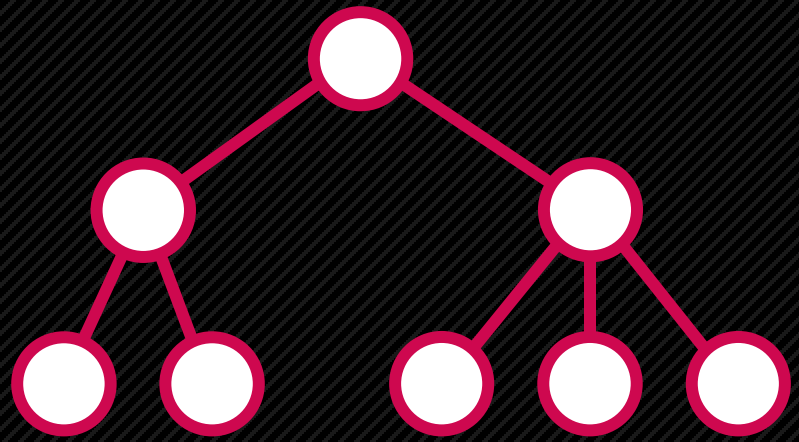
알고리즘 특강
트리

그래프의 특수한 형태로, 정확히 N개의 정점과 N-1개의 간선으로 이루어진 구조입니다.
트리만의 특수한 구조를 이해하고 접근해야 합니다.

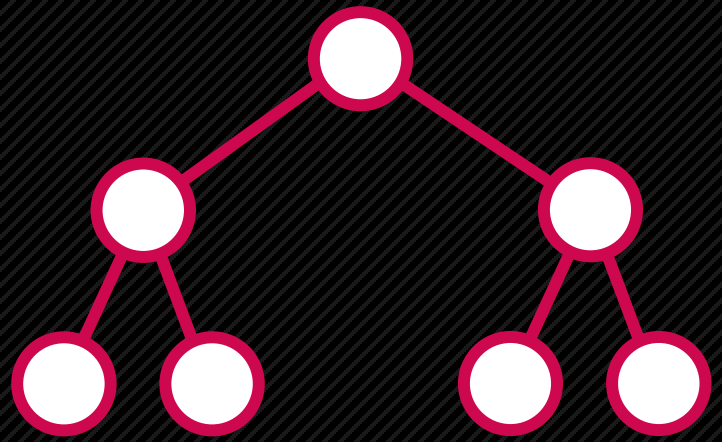
트리



Tree

- 그래프의 특수한 형태로, 계층 구조가 있으며, 단 1개의 간선만 존재하는 자료구조.
- 루트 노드의 부모는 0개이며, 자식 노드의 부모는 1개이다.
- 루트 노드는 0개 이상의 자식 노드를 갖고 있으며, 자식 노드 또한 0개 이상의 자식 노드를 가지고 있다.
- 사이클을 갖고 있지 않다.

이진 트리



Binary Tree

- 트리에서, 자식 노드의 개수가 0개 이상 2개 이하로 제한된 트리.

갑자기 문제??

Silver 3 - 상근이의 여행 (#9372)

요약

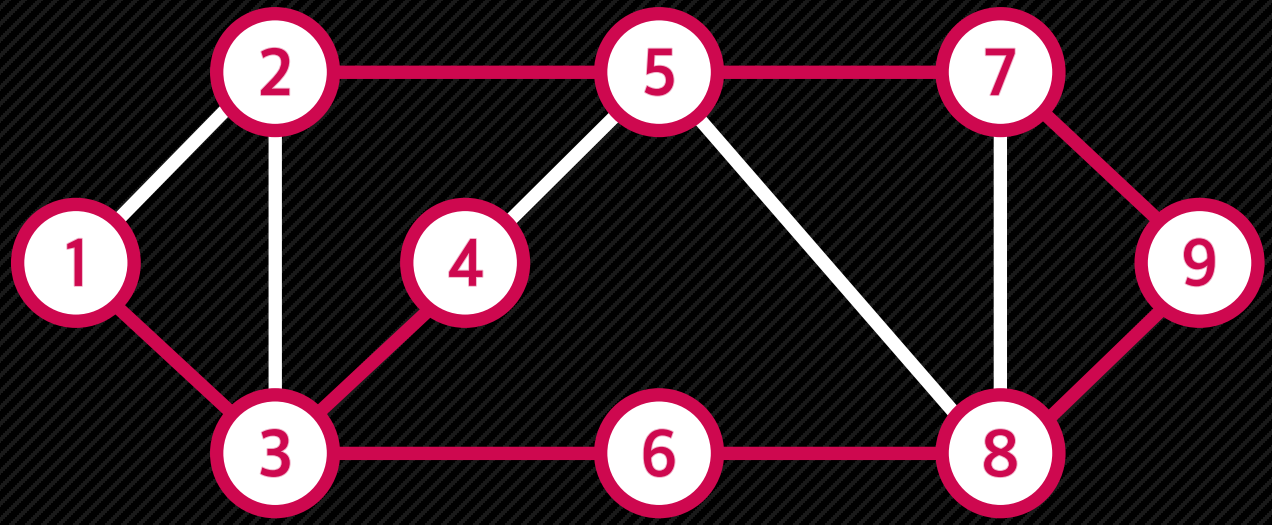
- 주어진 항공 루트를 이용하여 여행을 할 때, 가장 적은 종류의 루트를 통해 N개의 국가를 둘러보려고 한다.
- 가장 적은 루트의 수를 구하는 프로그램을 작성하시오.

제약조건

- 국가의 수 N은 2 <= N <= 1,000 이다.
- 모든 국가는 연결 그래프를 이루고 있음이 보장된다.

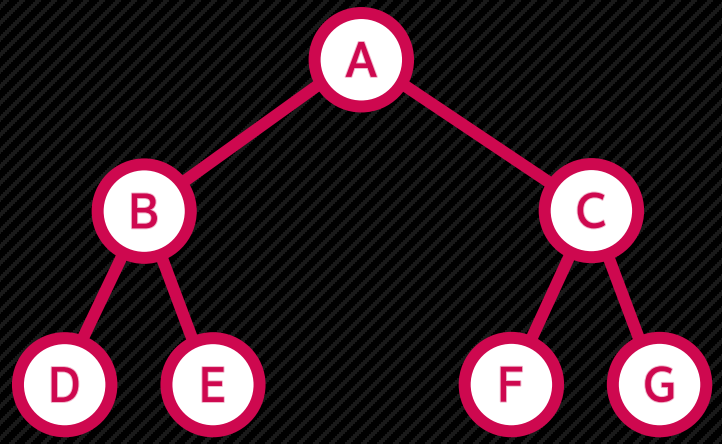
사실...

Silver 3 - 상근이의 여행 (#9372)



Spanning Tree → 무조건 N - 1개!

트리의 탐색



- **전위 순회 (Preorder Traversal)**

부모 → 왼쪽 자식 → 오른쪽 자식 순으로 탐색

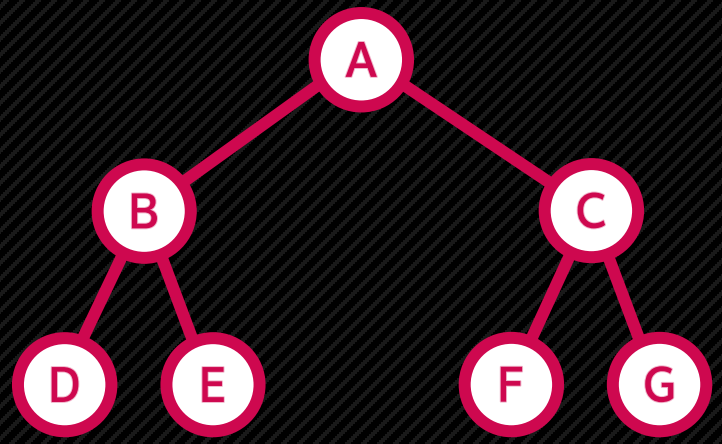
- **중위 순회 (Inorder Traversal)**

왼쪽 자식 → 부모 → 오른쪽 자식 순으로 탐색

- **후위 순회 (Postorder Traversal)**

왼쪽 자식 → 오른쪽 자식 → 부모 순으로 탐색

트리의 탐색



- **전위 순회 (Preorder Traversal)**

부모 → 왼쪽 자식 → 오른쪽 자식 순으로 탐색

A - B - D - E - C - F - G

- **중위 순회 (Inorder Traversal)**

왼쪽 자식 → 부모 → 오른쪽 자식 순으로 탐색

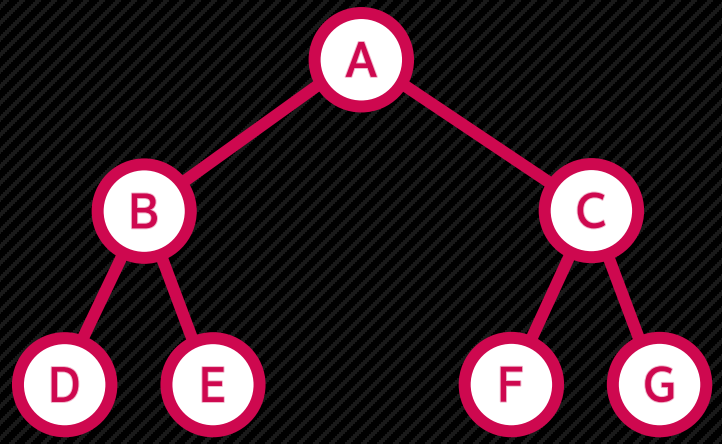
D - B - E - A - F - C - G

- **후위 순회 (Postorder Traversal)**

왼쪽 자식 → 오른쪽 자식 → 부모 순으로 탐색

D - E - B - F - G - C - A

트리의 탐색



전위 순회 (Preorder Traversal)

```
function PreOrder(node)
    visit(node)
    PreOrder(node.left)
    PreOrder(node.right)
```

중위 순회 (Inorder Traversal)

```
function InOrder(node)
    InOrder(node.left)
    visit(node)
    InOrder(node.right)
```

후위 순회 (Postorder Traversal)

```
function PostOrder(node)
    PostOrder(node.left)
    PostOrder(node.right)
    visit(node)
```

연습해보자!

Silver 1 - 트리 순회 (#1991)

요약

- 이진 트리를 입력 받아 전위, 중위, 후위 순회한 결과를 출력하는 프로그램을 작성하시오.

제약조건

- 정점의 수 N은 1 <= N <= 26 이다.

이건 뭐람?

Gold 3 - 트리의 순회 (#2263)

요약

- 이진 트리의 중위 순회와 후위 순위의 결과가 주어질 때, 전위 순회한 결과를 출력하는 프로그램을 작성하시오.

제약조건

- 정점의 수 N은 1 <= N <= 100,000 이다.

뜯어보자.

Gold 3 - 트리의 순회 (#2263)

- 전위 순회 (Preorder Traversal)

```
function PreOrder(node)
    visit(node)
    PreOrder(node.left)
    PreOrder(node.right)
```

- 중위 순회 (Inorder Traversal)

```
function InOrder(node)
    InOrder(node.left)
    visit(node)
    InOrder(node.right)
```

- 후위 순회 (Postorder Traversal)

```
function PostOrder(node)
    PostOrder(node.left)
    PostOrder(node.right)
    visit(node)
```

뜯어보자.

Gold 3 - 트리의 순회 (#2263)

- 전위 순회 (Preorder Traversal)

```
function PreOrder(node)
    visit(node)
    PreOrder(node.left)
    PreOrder(node.right)
```

- 중위 순회 (Inorder Traversal)

```
function InOrder(node)
    InOrder(node.left)
    visit(node)
    InOrder(node.right)
```

D - B - E - A - F - C - G

- 후위 순회 (Postorder Traversal)

```
function PostOrder(node)
    PostOrder(node.left)
    PostOrder(node.right)
    visit(node)
```

D - E - B - F - G - C - A

뜯어보자.

Gold 3 - 트리의 순회 (#2263)

- 전위 순회 (Preorder Traversal)

```
function PreOrder(node)
    visit(node)
    PreOrder(node.left)
    PreOrder(node.right)
```

- 중위 순회 (Inorder Traversal)

```
function InOrder(node)
    InOrder(node.left)
    visit(node)
    InOrder(node.right)
```

D - B - E - A - F - C - G

- 후위 순회 (Postorder Traversal)

```
function PostOrder(node)
    PostOrder(node.left)
    PostOrder(node.right)
    visit(node)
```

D - E - B - F - G - C - A

뜯어보자.

Gold 3 - 트리의 순회 (#2263)

- 전위 순회 (Preorder Traversal)

```
function PreOrder(node)
    visit(node)
    PreOrder(node.left)
    PreOrder(node.right)
```

- 중위 순회 (Inorder Traversal)

```
function InOrder(node)
    InOrder(node.left)
    visit(node)
    InOrder(node.right)
```

D - B - E - A - F - C - G

- 후위 순회 (Postorder Traversal)

```
function PostOrder(node)
    PostOrder(node.left)
    PostOrder(node.right)
    visit(node)
```

D - E - B - F - G - C - A

뜯어보자.

Gold 3 - 트리의 순회 (#2263)

- 중위 순회 (Inorder Traversal)

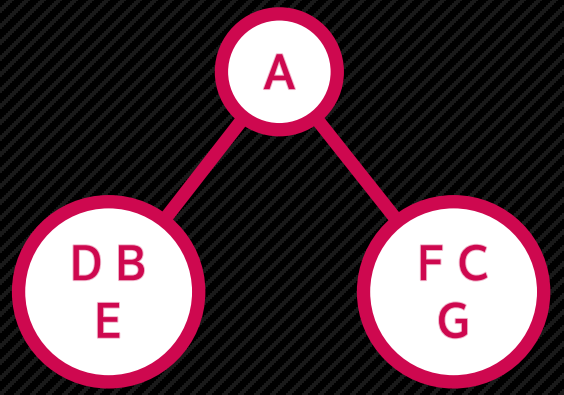
```
function InOrder(node)
    InOrder(node.left)
    visit(node)
    InOrder(node.right)
```

D - B - E - A - F - C - G

- 후위 순회 (Postorder Traversal)

```
function PostOrder(node)
    PostOrder(node.left)
    PostOrder(node.right)
    visit(node)
```

D - E - B - F - G - C - A



뜯어보자.

Gold 3 - 트리의 순회 (#2263)

- 중위 순회 (Inorder Traversal)

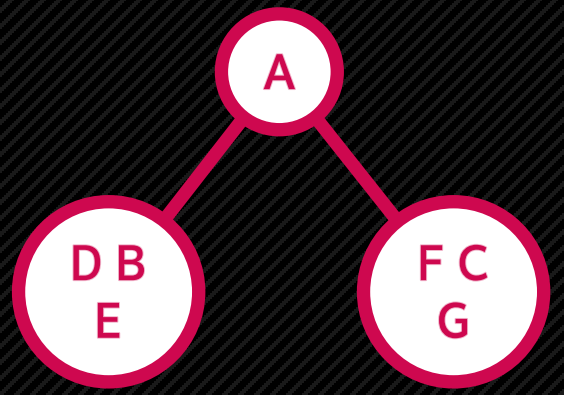
```
function InOrder(node)
    InOrder(node.left)
    visit(node)
    InOrder(node.right)
```

D - B - E - A - F - C - G

- 후위 순회 (Postorder Traversal)

```
function PostOrder(node)
    PostOrder(node.left)
    PostOrder(node.right)
    visit(node)
```

D - E - B - F - G - C - A



뜯어보자.

Gold 3 - 트리의 순회 (#2263)

- 중위 순회 (Inorder Traversal)

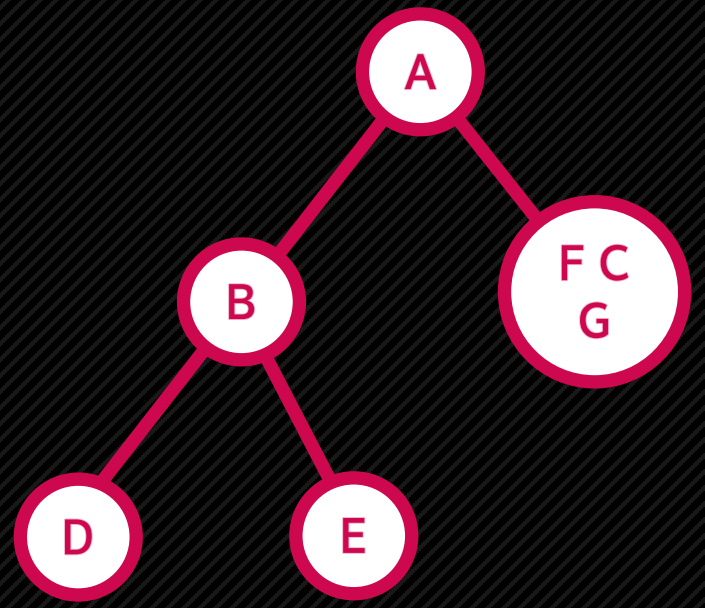
```
function InOrder(node)
    InOrder(node.left)
    visit(node)
    InOrder(node.right)
```

D - B - E - A - F - C - G

- 후위 순회 (Postorder Traversal)

```
function PostOrder(node)
    PostOrder(node.left)
    PostOrder(node.right)
    visit(node)
```

D - E - B - F - G - C - A



뜯어보자.

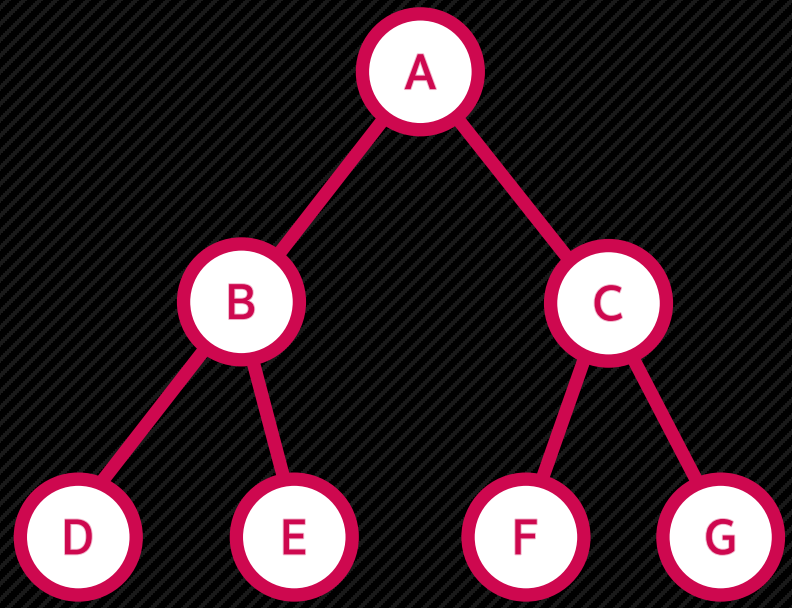
Gold 3 - 트리의 순회 (#2263)

- 중위 순회 (Inorder Traversal)

```
function InOrder(node)
    InOrder(node.left)
    visit(node)
    InOrder(node.right)
```

- 후위 순회 (Postorder Traversal)

```
function PostOrder(node)
    PostOrder(node.left)
    PostOrder(node.right)
    visit(node)
```



결론은...

Gold 3 - 트리의 순회 (#2263)

```
function GetPreOrder(post_start, in_start, length)
    if (length == 1)
        print InOrder[in_start]
        end

    set target -> PostOrder[post_start + length - 1]
    print(target)

    for(i = 0 ... length)
        if (InOrder[in_start + i] == target)
            set pos -> i

    GetPreOrder(post_start, in_start, pos)
    GetPreOrder(post_start + pos + 1, in_start + pos + 1, length - pos - 1)
```

비슷한 문제!

Level 3 - 길 찾기 게임 (2019 kakao Blind Recruitment 1차 코딩테스트)

문제가 깁니다. 중요한 부분만 골라서 해석하는 연습을 해봅시다!

지름!

Gold 3 - 트리의 지름 (#1167)

요약

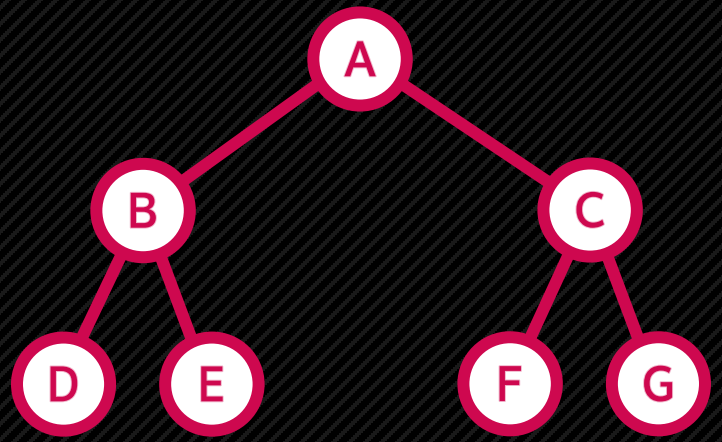
- 트리의 지름이란, 임의의 두 점 사이의 거리 중 가장 긴 것을 말한다. 지름을 구하는 프로그램을 작성하시오.

제약조건

- 정점의 수 V는 $2 <= V <= 100,000$ 이다.
- 정점과 정점 사이의 거리는 $1 <= E_i <= 10,000$ 이다.

가장 긴 거리?

Gold 3 - 트리의 지름 (#1167)



- 한 지점에서 가장 먼 거리를 구함.
- 그 지점에서 제일 먼 거리를 구하면, 그것이 지름이 됨.
- 귀류법을 통해 증명할 수 있으나, 생략함. (https://www.quora.com/How-does-following-algorithm-for-finding-longest-path-in-tree-work)

근데 잠깐, 입력의 상태가?

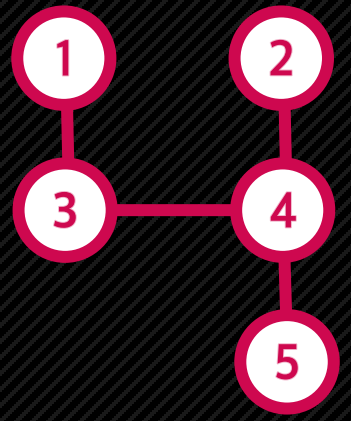
Gold 3 - 트리의 지름 (#1167)

```
5
1 3 2 -1
2 4 4 -1
3 1 2 4 3 -1
4 2 4 3 3 5 6 -1
5 4 6 -1
```

근데 잠깐, 입력의 상태가?

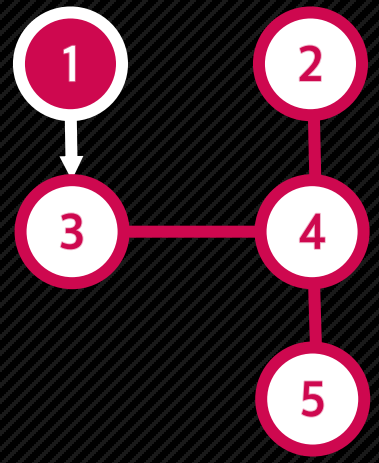
Gold 3 - 트리의 지름 (#1167)

```
5
1 3 2 -1
2 4 4 -1
3 1 2 4 3 -1
4 2 4 3 3 5 6 -1
5 4 6 -1
```

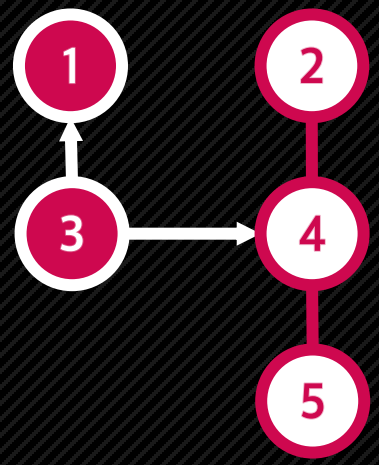


- 부모-자식 관계를 어떻게 찾을까?

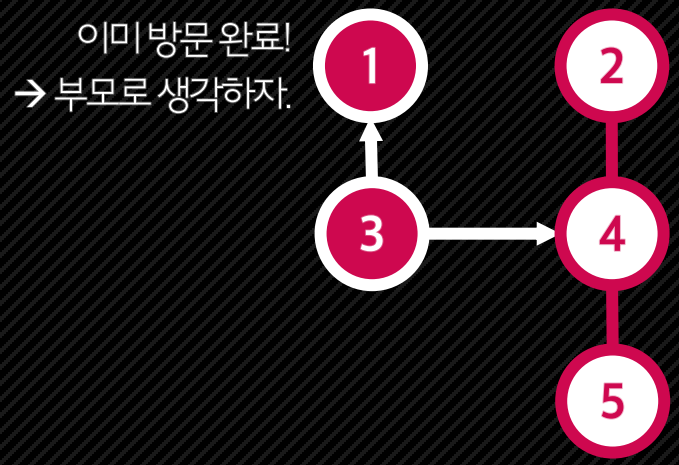
Gold 3 - 트리의 지름 (#1167)



Gold 3 - 트리의 지름 (#1167)

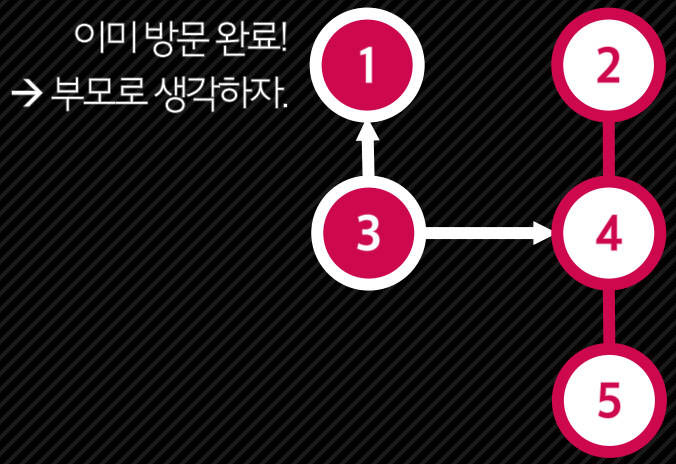


Gold 3 - 트리의 지름 (#1167)



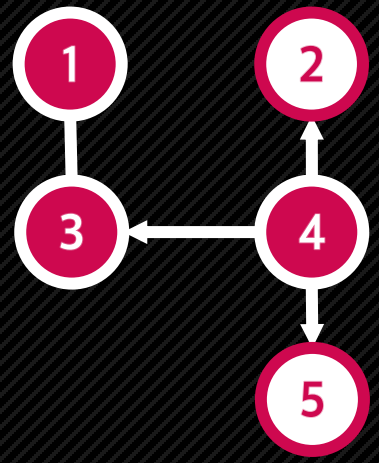
트리 찾기

Gold 3 - 트리의 지름 (#1167)



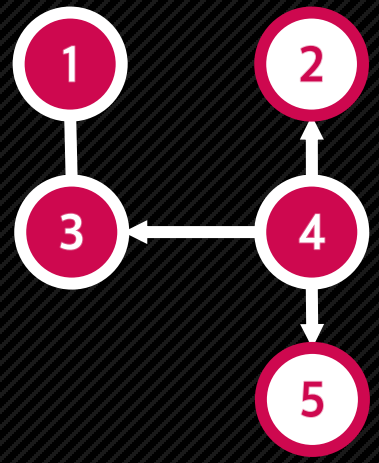
트리 찾기

Gold 3 - 트리의 지름 (#1167)



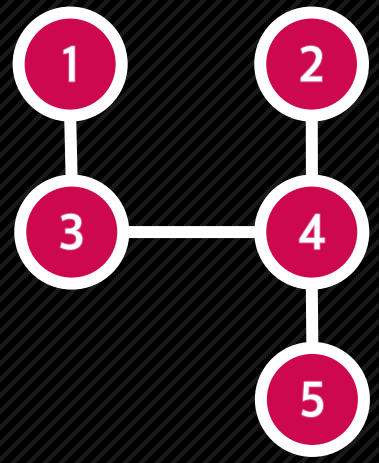
트리 찾기

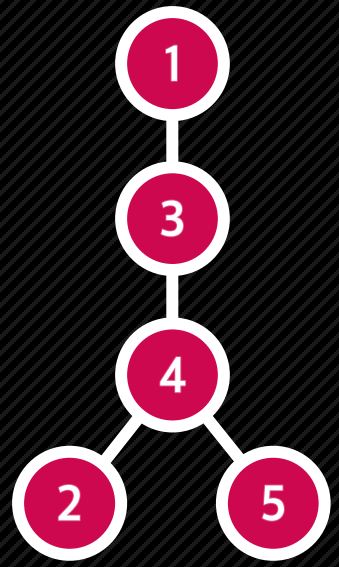
Gold 3 - 트리의 지름 (#1167)



트리 찾기

Gold 3 - 트리의 지름 (#1167)





트리 찾기

Gold 3 - 트리의 지름 (#1167)

```
function DFS(node)
    visited[node] = true
    for (next <- graph[node])
        if visited[next] == false
            append next to tree[node]
            DFS(next)
```

트리에서 DP를?

Gold 1 - 우수 마을 (#1949)

요약

- 트리 구조로 구성된 마을이 있다. 이 마을 중 몇 개를 골라 우수 마을로 선정하려고 한다.
- 단, 우수 마을끼리는 인접할 수 없으며, 선정되지 않은 마을은 최소 한 개 이상의 우수 마을과 인접해 있어야 한다.
- 이 때, 우수 마을에 선정된 마을의 주민수의 합의 최댓값을 구하는 프로그램을 작성하시오.

제약조건

- 정점의 수 N은 1 <= N <= 10,000 이다.
- 주민의 수는 10,000 이하 이다.

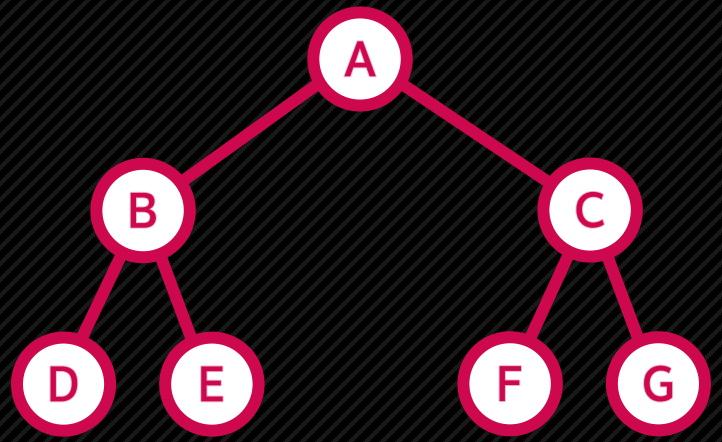
이게 왜 DP야?

Gold 1 - 우수 마을 (#1949)

- 큰 문제를 작은 문제로 쪼갰을 때, **작은 문제의 답을 통해 큰 문제의 답을 도출할 수 있는가?**
- 큰 문제를 해결하기 위해 작은 문제의 답을 **여러 번 구해야 하는가?**

트리의 특성을 고려해보자.

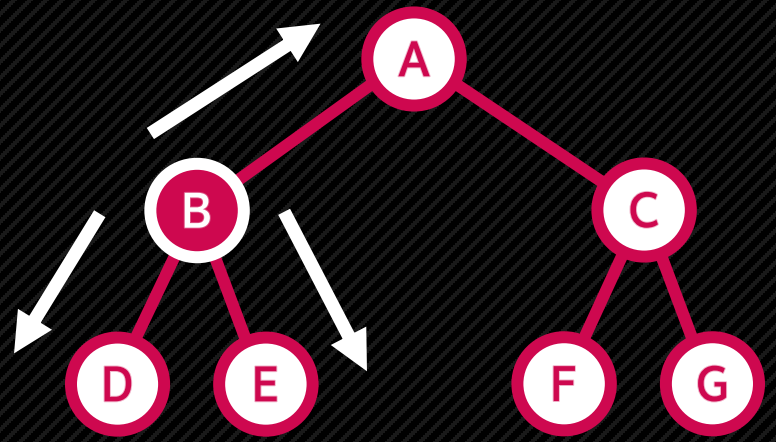
Gold 1 - 우수 마을 (#1949)



- 단, 우수 마을끼리는 인접할 수 없으며, 선정되지 않은 마을은 최소 한 개 이상의 우수 마을과 인접해 있어야 한다.
- 이 때, 우수 마을에 선정된 마을의 주민수의 합의 최댓값을 구하는 프로그램을 작성하시오.

위/아래를 일일히 따지긴 어려우니까…

Gold 1 - 우수 마을 (#1949)

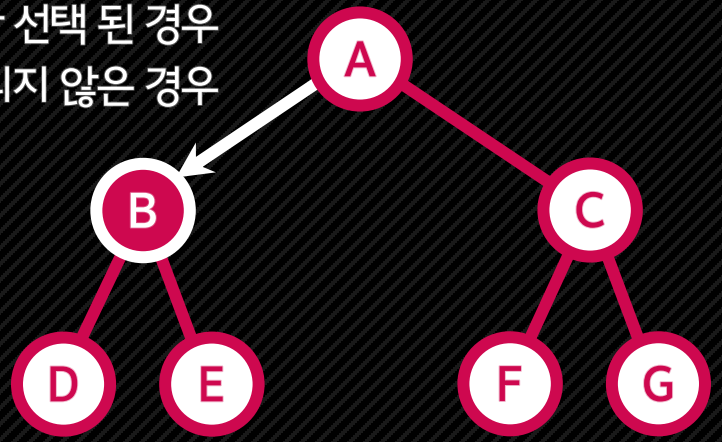


- 단, 우수 마을끼리는 인접할 수 없으며, 선정되지 않은 마을은 최소 한 개 이상의 우수 마을과 인접해 있어야 한다.
- 이 때, 우수 마을에 선정된 마을의 주민수의 합의 최댓값을 구하는 프로그램을 작성하시오.

윗 부분을 고정시켜버리자!

Gold 1 - 우수 마을 (#1949)

- A가 선택 된 경우
- A가 선택되지 않은 경우

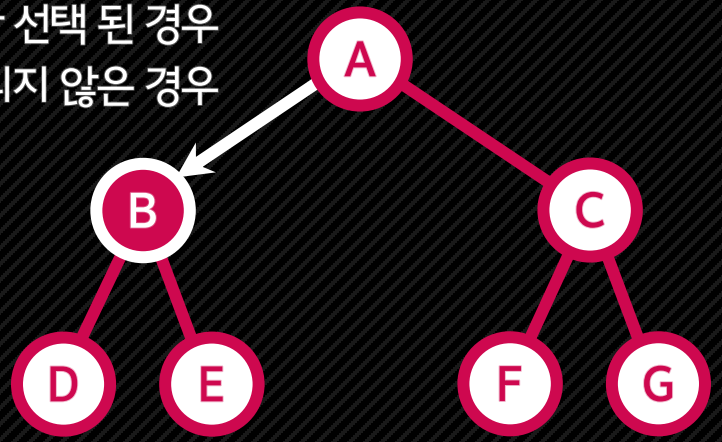


- 단, 우수 마을끼리는 인접할 수 없으며, 선정되지 않은 마을은 최소 한 개 이상의 우수 마을과 인접해 있어야 한다.
- 이 때, 우수 마을에 선정된 마을의 주민수의 합의 최댓값을 구하는 프로그램을 작성하시오.

윗 부분을 고정시켜버리자!

Gold 1 - 우수 마을 (#1949)

- A가 선택 된 경우
- A가 선택되지 않은 경우

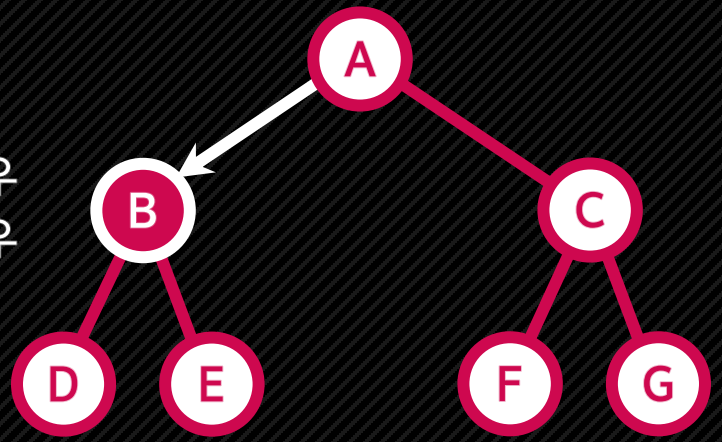


- 단, 우수 마을끼리는 인접할 수 없으며, 선정되지 않은 마을은 최소 한 개 이상의 우수 마을과 인접해 있어야 한다.
- 이 때, 우수 마을에 선정된 마을의 주민수의 합의 최댓값을 구하는 프로그램을 작성하시오.

윗 부분을 고정시켜버리자!

Gold 1 - 우수 마을 (#1949)

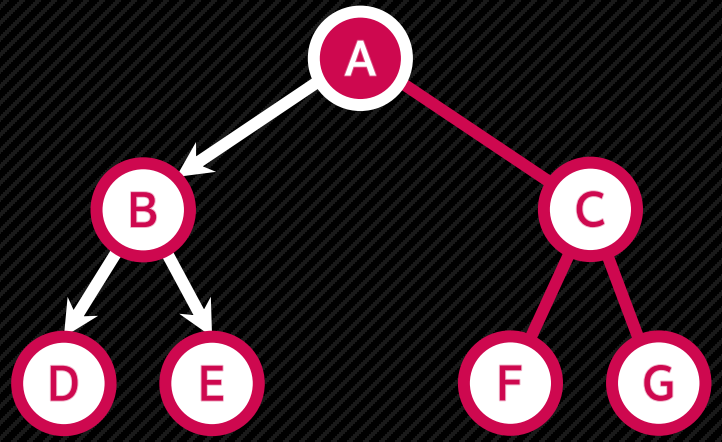
- A가 선택 된 경우
- A가 선택되지 않은 경우



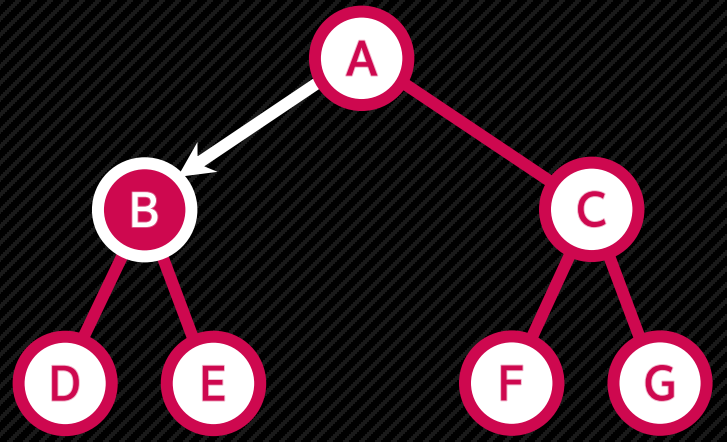
- 단, 우수 마을끼리는 인접할 수 없으며, 선정되지 않은 마을은 최소 한 개 이상의 우수 마을과 인접해 있어야 한다.
- 이 때, 우수 마을에 선정된 마을의 주민수의 합의 최댓값을 구하는 프로그램을 작성하시오.

경우를 분리해보자.

Gold 1 - 우수 마을 (#1949)



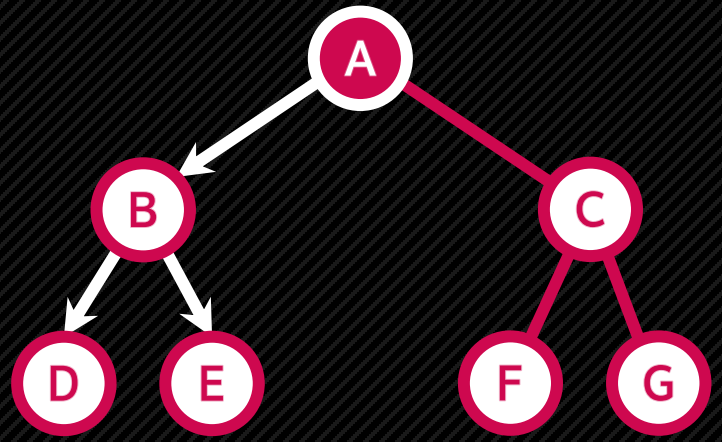
A가 선택 된 경우



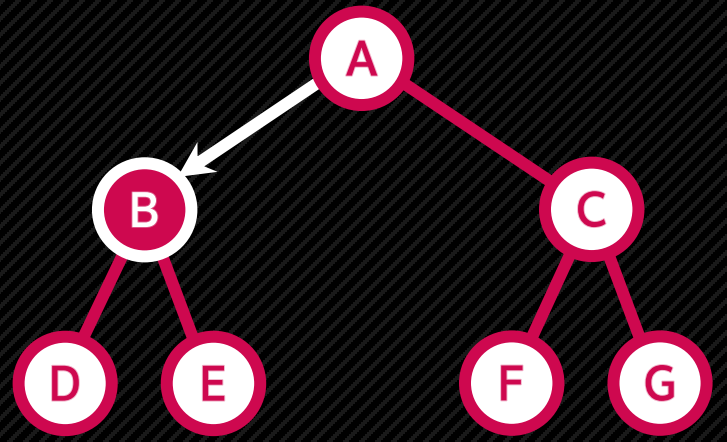
A가 선택 되지 않은 경우

경우를 분리해보자.

Gold 1 - 우수 마을 (#1949)



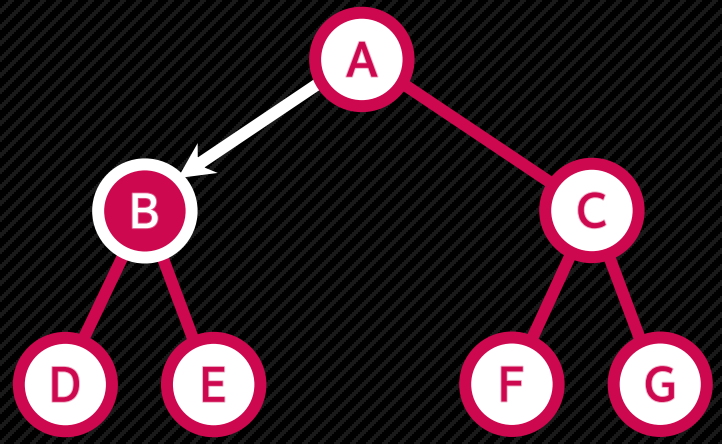
B는 절대 선택 되어서는 안됨.



B는 선택해도 되고, 아니어도 되고…

이거는요?

Gold 1 - 우수 마을 (#1949)

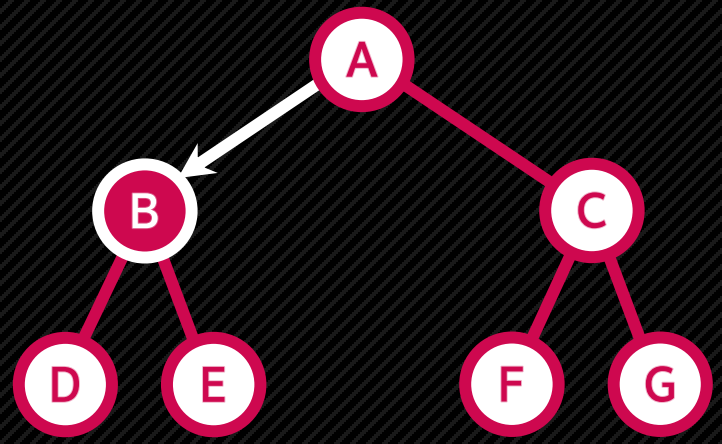


선정되지 않은 마을은 최소 한 개 이상의 우수 마을과 인접해 있어야 한다.

B는 선택해도 되고, 아니어도 되고…

이거는요?

Gold 1 - 우수 마을 (#1949)



B는 선택해도 되고, 아니어도 되고…

선정되지 않은 마을은 최소 한 개 이상의 우수 마을과 인접해 있어야 한다.

→ 만약에 자기 주변 애들이 다 안 선택되면 어쩌려고요??

이거는요?

Gold 1 - 우수 마을 (#1949)

선정되지 않은 마을은 최소 한 개 이상의 우수 마을과 인접해 있어야 한다.

→ 만약에 자기 주변 애들이 다 안 선택되면 어쩌려고요??

이거는요?

Gold 1 - 우수 마을 (#1949)

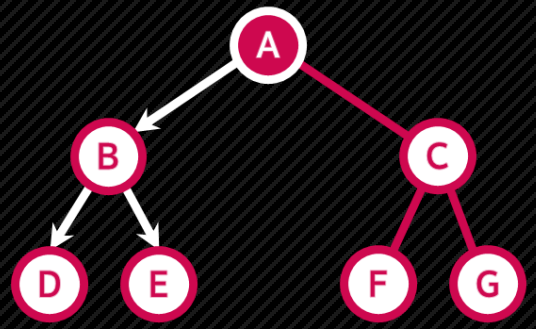
선정되지 않은 마을은 최소 한 개 이상의 우수 마을과 인접해 있어야 한다.

→ 만약에 자기 주변 애들이 다 안 선택되면 어쩌려고요??

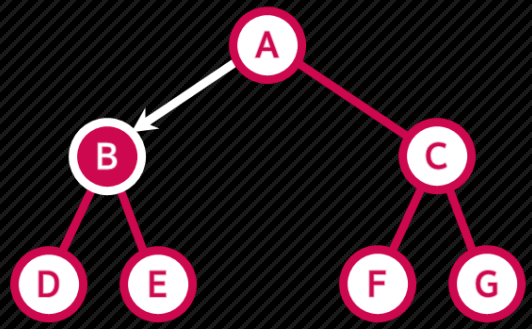
→ 어차피 최댓값으로 비교하면 밀려서 괜찮다!

아무튼…

Gold 1 - 우수 마을 (#1949)



B는 절대 선택 되어서는 안됨.



B는 선택해도 되고, 아니어도 되고…

- 두 경우를 테스트 하기 위해 **트리 아래쪽으로 반복적으로 내려감.**
- 결국, 똑같은 부분 문제를 계속해서 풀어야 함! → DP 맞네!

힌트만 드릴게.

Gold 1 - 우수 마을 (#1949)

DP[10001][2]

몇 번째? 선택? 아냐?

마지막 문제.

 Level 4 - 매출 하락 최소화 (2021 kakao Blind Recruitment 1차 코딩테스트)

추가 추천 문제

Gold 4 - 트리 (#4803)

- 트리의 성질을 기억하세요!

Gold 2 - 트리의 높이와 너비 (#2250)

- 트리의 높이는 공부하였습니다. 그렇다면 너비는 어떻게 구할까요?

Gold 1 - 트리의 독립집합 (#2213)

- 트리 DP는 코테에 최근에 등장한 유형입니다. 문제가 적은 만큼 하나 하나 꼼꼼히 풀어봅시다.

Level 4 - 동굴 탐험 (2020 카카오 인턴십 코딩테스트)

- 트리의 구조를 잘 생각해보면 그렇게 어렵지 않게 구현할 수 있습니다.

끝!

그동안 쉬웠지 떠라잇시느라 고생 많았습니다!

코테에서 다루는 대부분의 자료구조/알고리즘을 학습하셨으나, 아직 배우지 못한 부분이 있습니다,

추가적인 토픽 (위상정렬, 비트마스크, 트라이, 세그먼트 트리)는 PPT가 아닌 문서 형태로 업로드 예정입니다,

한 번 복습해 보시면서 문제들을 직접 풀어 보시고, 하나씩 본인의 풀이를 체화하는 것이 중요합니다,

원하는 회사나, 원하는 목표를 달성할 수 있길 응원하겠습니다!

더 많은 문제는…

https://github.com/tony9402/baekjoon

- 최근 기업 코딩테스트 알고리즘 분류
- 특정 알고리즘 관련 추천 문제 모음
- 코딩테스트와 유사한 모의고사 (예정)

“Any Question?”

“Thank You!”