Day 4 [Tree]

enumerate 함수 찾아보기 (열거형 함수[배열])

- Index값과 Index에 할당된 값을 둘다 리턴해준다.
- 주로 배열(리스트, 튜플)에 쓰인다.
- 파이썬이랑 자바(Class로 들어간다) 에서 주로 사용한다.
- 이노무새끼라고 외우자.

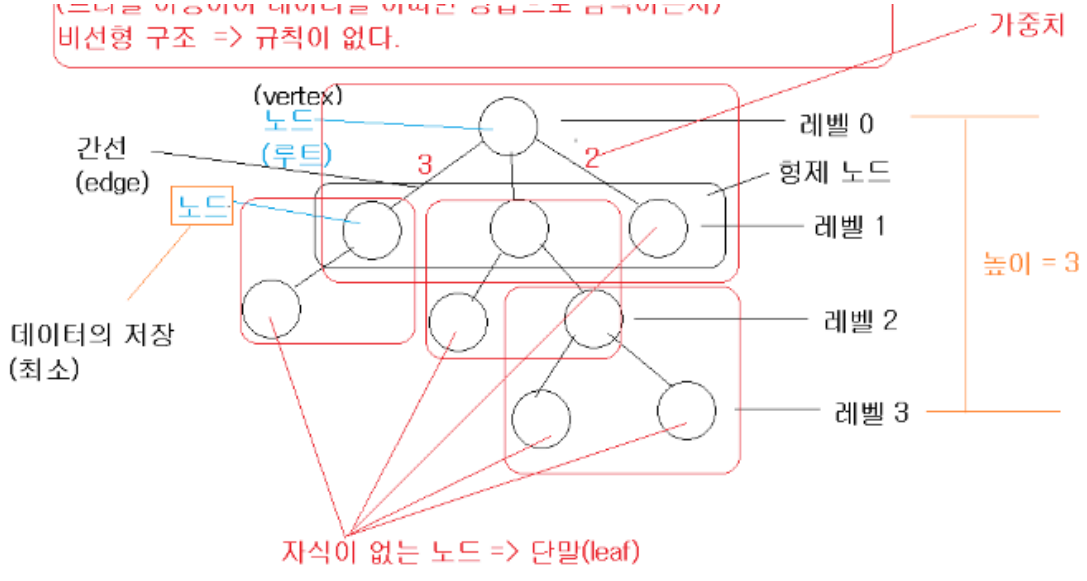
```
lst = ('dog','cat','rat','donkey')

for x,y in enumerate(lst):
    print(x)
    print(y)
```

```
0
dog
1
cat
2
rat
3
donkey
```

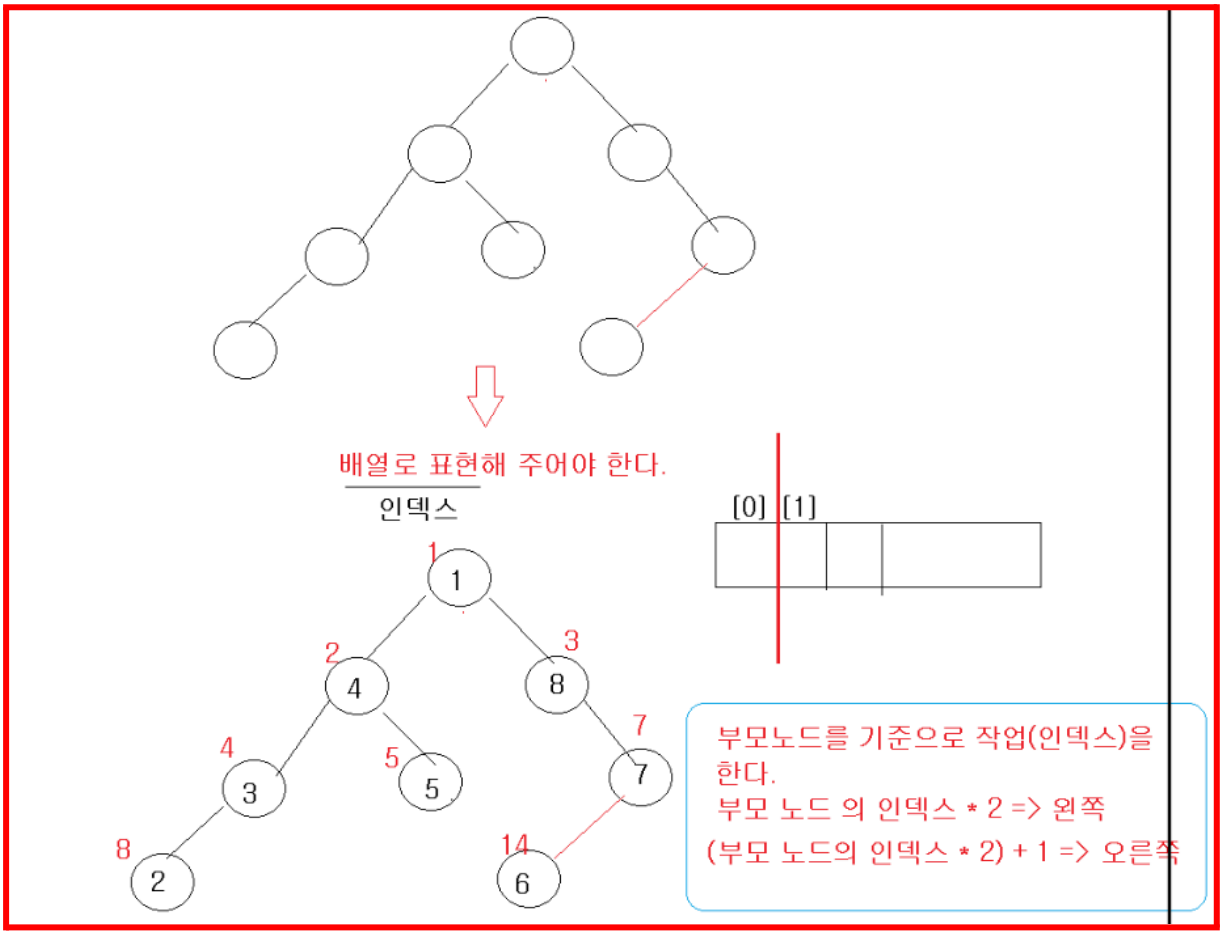
트리(Tree)

- 코딩 테스트에서 필수적으로 나오는 part이다. (그래프와도 함께)
- 데이터를 저장하고 탐색하기 유용한 구조를 가지고 있다.
    - (트리를 이용하여 데이터를 어떠한 방법으로 탐색하는지)
- **트리는 비선형 구조이다 (non-linear structure)**
    - 규칙이 없다. (어떠한 방법으로도 만들 수 있다)



- **트리의 구성은 1차수, 2차수, 3차수 … n차수**
    - 부모노드에서 자식이 몇개가 나가는지에 따라 정해진다
    - 코테에서는 이진트리(계산 하기 편하게 하기 위해서) 를 주로 쓴다
        - 2차수 (자식노드가 2개가 나감)
        - 컴퓨터는 0과 1로 이루어져 있다.

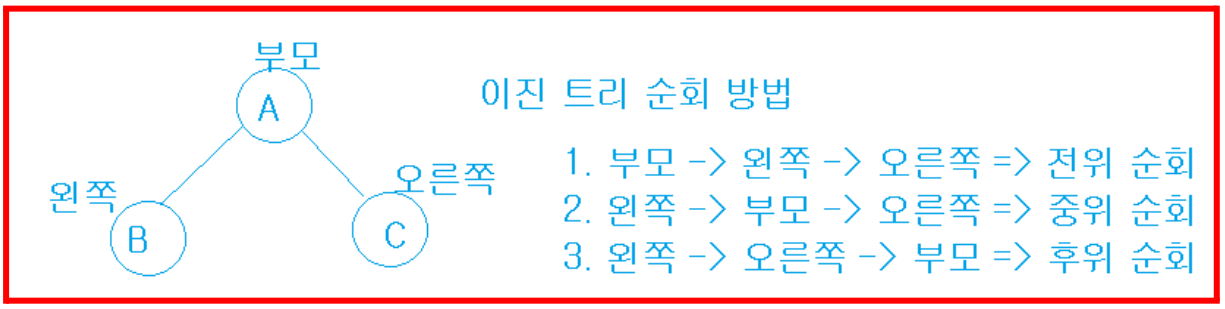
- **이진트리의 기준은 왼쪽 서브트리부터 시작**
    - **이진 트리를 배열로 표연해 주는 방법:**
        - 배열의 인덱스 0은 사용하지 않는다. 1부터 시작한다.
        - 부모노드를 기준으로 작업(인덱스)을 한다.
        - 부모노드의 인덱스 * 2 ⇒ 왼쪽 서브트리
        - (부모노드의 인덱스 * 2) +1 ⇒ 오른쪽 서브트리



ex) 위에 있는 사진에 있는 트리를 배열로 바꾸면:

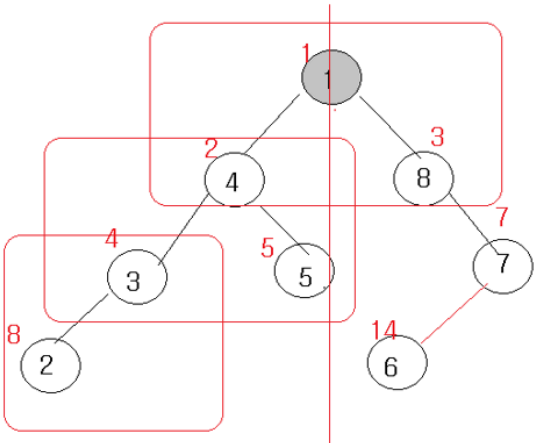
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
	1	4	8	3	5	14	7	2						

이진 트리 순회 방법 (1:28:00)

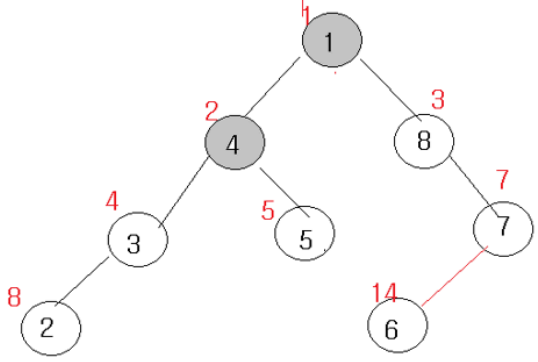


전위 순회:

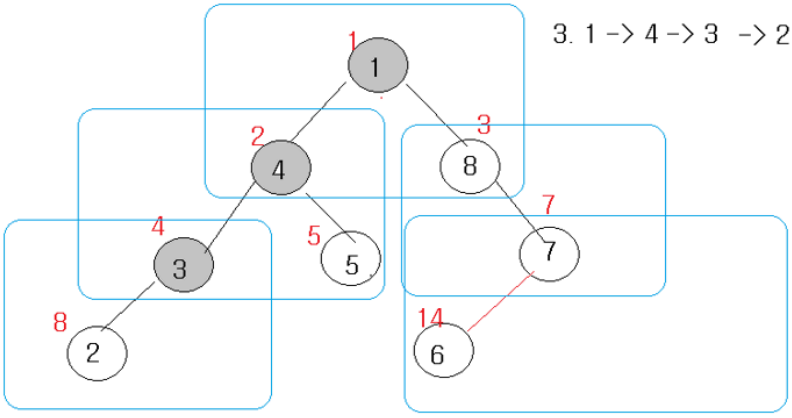
- **부모 → 왼쪽 → 오른쪽**



1. 루트 노드에서 시작한다.
(현재 노드를 부모 노드로 생각한다)
방문 순서는 1 -> 4 -> 8의 순서이다.



2. 노드 4로 이동한다.



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

중위 순회

- 왼쪽 → 부모 → 오른쪽

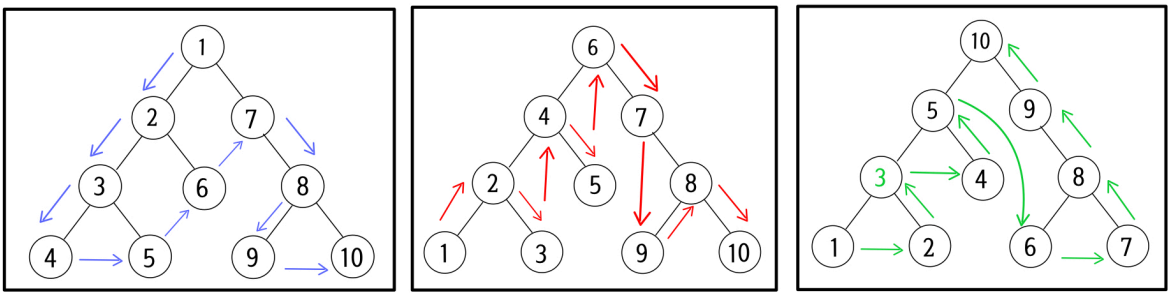
2 → 3 → 4 → 5 → 1 → 8 → 6 → 7

후위 순회

- 왼쪽 → 오른쪽 → 부모

2 → 3 → 5 → 4 → 6 → 7 → 8 → 1

방법 예시)

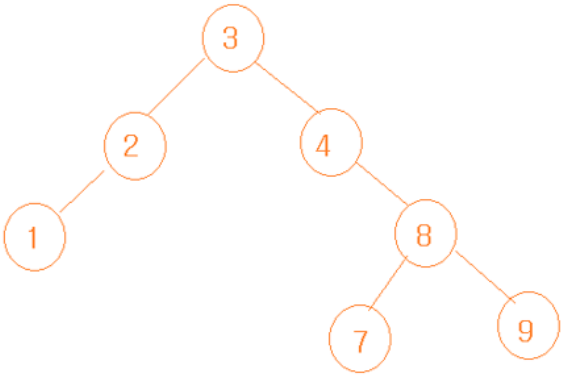


전위 순회 중위 순회 후위 순회

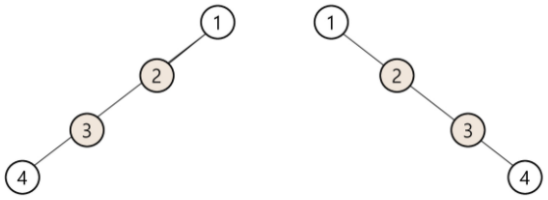
이진 트리 탐색

- 어떤 특정한 데이터를 탐색하기 위해서 사용
- 데이터를 효율적으로 탐색 할 수 있도록 구축하는 것
- 만약 데이터가 3 → 4 → 2 → 8 → 9 → 7 → 1 이 순서대로 들어온다고 가정하자
- **이진 탐색 트리를 구축하는데 있어 왼쪽, 오른쪽 자식의 위치의 배치는 특정한 규칙에 따라 정렬한다.**
  - 루트노드보다 들어오는 데이터 값이 크면 오른쪽, 작으면 왼쪽으로 배치한다.

이진 탐색 트리를 구축하는데 있어 왼쪽 , 오른쪽 자식의 위치의 배치는 특정한 규칙에 따라 정렬한다.



편향 이진 트리(왼쪽, 오른쪽):



이진 트리 순회 방법 함수로 구현:

전위

```
def order(nodes, idx):
    if idx < len(nodes):
        ret = str(nodes[idx]) + " "
        ret += order(nodes, idx * 2 + 1)
        ret += order(nodes, idx * 2 + 2)
        return ret
    else:
        return ""
```

중위

```
def inorder(nodes, idx):
    if idx < len(nodes):
        ret = inorder(nodes, idx * 2 + 1)
        ret += str(nodes[idx]) + " "
        ret += inorder(nodes, idx * 2 + 2)
        return ret
    else:
        return " "
```

후위 순회 및 다른 순회 코드화:

binary tree search.ipynb

Binary search tree 문제:

binary tree search pbs.ipynb