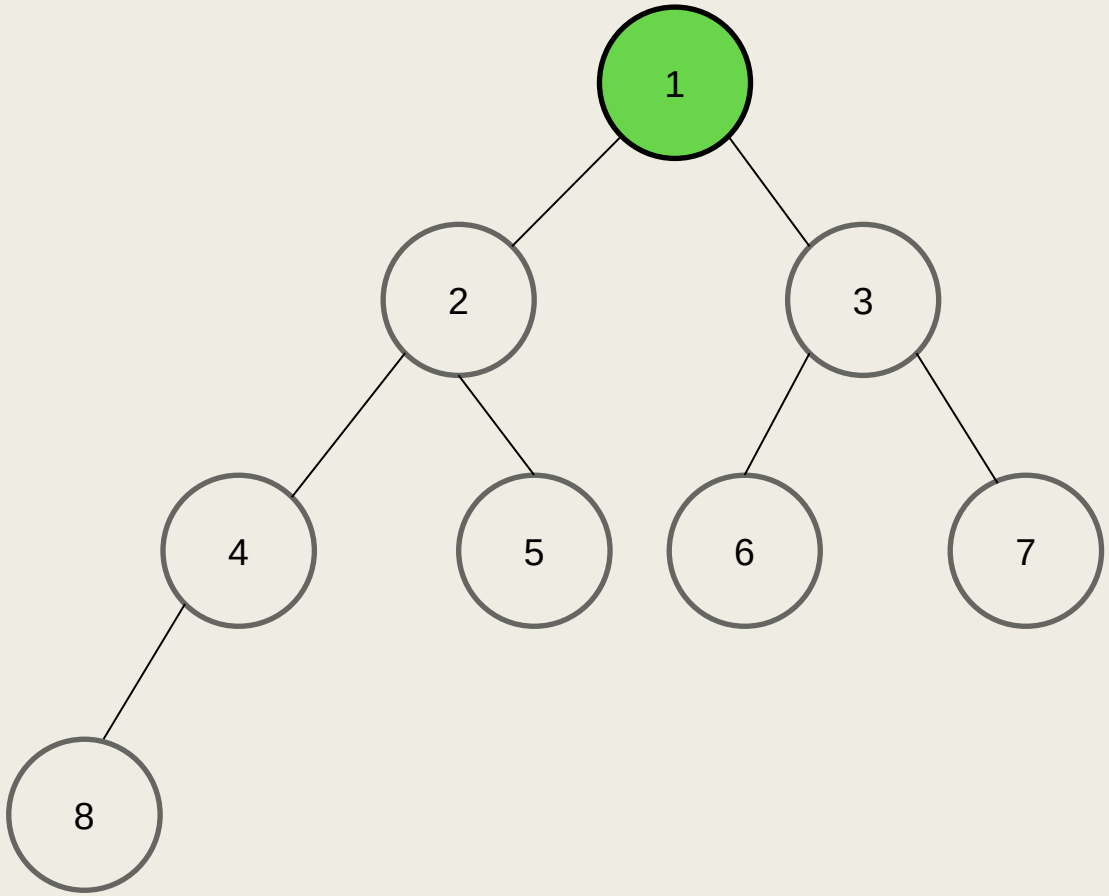
8 -> 4 -> 2 -> 5 -> 1 -> 6 -> 3 -> 7

트리의 순회(중위 순회)




```
void in_order(int root) {
    if (tree[root].left_child != -1)in_order(tree[root].left_child);
    cout << root << " ";
    if (tree[root].right_child != -1)in_order(tree[root].right_child);
}
```

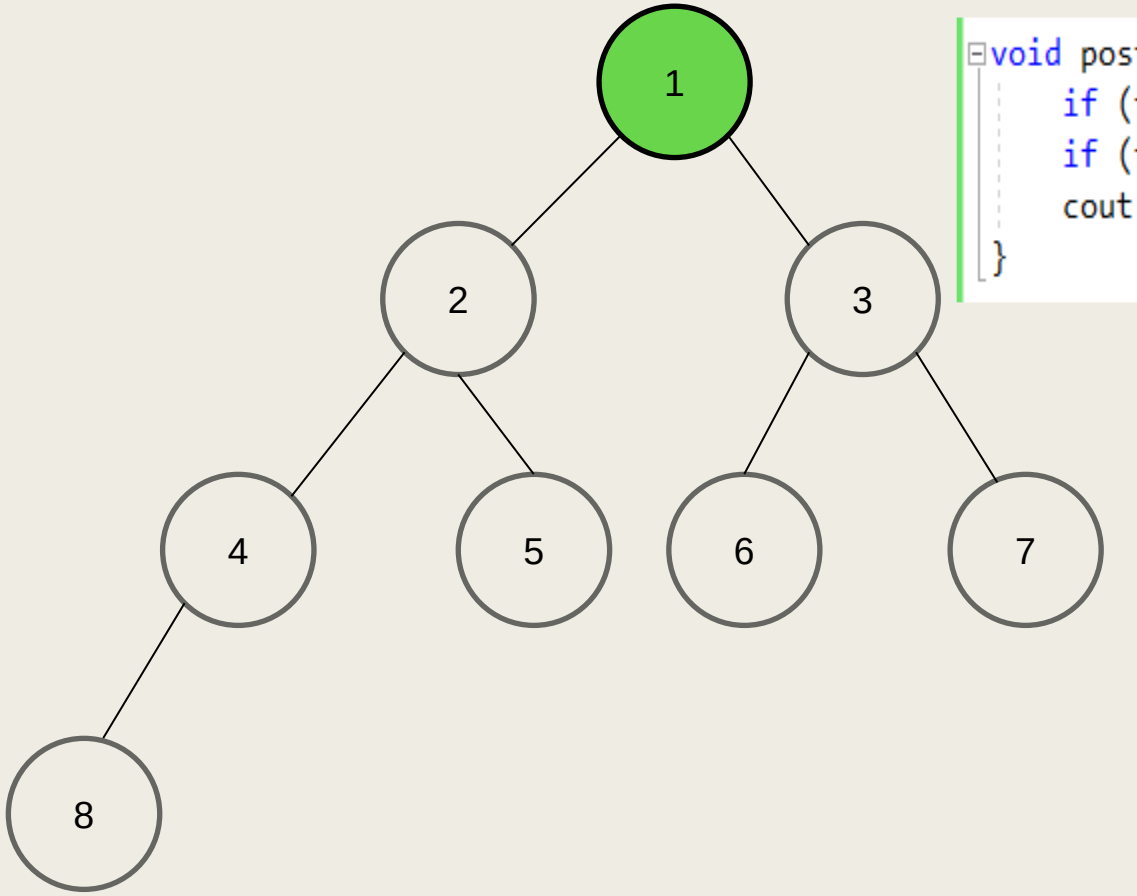
트리의 순회(후위 순회)



후위 순회 (LRV)

8 -> 4 -> 5 -> 2 -> 6 -> 7 -> 3 -> 1

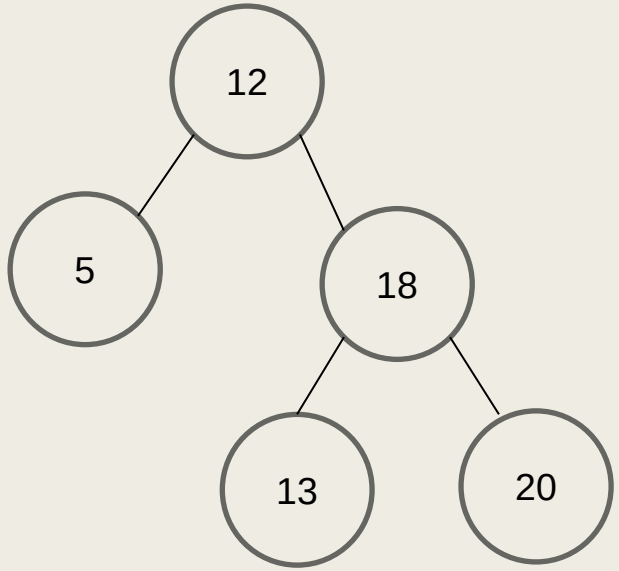
트리의 순회(후위 순회)




```
void post_order(int root) {
    if (tree[root].left_child != -1)post_order(tree[root].left_child);
    if (tree[root].right_child != -1)post_order(tree[root].right_child);
    cout << root << " ";
}
```

이진 탐색 트리(binary search tree)

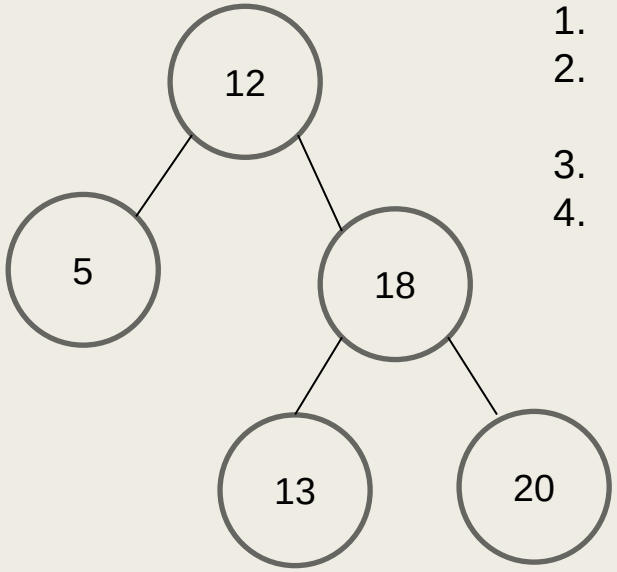
- 왼쪽 서브 트리의 키들은 루트의 키보다 작아야 한다.
- 오른쪽 서브 트리의 키들은 루트의 키보다 커야 한다.
- 오른쪽 서브 트리와 왼쪽 서브 트리 모두 이진 탐색 트리이다.



중위 순회를 하게 되면

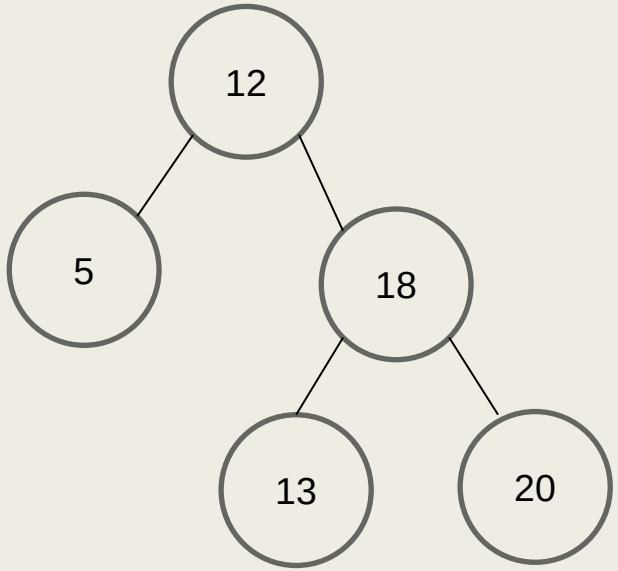
5 -> 12 -> 13 -> 18 -> 20

이진 탐색 트리 (탐색)



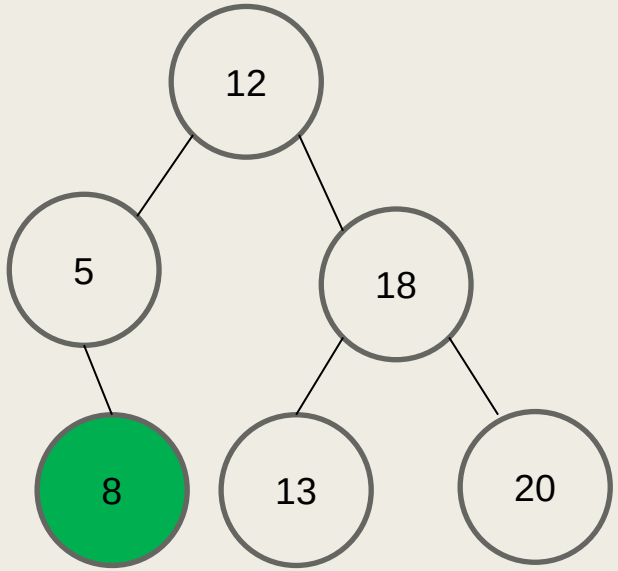
1. 루트부터 시작
2. 노드의 key값과 찾으려는 key값을 비교, 찾으려는 값이 더 크면 우측 서브 트리로 진행 더 작으면 좌측 서브 트리로 진행
3. 노드의 key값과 찾으려는 key값이 같으면 탐색 성공
4. 만약 2.에서 이동해야 하는 서브 트리가 존재하지 않으면 탐색 실패

이진 탐색 트리 (삽입)



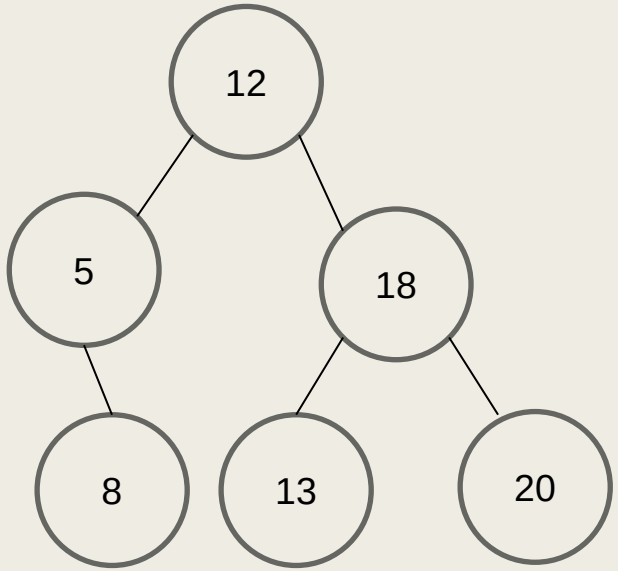
8을 삽입해보자

이진 탐색 트리 (삽입)



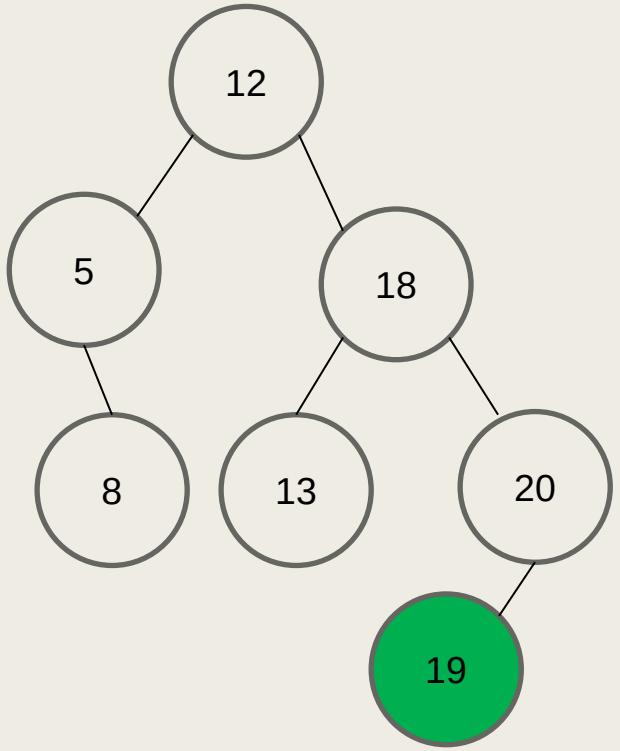
8을 삽입해보자

이진 탐색 트리 (삽입)



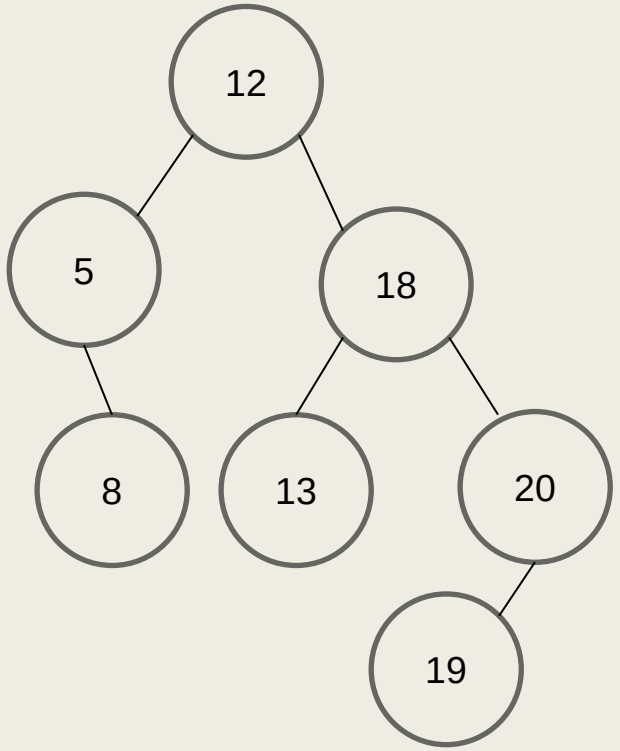
19를 삽입해보자

이진 탐색 트리 (삽입)



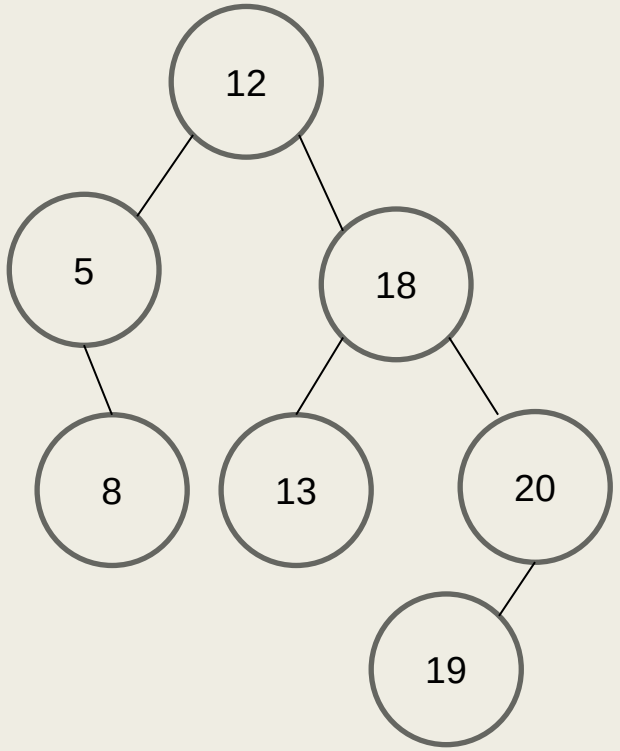
19를 삽입해보자

이진 탐색 트리 (삽입)

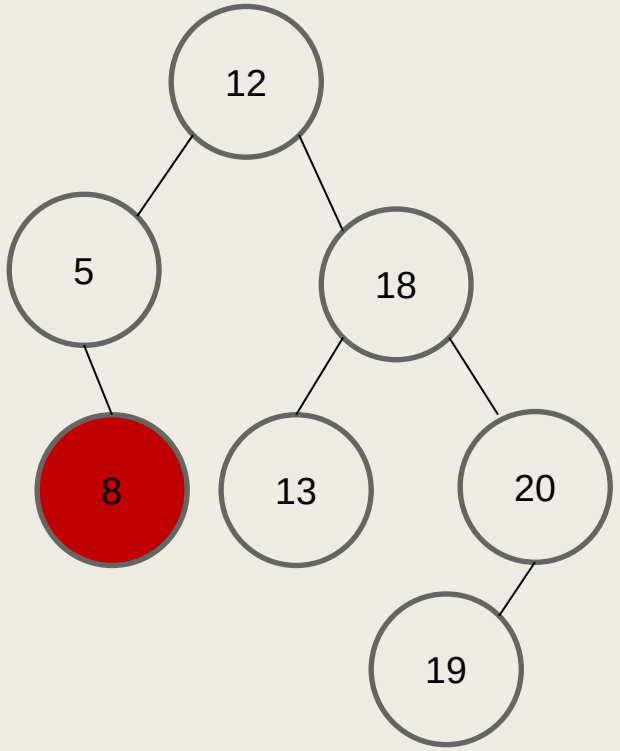


탐색에서 실패한 곳에 삽입을 한다.

이진 탐색 트리 (삭제)

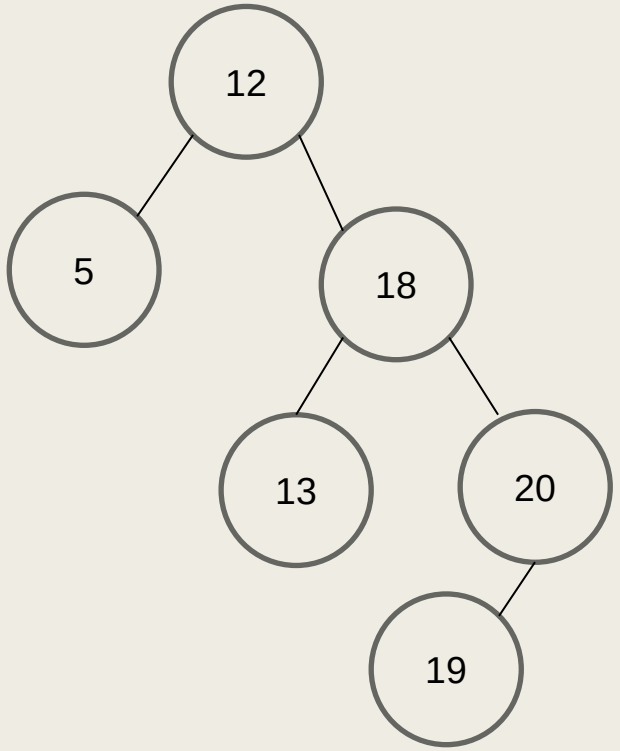


이진 탐색 트리 (삭제)



8을 삭제해보자

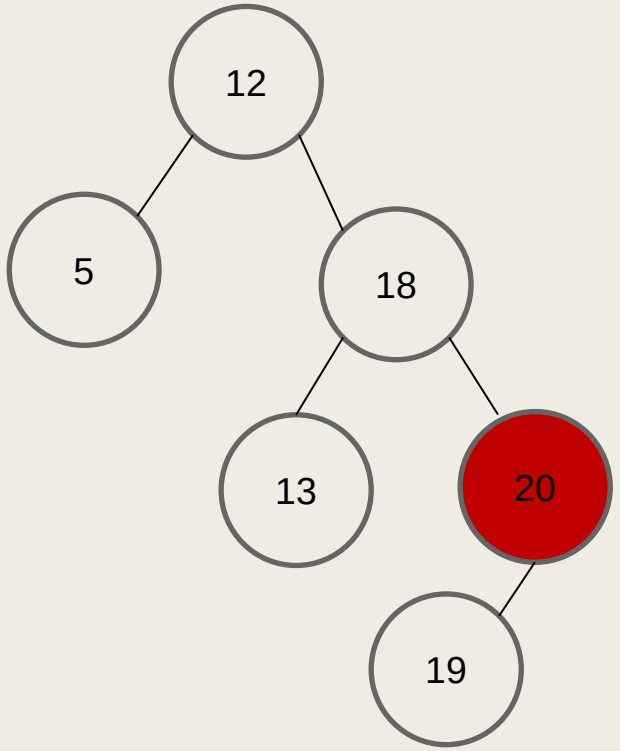
이진 탐색 트리 (삭제)



8을 삭제해보자

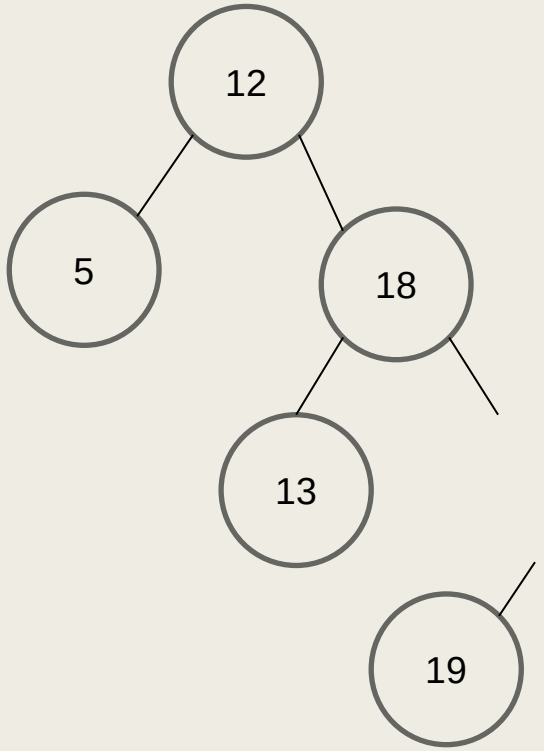
리프 노드를 삭제하는 경우에는 부모 노드가 가리키던 값을 -1로 변경

이진 탐색 트리 (삭제)



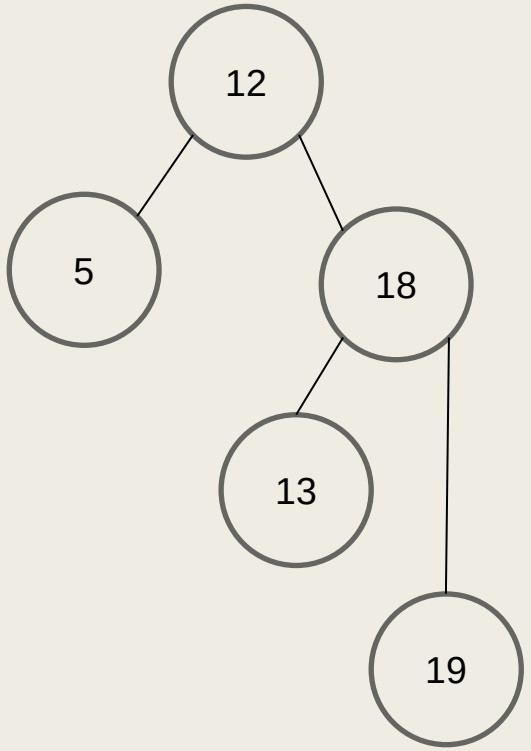
20을 삭제해보자

이진 탐색 트리 (삭제)



20을 삭제해보자

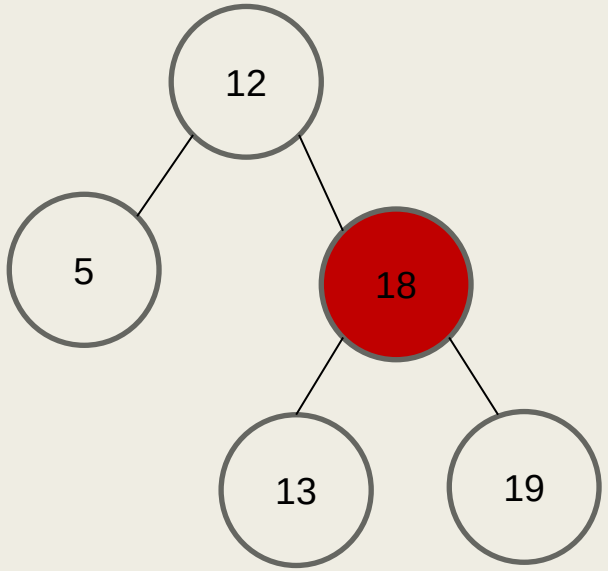
이진 탐색 트리 (삭제)



20을 삭제해보자

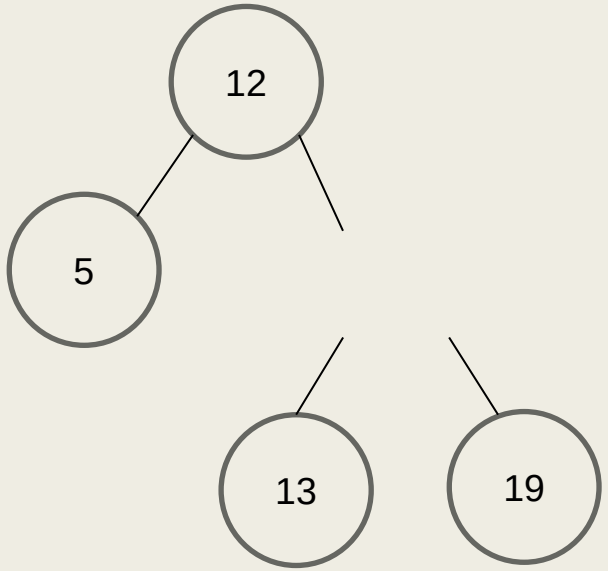
하나의 서브 트리만 존재하는 경우 부모 노드가 그 서브 트리의 루트를 가리키게 바꾼다.

이진 탐색 트리 (삭제)



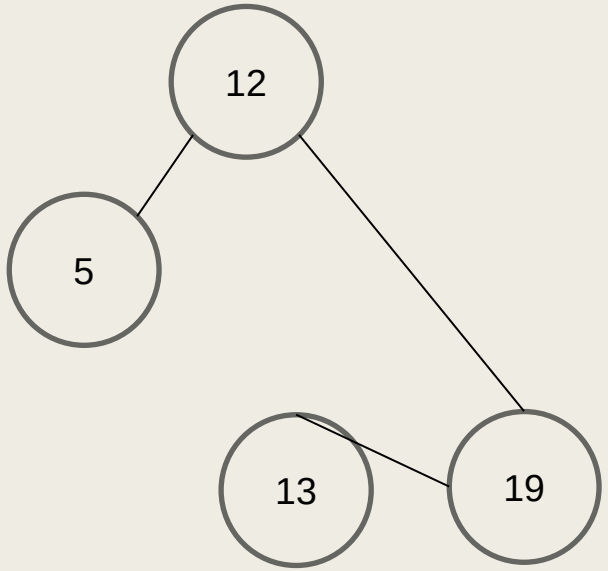
18을 삭제해보자

이진 탐색 트리 (삭제)



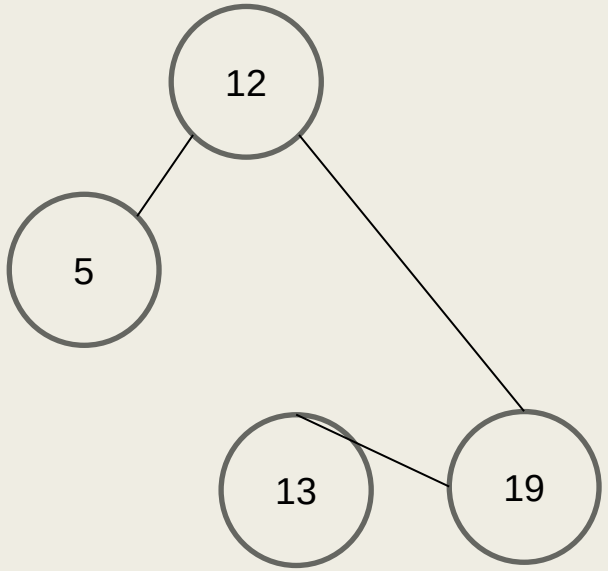
18을 삭제해보자

이진 탐색 트리 (삭제)



18을 삭제해보자

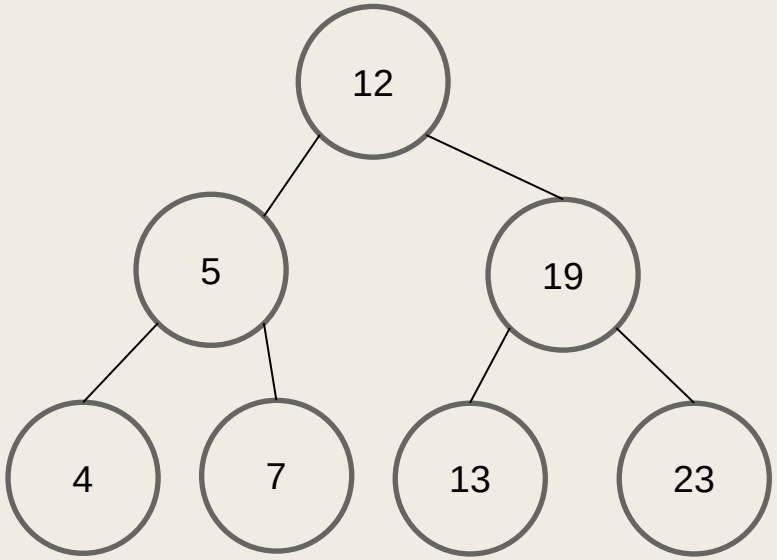
이진 탐색 트리 (삭제)



18을 삭제해보자

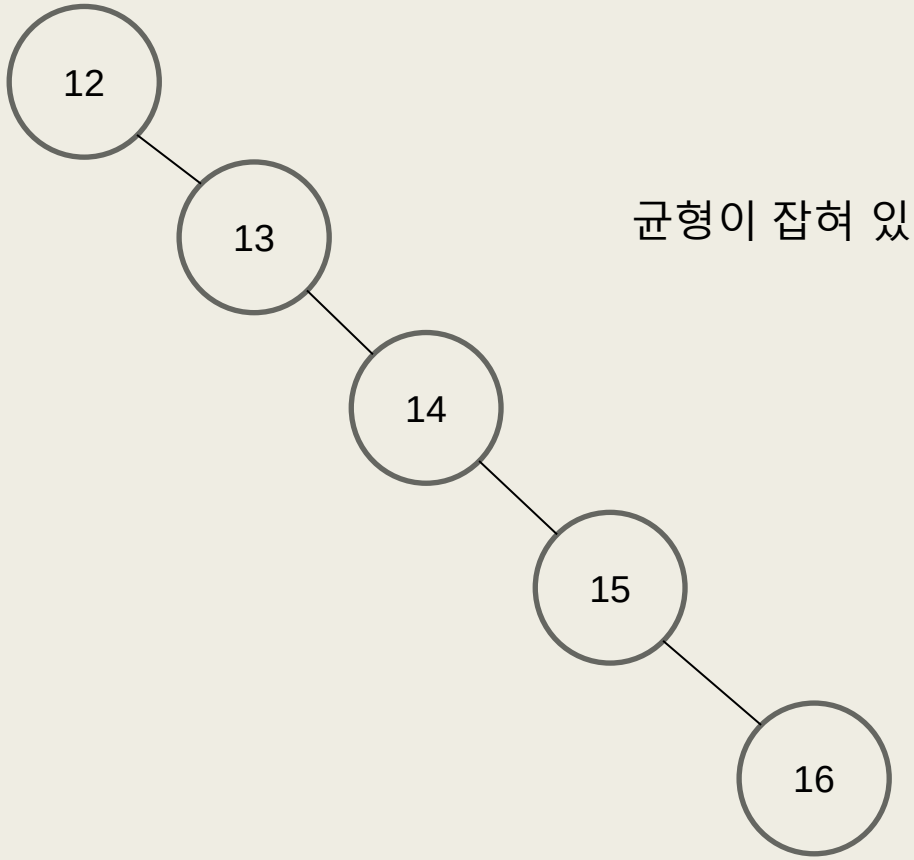
삭제된 노드의 부모 노드가 왼쪽 서브 트리의 최대값을
가리키거나 오른쪽 서브 트리의 최소값을 가리키도록 한다.

시간 복잡도



균형이 잡혀 있는 트리라면 O(logN)

시간 복잡도



균형이 잡혀 있지 않는 트리라면 O(N)

트리의 지름

가장 먼 두 정점 사이의 거리 혹은 가장 먼 두 정점을 연결하는 경로

트리의 지름

- 트리에서 임의의 정점 x를 잡는다.
- 정점 x에서의 가장 먼 정점 y를 찾는다.
- 정점 y에서 가장 먼 정점 z를 찾는다.

y-z 의 경로가 트리의 지름

트리의 지름(증명)

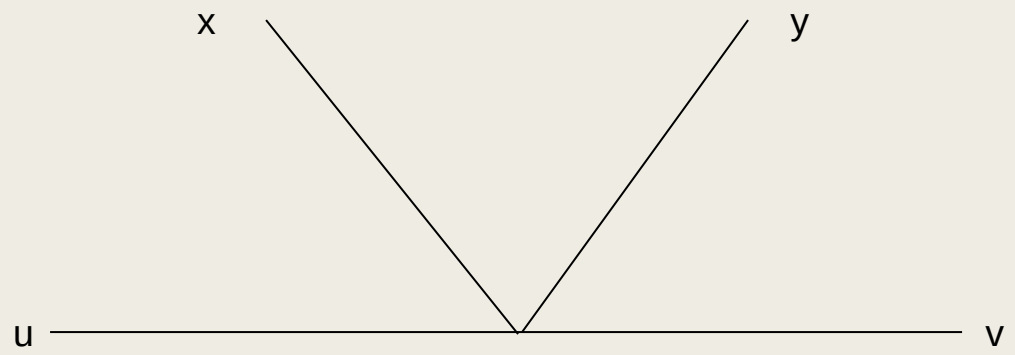
u-v 가 실제 트리의 지름이라고 가정해보자

1. x가 u 혹은 v 인 경우
2. y가 u 혹은 v 인 경우
3. x, y, u, v가 모두 다른 경우

트리의 지름(증명)

1. x-y를 연결하는 경로가 u-v의 경로와 하나의 점 이상 공유하는 경우
2. x-y를 연결하는 경로가 u-v의 경로와 독립인 경우

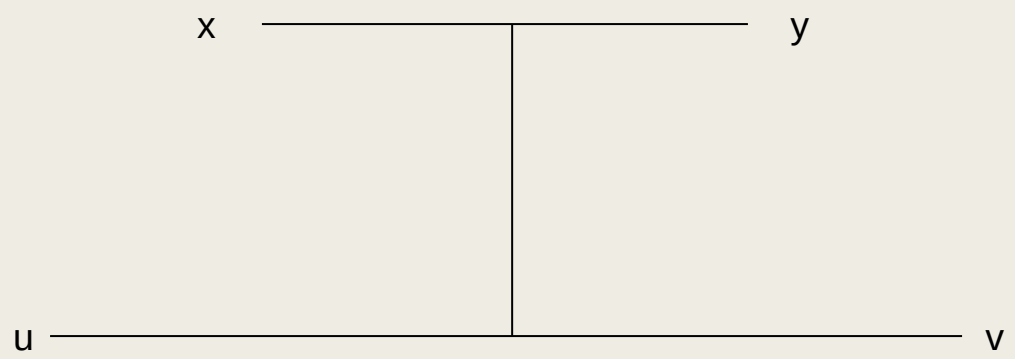
1. case



트리의 지름(증명)

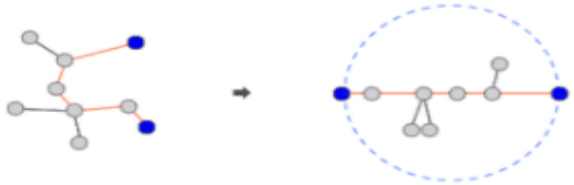
1. x-y를 연결하는 경로가 u-v의 경로와 하나의 점 이상 공유하는 경우
2. x-y를 연결하는 경로가 u-v의 경로와 독립인 경우

2. case



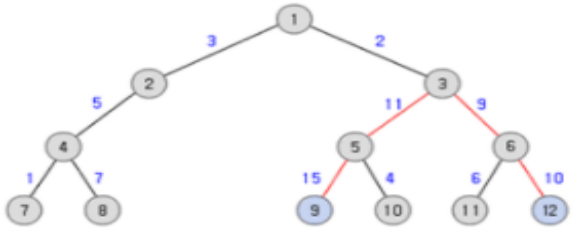
백준 1967 트리의 지름

트리(tree)는 사이클이 없는 무방향 그래프이다. 트리에서는 어떤 두 노드를 선택해도 둘 사이에 경로가 항상 하나만 존재하게 된다. 트리에서 어떤 두 노드를 선택해서 양쪽으로 쫙 당길 때, 가장 길게 늘어나는 경우가 있을 것이다. 이럴 때 트리의 모든 노드들은 이 두 노드를 지름의 끝 점으로 하는 원 안에 들어가게 된다.



이런 두 노드 사이의 경로의 길이를 트리의 지름이라고 한다. 정확히 정의하자면 트리에 존재하는 모든 경로들 중에서 가장 긴 것의 길이를 말한다.

입력으로 루트가 있는 트리를 가중치가 있는 간선들로 줄 때, 트리의 지름을 구해서 출력하는 프로그램을 작성하시오. 아래와 같은 트리가 주어진다면 트리의 지름은 45가 된다.



트리의 노드는 1부터 n까지 번호가 매겨져 있다.

입력

파일의 첫 번째 줄은 노드의 개수 n(1 ≤ n ≤ 10,000)이다. 둘째 줄부터 n-1개의 줄에 각 간선에 대한 정보가 들어온다. 간선에 대한 정보는 세 개의 정수로 이루어져 있다. 첫 번째 정수는 간선이 연결하는 두 노드 중 부모 노드의 번호를 나타내고, 두 번째 정수는 자식 노드를, 세 번째 정수는 간선의 가중치를 나타낸다. 간선에 대한 정보는 부모 노드의 번호가 작은 것이 먼저 입력되고, 부모 노드의 번호가 같으면 자식 노드의 번호가 작은 것이 먼저 입력된다. 루트 노드의 번호는 항상 1이라고 가정하며, 간선의 가중치는 100보다 크지 않은 양의 정수이다.

출력

첫째 줄에 트리의 지름을 출력한다.

백준 1967 트리의 지름

백준 1967 트리의 지름

```
void solve(int w, int cur, int prev = -1) {
    if (ans < w) {
        ans = w;
        x = cur;
    }
    for (int i = 0; i < tree[cur].size(); i++) {
        int next = tree[cur][i].first;
        int d = tree[cur][i].second;
        if (next == prev) continue;
        solve(w + d, next, cur);
    }
}
```

필수문제

- A - 트리
- B - 트리의 부모 찾기
- C - 트리 순회
- D - 이진 검색 트리
- E - 트리의 지름

연습문제

- A - 트리의 순회
- B - 트리
- C - 트리
- D - 양 구출 작전
- E - 트리의 높이와 너비
- F - Labyrinth
- G - 사회망 서비스 (SNS)

피드백 및 질의응답

감사합니다!