/* elice */

실리콘밸리 문제로 배우는 알고리즘 기초

자료구조의 끝판왕

박지나 선생님

목차

1. 지난 수업 복습
2. 재귀 함수
3. 트리

지난 수업 복습

지난 수업 복습

1) 연결 리스트

2) 큐

3) 스택

재귀 함수

프로그램의 핵심 : 되풀이

비슷한 일을 여러번 **되풀이**해서 풀어내기

```
def doSomething(nums):
    sum = 0
    for num in nums:
        sum = sum + num
    return sum
```

sum = sum + num의 반복!

반복 VS 재귀

반복 (Iteration)

우리가 지금까지 익히 썼던 개념

for, while 등을 사용한 되풀이법

재귀 (Recursion)

이제 배워봅시다!

재귀 (Recursion)

스스로를 호출하는 방식의 반복법

어떤 주어진 **일(Task)**이 같은 과정을 필요로 하지만
더 범위가 **작은 일(Sub Task)**들로 나눠질 수 있다면,
재귀를 적용할 수 있다.

재귀 (Recursion)

언젠가는 끝이 나야하므로 식의 종료 조건이 필요

= **Base** 조건 (ex. f(0) = 1, f(1) = 2 …)

예) 1을 n번 더하기

Task(n) = 1 + Task(n-1)

= 2 + Task(n-2)

…

= n + **Task(0)** = n + **0** = n

예) 팩토리얼 계산

$$factorial(n) = n! = n * n-1 * n-2 * \cdots * 1$$

$$factorial(n-1) = (n-1)! = n-1 * n-2 * \cdots * 1$$

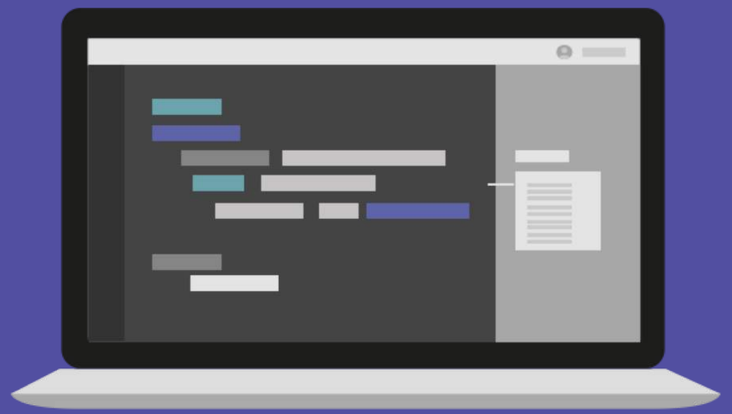
이 경우,

$$factorial(n) = n * factorial(n-1)$$

즉, factorial 구현 안에서 factorial 사용 가능 = **재귀**

[실습1]

팩토리얼 계산



심화 : 동적 프로그래밍

Dynamic Programming

재귀 + 정보 저장 (메모이제이션)

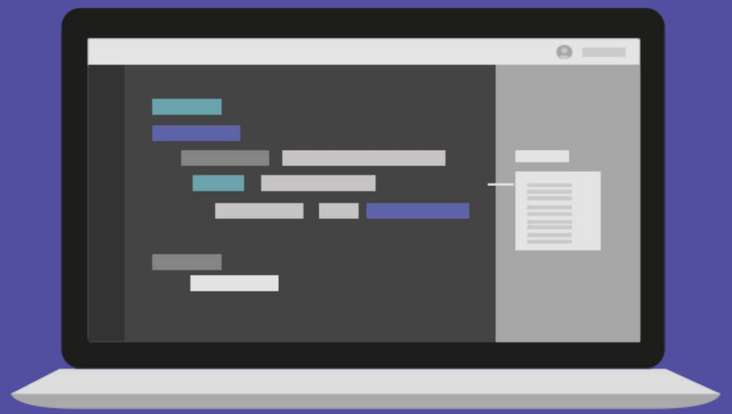
한 부분 문제를 한 번 계산했다면

다시 계산 할 필요가 없도록!

→ 저장 값을 다른 자료 구조에 저장

[실습2]

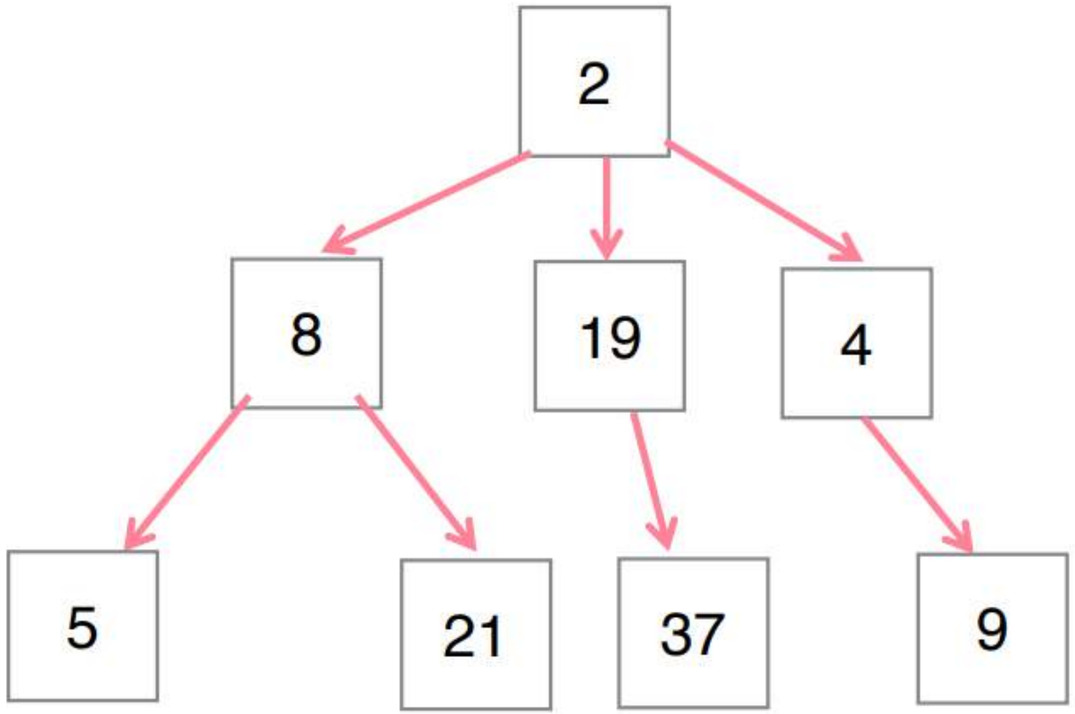
피보나치 수



트리

트리

나무 형태의 자료구조



부모와 자식

부모 노드 → 자식 노드 방향으로 연결이 존재

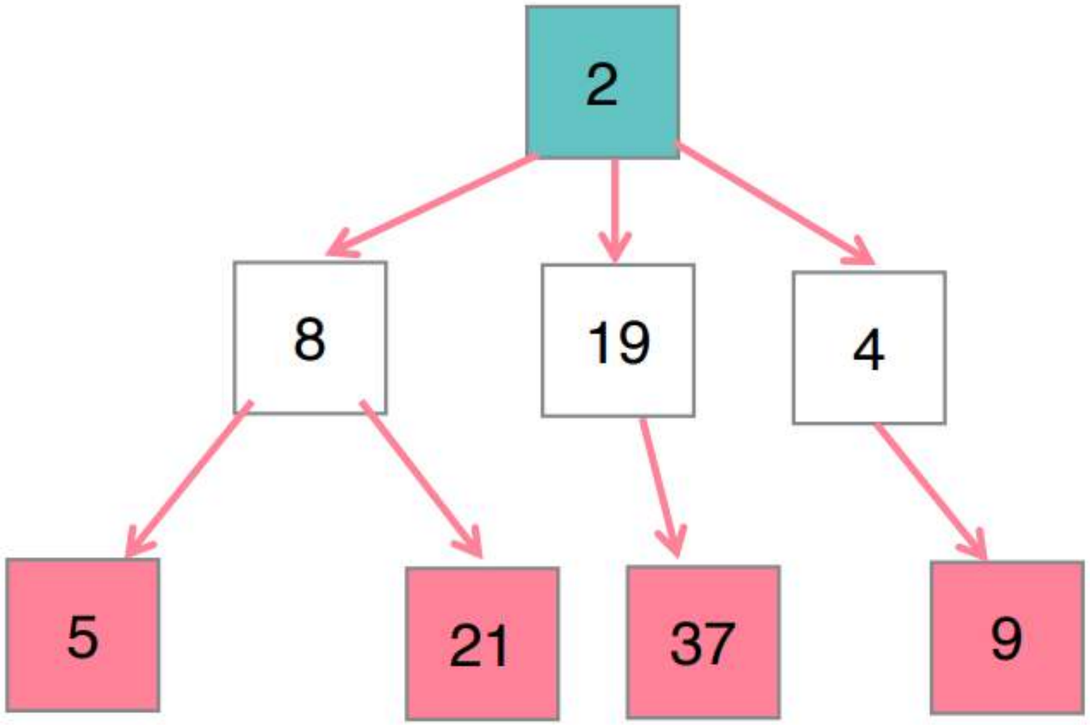
2 노드를 8 노드의 **부모 노드**

8 노드를 2 노드의 **자식 노드**

루트와 리프

루트(root) : 부모가 없는 노드

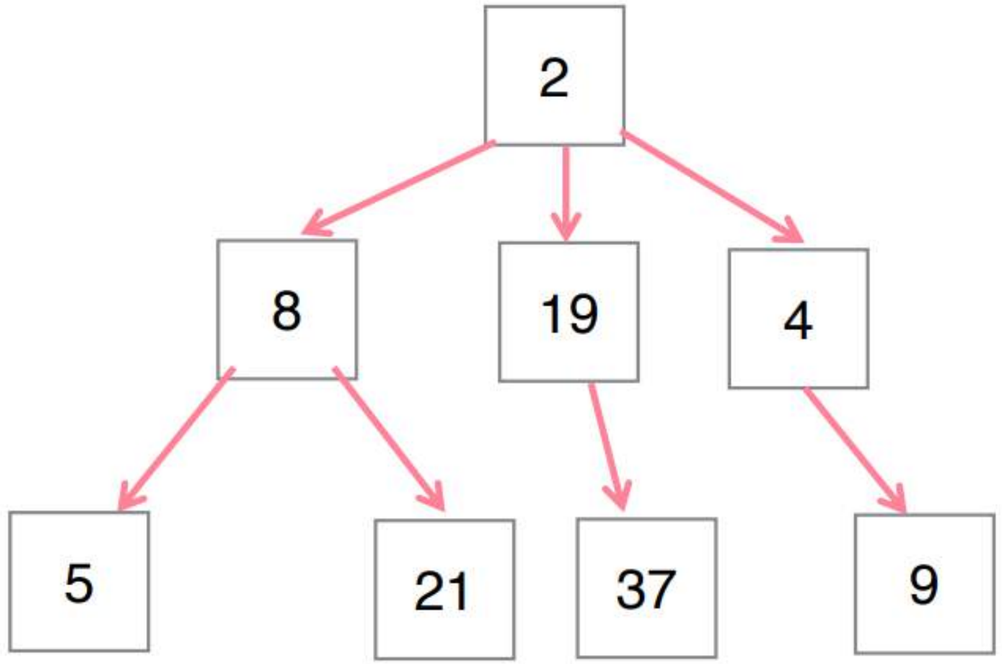
리프(leaf) : 자식이 없는 노드



트리의 깊이

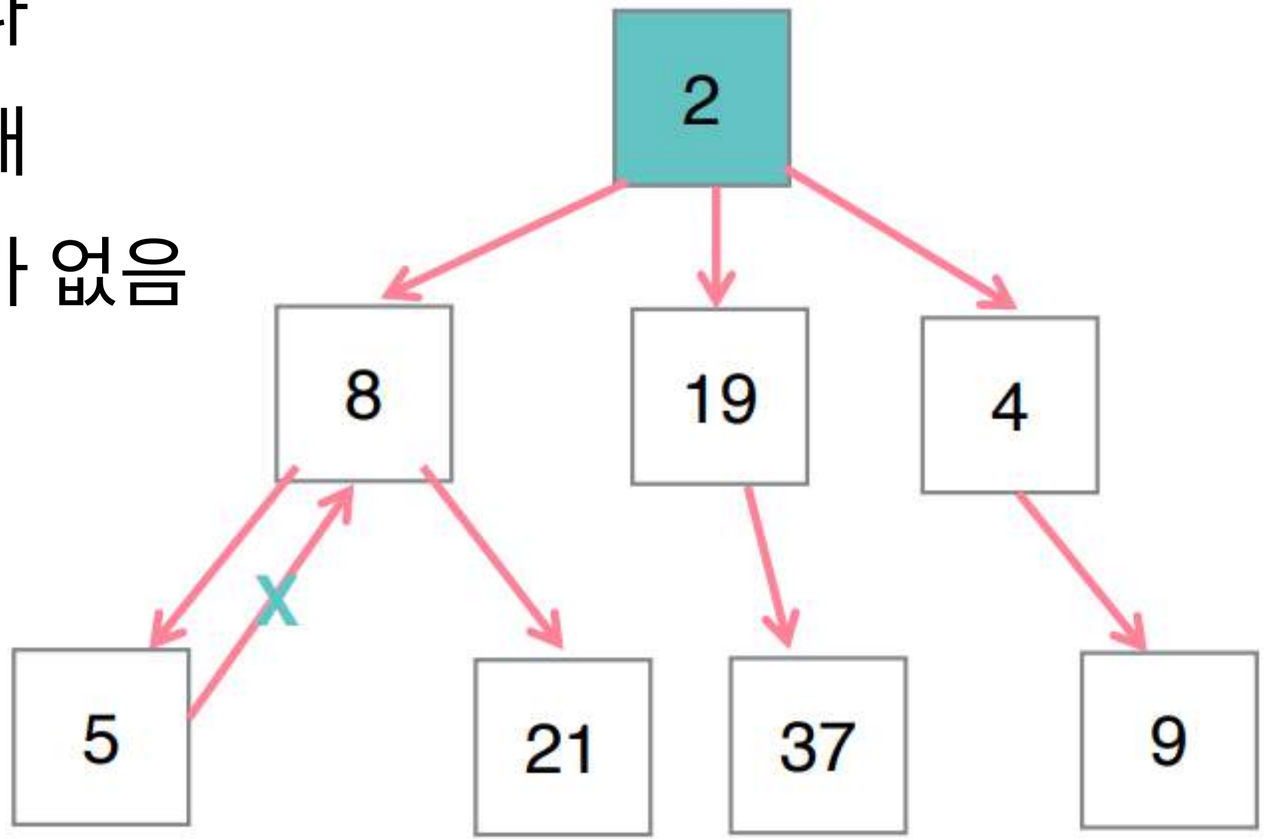
루트에서 **리프**까지의 경로의 길이, Depth

트리의 깊이 =



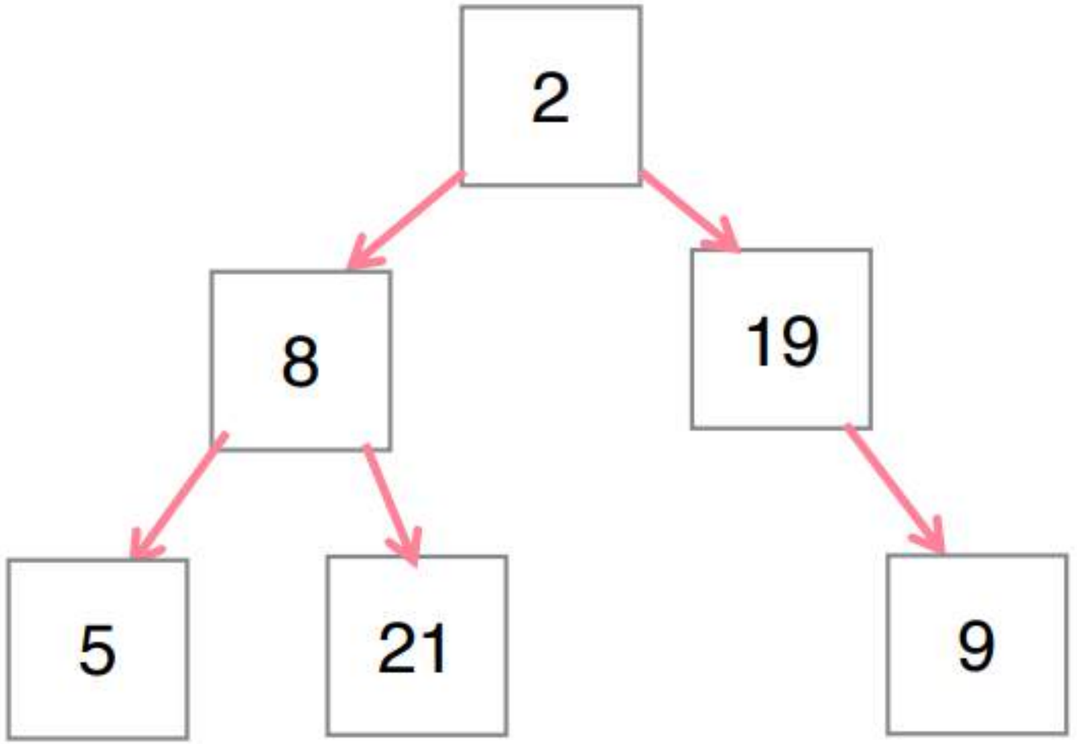
트리의 특성

1. 루트는 하나
2. 방향성 존재
3. 순환 구조가 없음



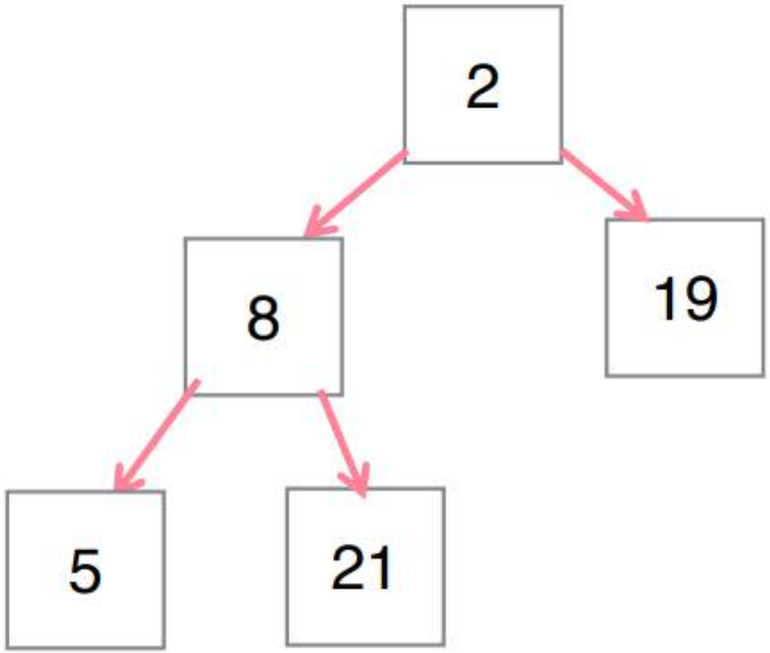
이진 트리

모든 노드가 최대 2개의 자식 노드를 가지는 트리

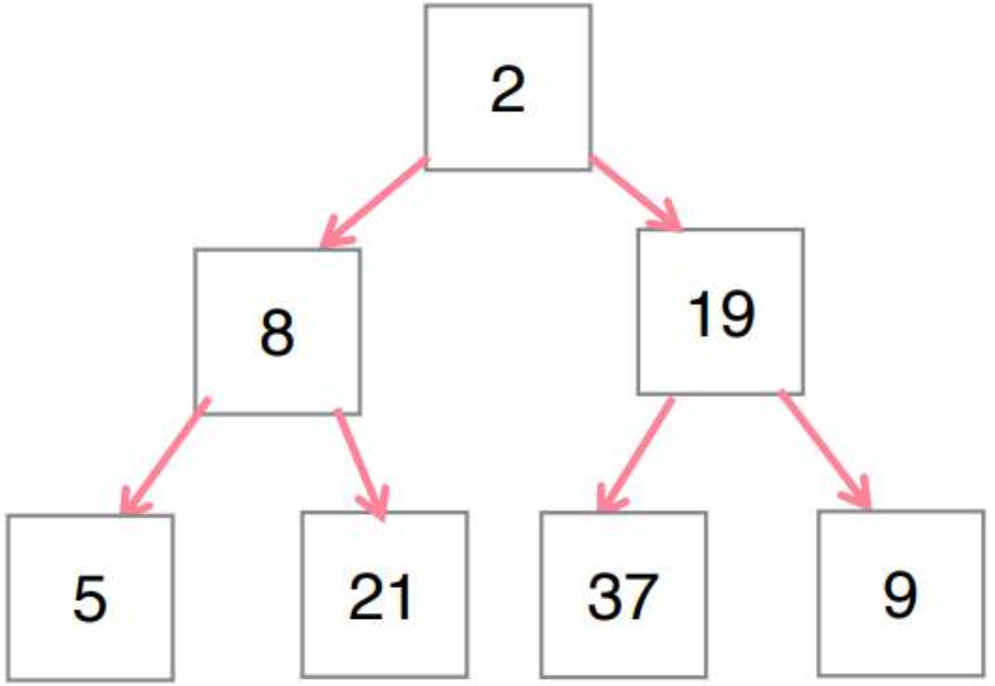


완전/포화 이진 트리

완전 이진 트리

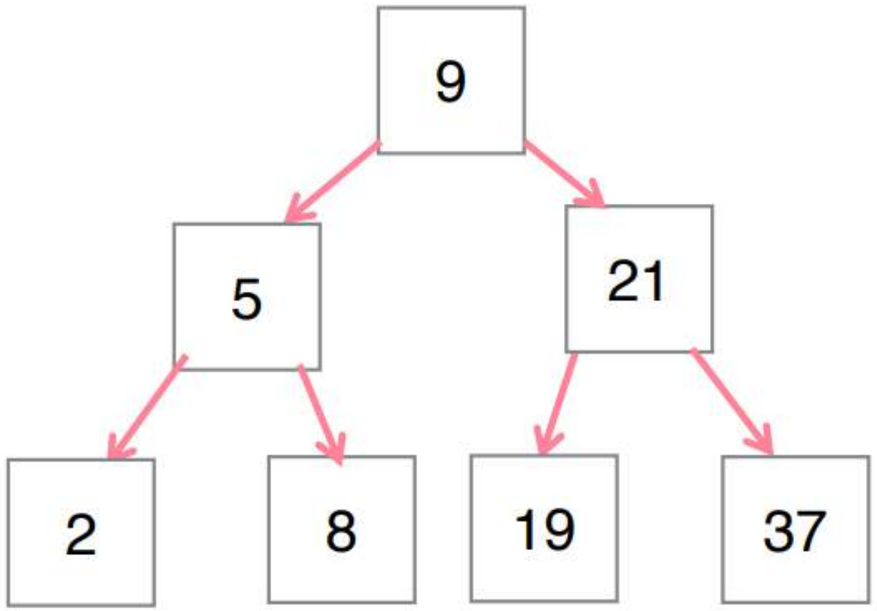


포화 이진 트리



이진 탐색 트리

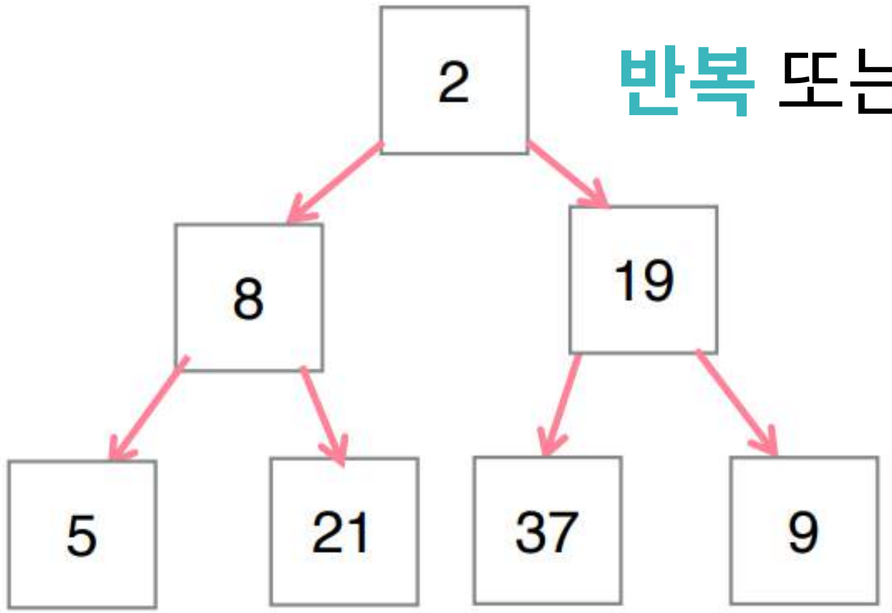
Binary Search Tree



모든 부모 노드의 값이
왼쪽 자식 트리에 있는
값보다는 **크고**
오른쪽 자식 트리에 있는
값보다는 **작은**
형태의 트리

트리 관련 문제의 핵심 = 탐색

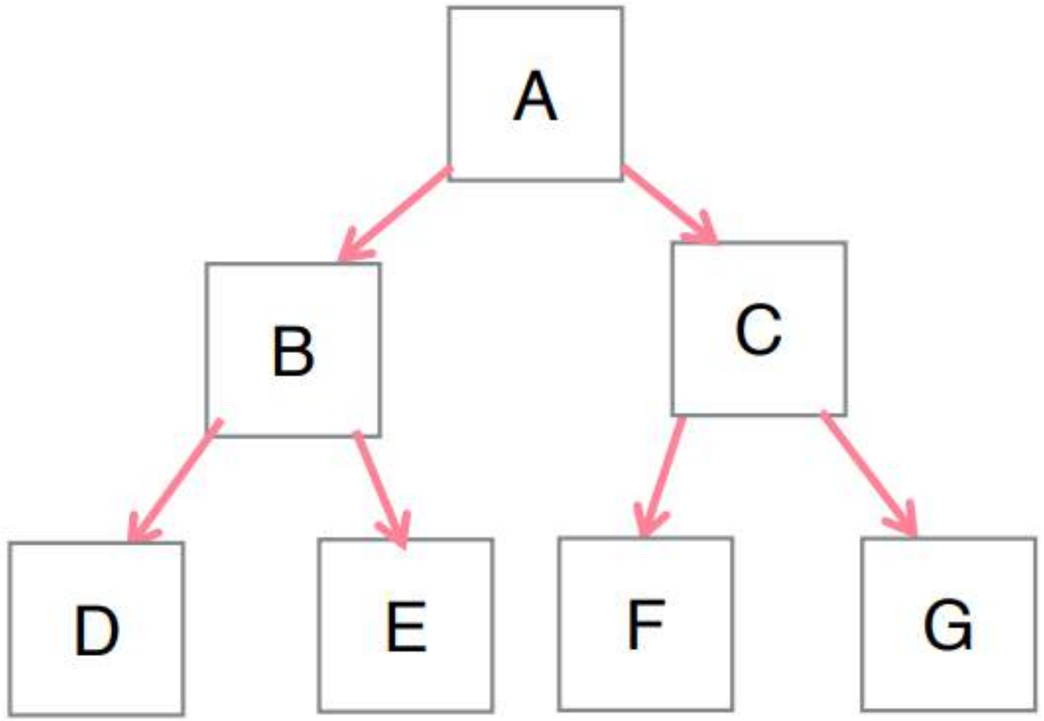
루트 노드가 주어졌을 때 트리를 구석구석 훑어가며
원하는 목적을 달성하는 것!



반복 또는 **재귀**를 활용!

너비 우선 탐색 (BFS)

Breadth First Search : **반복 기반**의 탐색

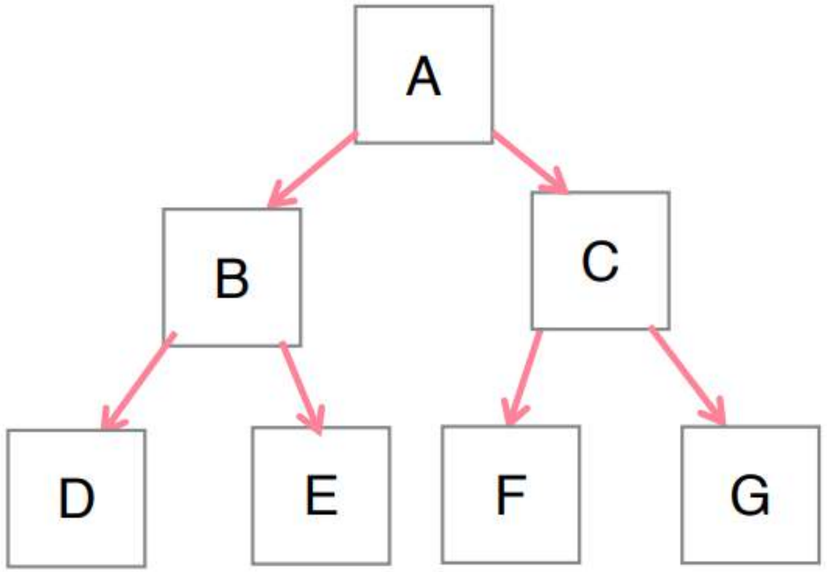


너비 우선 탐색 (BFS)

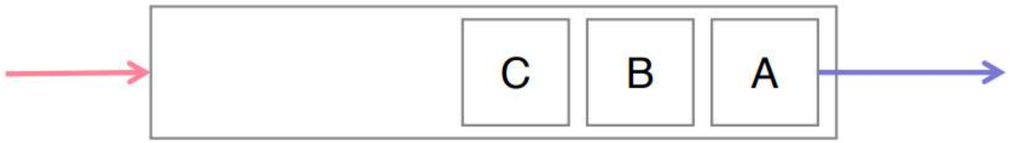
```
def BFS(root):
  q = queue.Queue()
  q = put(root)
    while q.qsize() > 0:
      node = q.get()
      if node:
        //doSomething
      q.put(node.left)
      q.put(node.right)
```

너비 우선 탐색 (BFS)

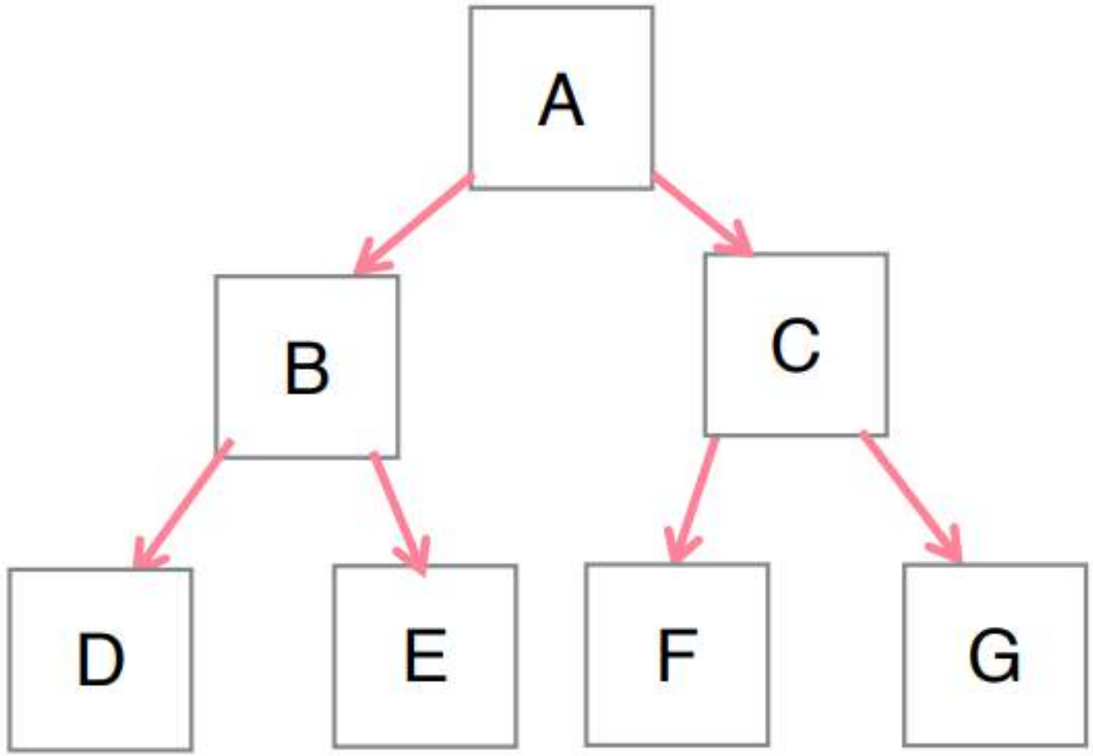
큐에 노드를 순서대로 넣고 빼는 방식으로 탐색



큐 (Queue)

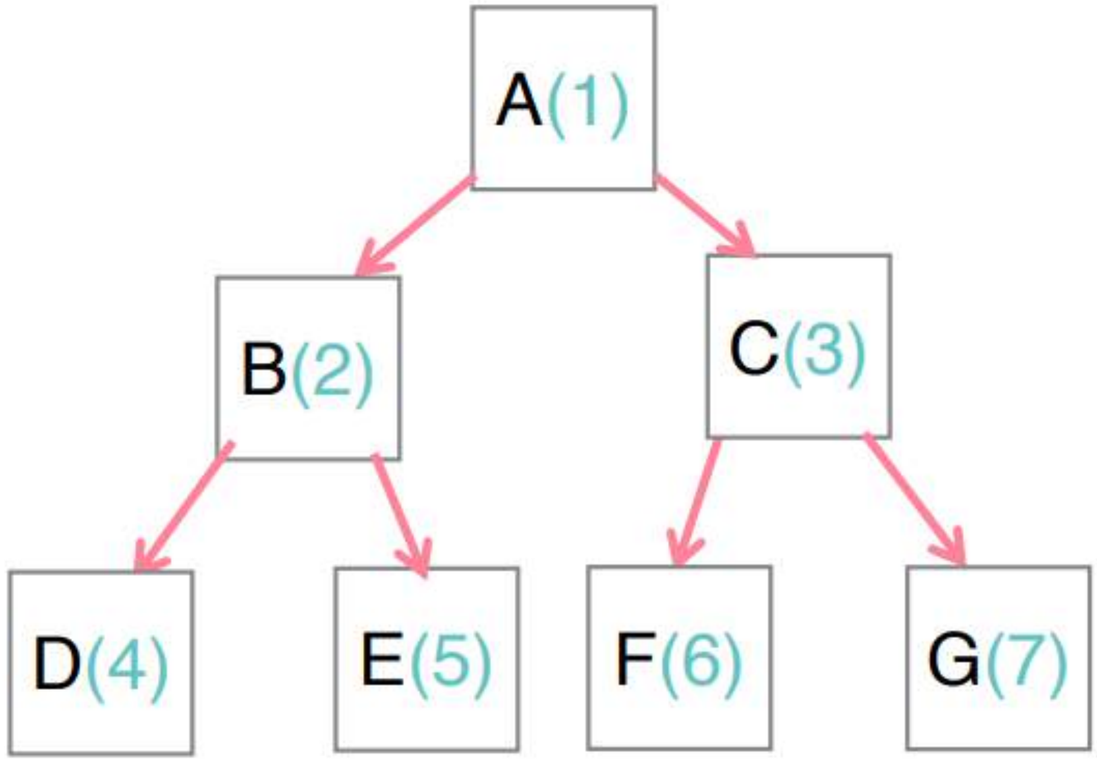


너비 우선 탐색 (BFS) : 예시



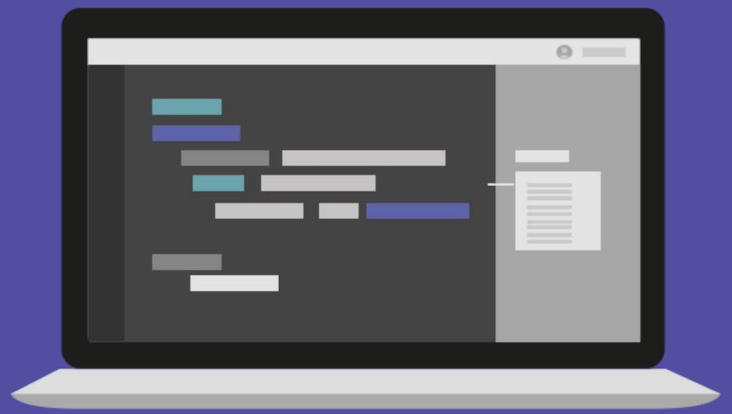
너비 우선 탐색 (BFS)

횡적으로 한 층씩 탐색하는 방식



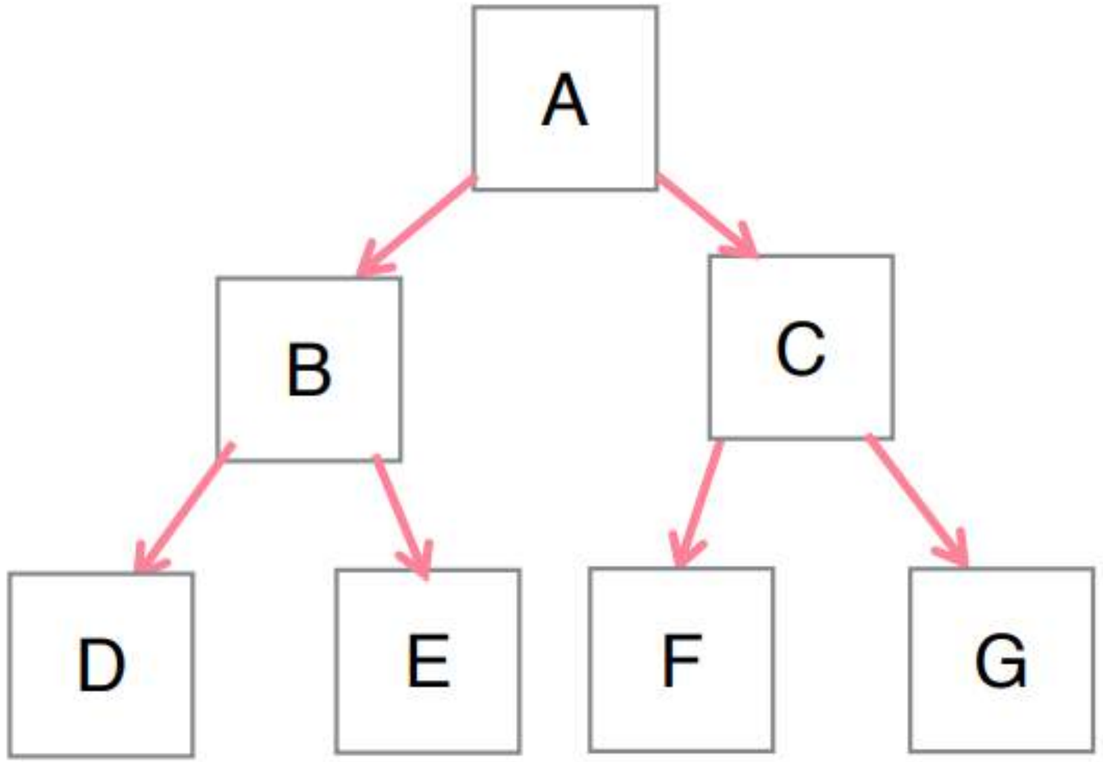
[실습3]

이진 트리 출력하기



깊이 우선 탐색 (DFS)

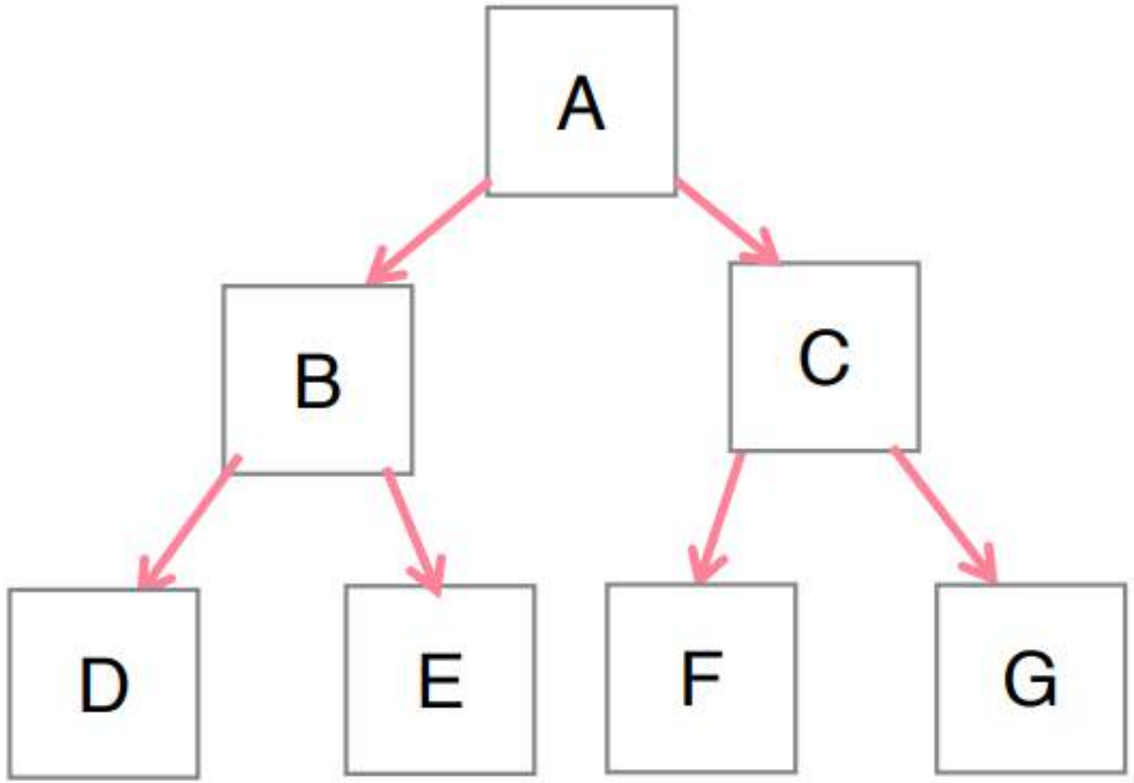
Depth First Search : **재귀 기반**의 탐색



깊이 우선 탐색 (DFS)

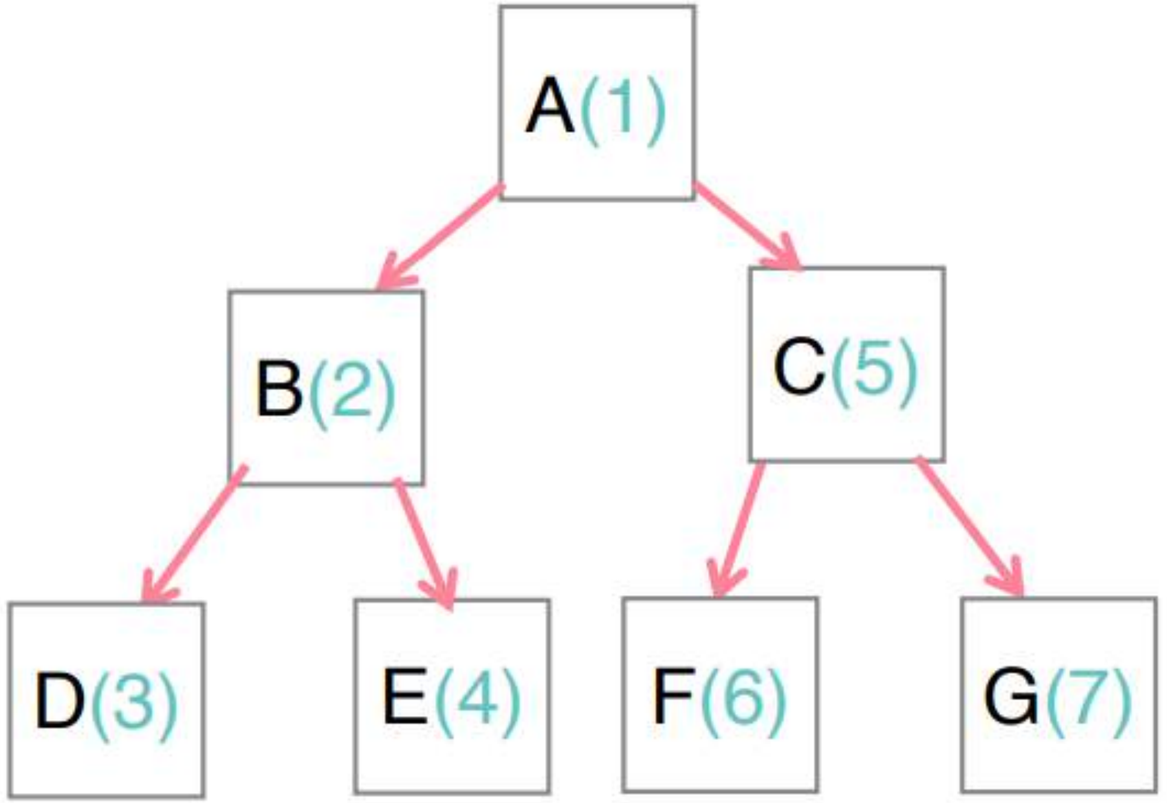
```
def DFS(node):
    //doSomething
    if node == 리프노드:
        doSomething
        return
    DFS(node.left)
    DFS(node.right)
```

깊이 우선 탐색 (DFS) : 예시



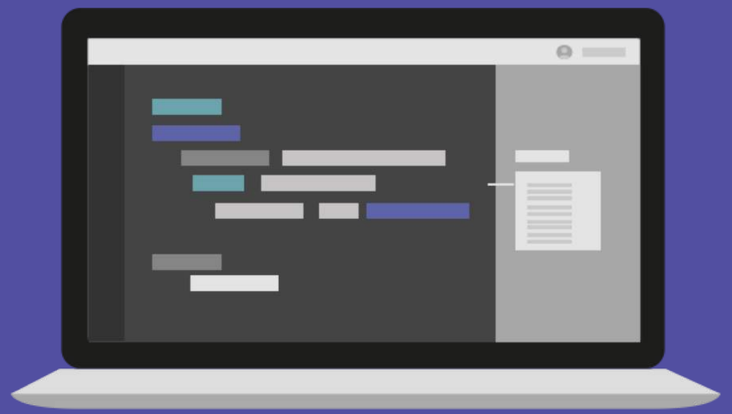
깊이 우선 탐색 (DFS)

가장 깊은 곳까지 내려갔다가 오는 방식의 탐색



[실습4]

트리 경로의 합



오늘 배운 내용

1) 재귀 함수

2) 트리

/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

medium.com/elice