09-02, 03 트라이 & 이진트리

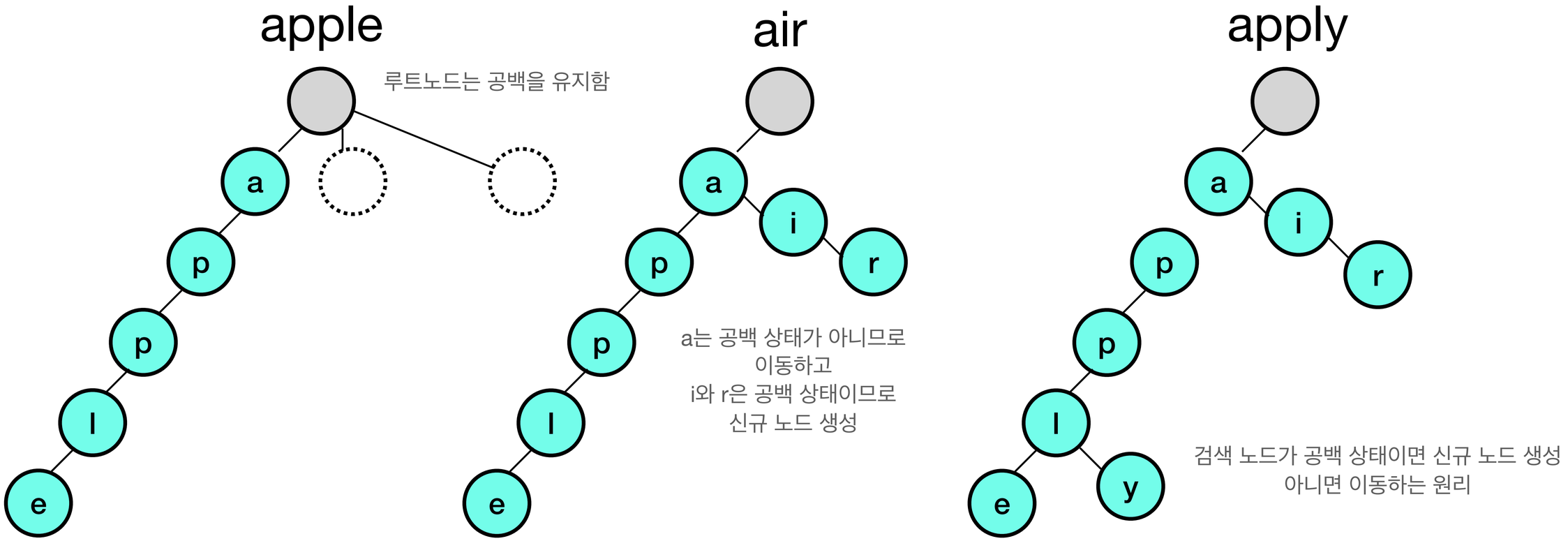
23.10.03 박송이

트라이란?

- 문자열 검색에 특화되어 검색을 빠르게 실행할 수 있도록 설계된 트리 형태의 일종
- 단어들을 사전의 형태로 생성한 후 트리의 부모 자식 노드 관계를 이용해 검색 수행
- N진 트리 : 문자 종류의 개수에 따라 N이 결정된다. ex) 알파벳 26개의 문자 -> 26진 트리
- 루트 노드는 항상 빈 문자열을 뜻하는 공백 상태를 유지한다.

트라이에 단어 삽입하는 예제

apple , air, apply



apple의 각 알파벳에 해당하는 노드 생성

69) 문자열 찾기

- 총 n개의 문자열로 이뤄진 집합 s가 있을 때 입력으로 주어지는 m개의 문자열 중 집합 s에 포함돼 있는 것이 총 몇 개인지 구하는 프로그램 작성
- 1번째 줄에 문자열의 개수 n과 m이 주어진다.
- 그 다음 n개의 줄에는 집합 s에 포함돼 있는 문자열이 주어지고
- 그 다음 m개의 줄에는 검사해야 하는 문자열이 주어진다.

5 11
baekjoononlinejudge
startlink
codeplus
sundaycoding
codings
baekjoon
codeplus
codeminus
startling
starlink
sundaycoding
codings
codings
sondaycoding
startrink
icerink

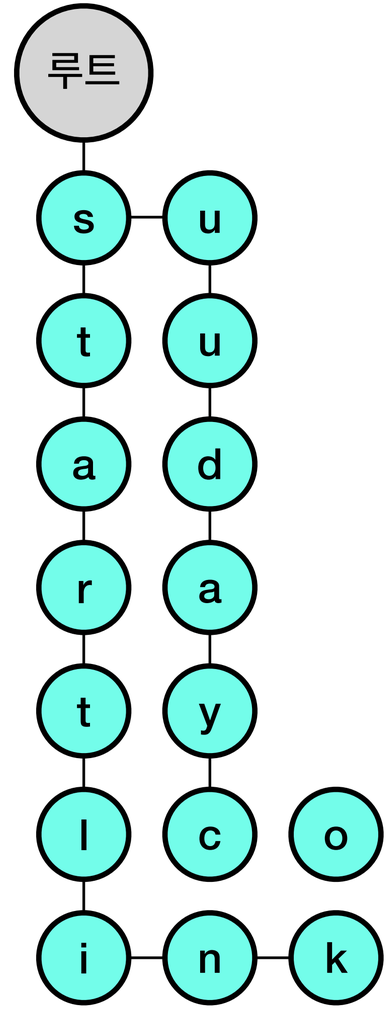
69) 문자열 찾기 - 트라이 자료 구조 생성

문자열의 마지막에 도달하면 리프 노드라고 표시

69) 문자열 찾기 - 코드

```python
class Node(object):
    def __init__(self,isEnd):
        self.isEnd = isEnd
        self.childNode = {}

class Trie(object):
    def __init__(self):
        self.parent = Node(None)

    def insert(self,string):


    def search(self,string):
```

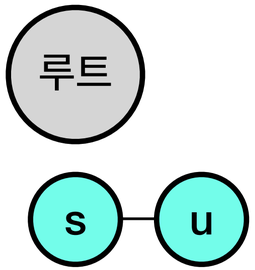
```python
N,M = map(int,input().split())
myTrie = Trie()

for _ in range(N):
    word = input().strip()
    myTrie.insert(word)

result = 0

for _ in range(M):
    word = input().strip()
    if myTrie.search(word):
        result +=1


print(result)
```



69) 문자열 찾기 - 코드

```
class Trie(object):
    def __init__(self):
        self.parent = Node(None)
```

루트노드는 공백으로 비워주기



string이 ‘coding’인 경우

```
    def insert(self,string):
        nowNode = self.parent
        temp_length = 0
        for char in string:
            if char not in nowNode.childNode:
                nowNode.childNode[char] = Node(char)
            nowNode = nowNode.childNode[char]
            temp_length+=1
            if temp_length== len(string):
                nowNode.isEnd = True
```

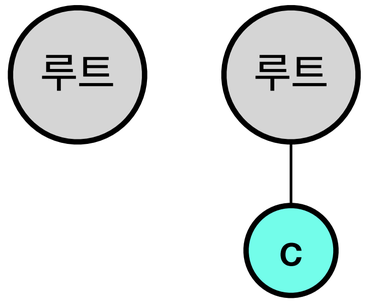
처음 노드는 공백인 루트노드

리프노드인지 표시하는 isEnd를 설정해주기 위해 필요한 변수

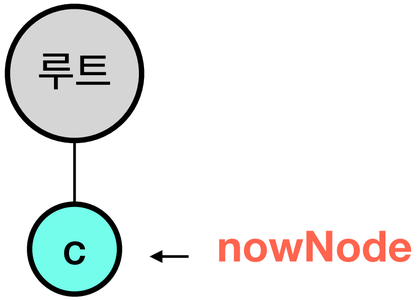
C O D I N G

검색 노드가 공백 상태이면 신규 노드 생성
아니면 이동하는 원리

신규 노드 생성



이동



69) 문자열 찾기 - 코드

```
    def search(self,string):
        nowNode = self.parent
        temp_length = 0
        for char in string:
            if char in nowNode.childNode:
                nowNode = nowNode.childNode[char]
                temp_length+=1
                if temp_length==len(string) and nowNode.isEnd==True:
                    return True
                else:
                    return False
            return False
```

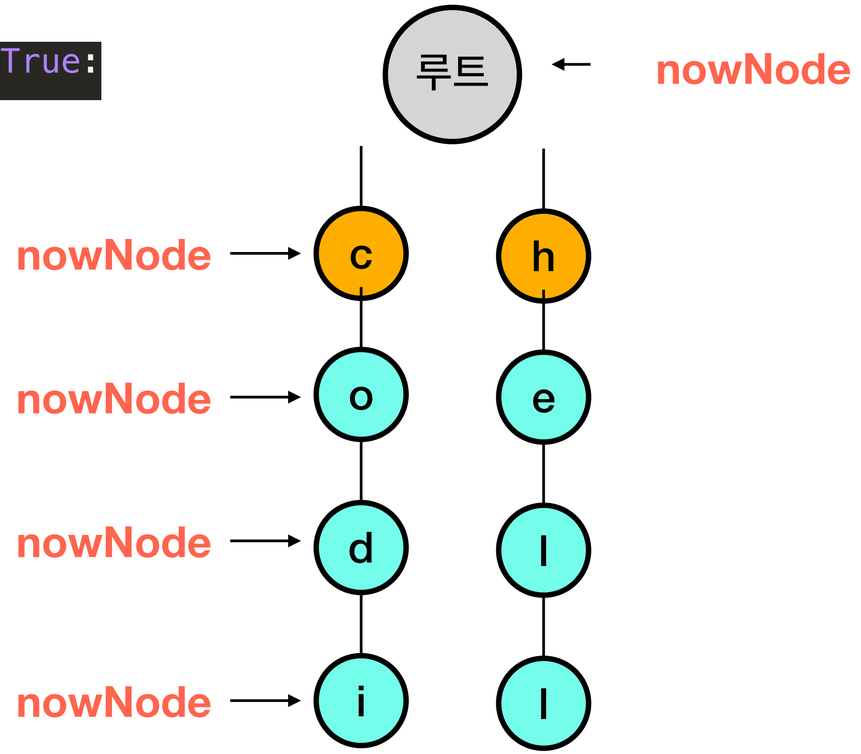
현재노드는 루트 노드로 설정

찾고자 하는 모든 문자에 대해 검색을 수행

현재 노드의 자식노드 중에 찾고자 하는 char가 있다면

현재 노드를 현재 노드의 자식노드로 바꾸어 준다.

CODING

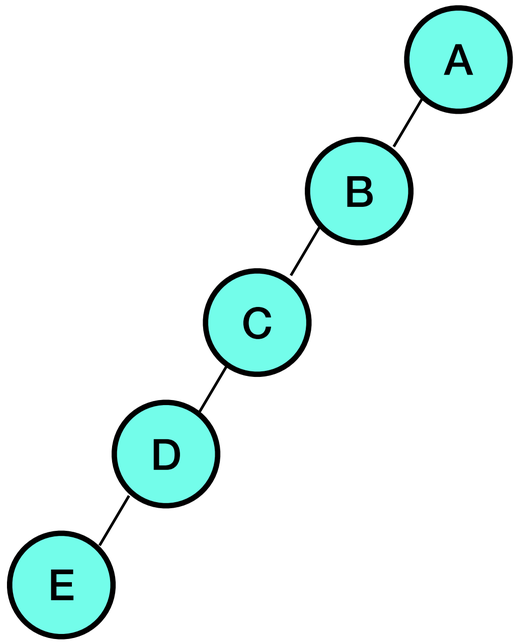


이진 트리란?

- 이진트리란 각 노드의 자식 노드(차수)의 개수가 2 이하로 구성돼 있는 트리
- 트리 영역에서 가장 많이 사용되는 형태

이진 트리의 종류

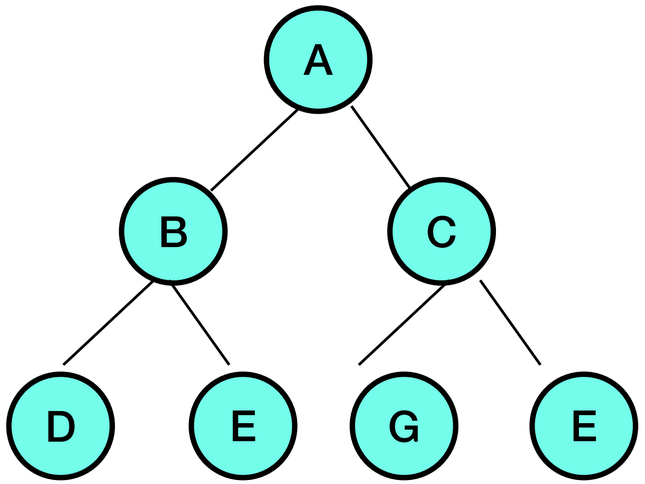
편향 이진 트리



노드들이 한쪽으로 편향되었음.

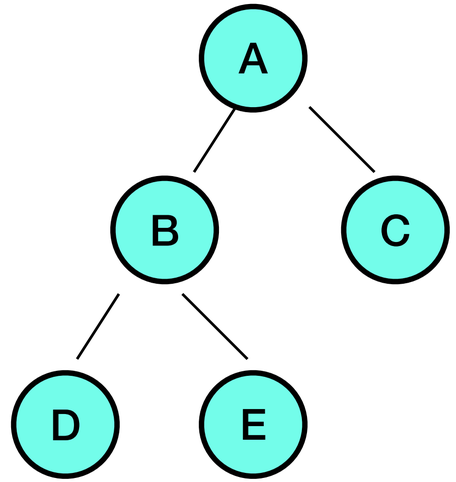
편향 이진트리로 저장하면 탐색 속도가 저하되고
공간이 낭비된다.

포화 이진 트리



트리의 높이가 모두 일정하며 리프 노드가 꽉찬 이진트리

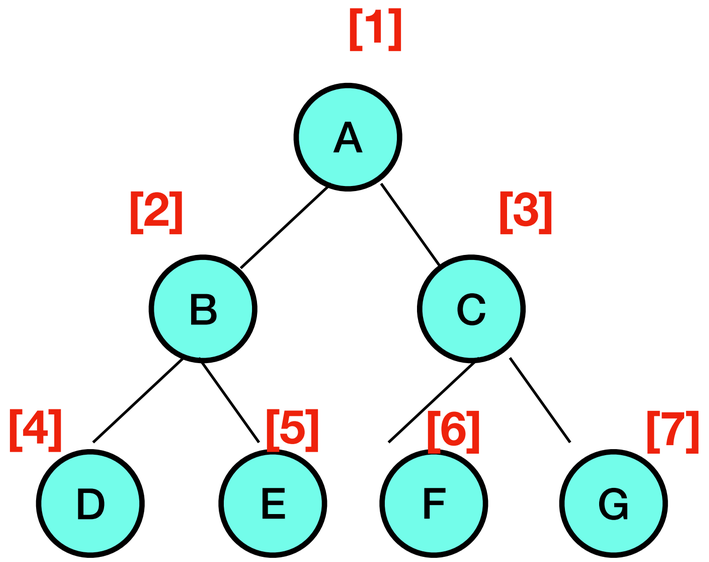
완전 이진 트리



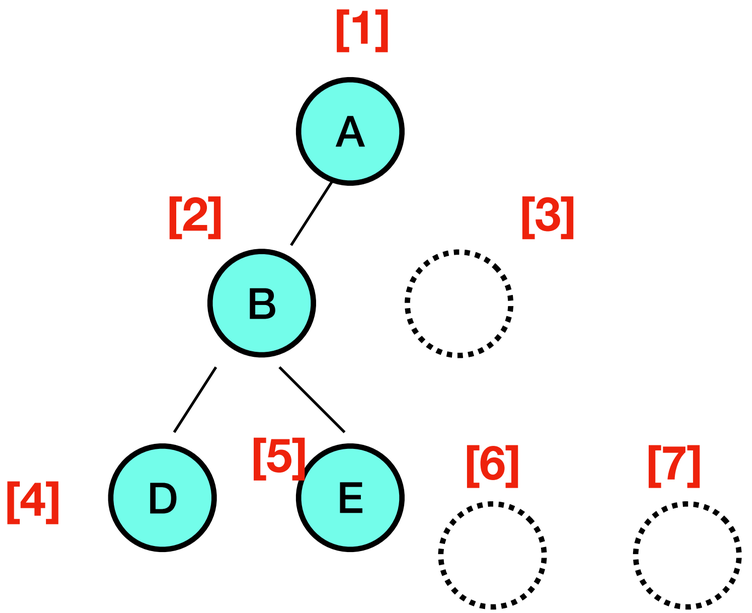
마지막 레벨을 제외하고 완전하게 노드들이
채워져 있고 마지막 레벨은 왼쪽부터 채워져 있음.

일반적으로 코딩 테스트에서 데이터를 트리에 담는다고 하면
완전 이진 트리 형태를 떠올리면 된다.

이진 트리의 순차표현



1	2	3	4	5	6	7
A	B	C	D	E	F	G



1	2	3	4	5	6	7
A	B		D	E	F	

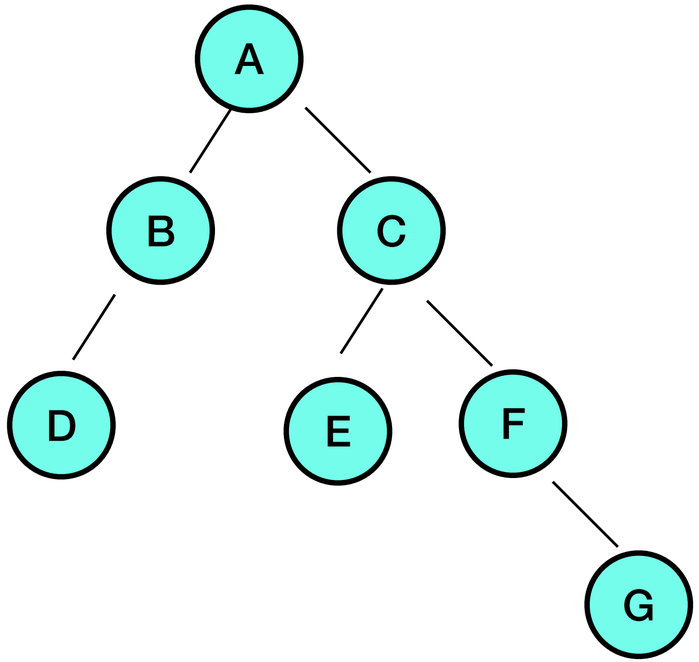
이진 트리의 순차표현

이동 목표 노드	인덱스 연산	제약 조건
루트노드	index = 1	
부모노드	index = index / 2	현재 노드가 루트 노드가 아님
왼쪽 자식 노드	index = index * 2	index * 2 <=N
오른쪽 자식 노드	index = index * 2 + 1	index * 2 + 1 <=N

위의 연산 방식은 세그먼트 트리나 LCA에서도 사용되므로 기억해두기!

70) 트리 순회하기 - 전위, 중위, 후위 순회 알아보기

- 이진 트리를 입력받아 전위 순회 , 중위 순회, 후위 순회한 결과를 출력하라.



전위 순회 **A-> BD-> (C->E->FG)** 루트 -> 왼쪽 자식-> 오른쪽 자식

중위 순회 **DB-> A->(E->C->FG)** 왼쪽자식->루트->오른쪽자식

후위 순회 **DB->EGFC->A** 왼쪽자식->오른쪽자식->루트

70) 트리 순회하기

- 이진 트리를 입력받아 전위 순회 , 중위 순회, 후위 순회한 결과를 출력하라.

- 이진 트리 노드의 개수 N이 첫번째 줄에 주어진다.
- 2번째 줄부터 N개의 줄에 걸쳐 각 노드와 그의 왼쪽 자식, 오른쪽 자식 노드가 주어진다.
- 노드의 이름은 A부터 차례대로 영문자 대문자로 매겨진다.
- 항상 A가 루트 노드가 된다.
- 자식 노드가 없을때는 .으로 표현된다.
- 출력은 1번째 줄에 전위 순회 , 2번째 줄에 중위 순회, 3번째 줄에 후위 순회한 결과 출력

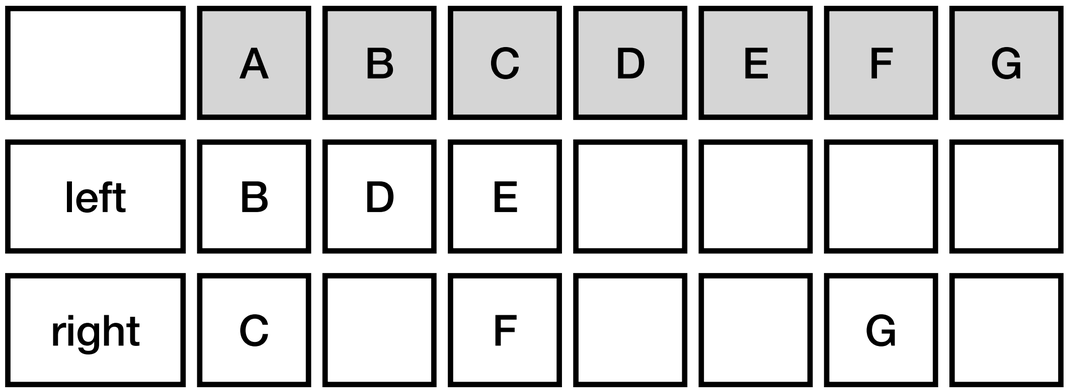
```
7
A B C
B D .
C E F
E . .
F . G
D . .
G . .
```

```
ABDCEFG
DBAECFG
DBEGFCA
```

70) 트리 순회하기

- 딕셔너리 자료형에 저장하기

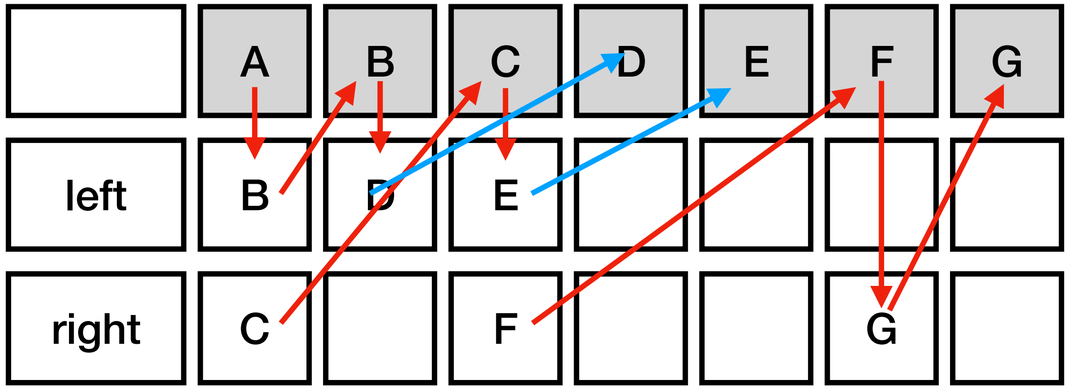
```
7
A B C
B D .
C E F
E . .
F . G
D . .
G . .
```

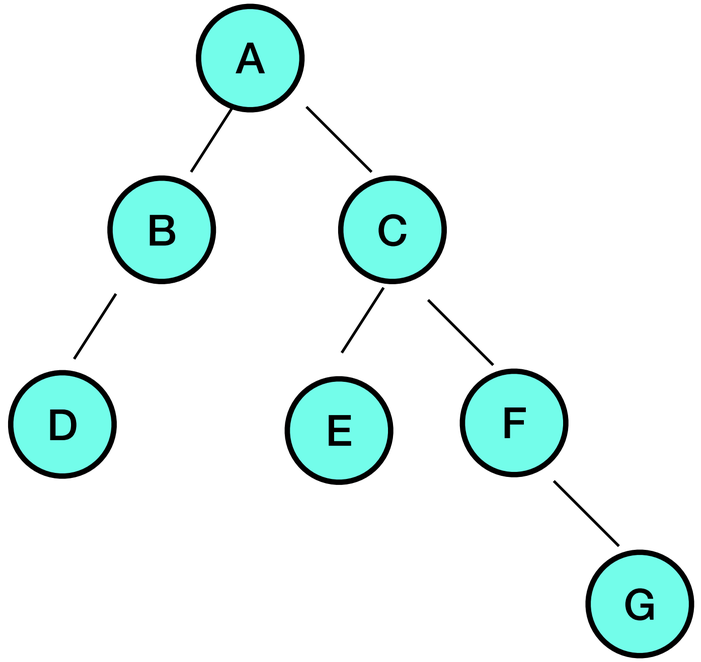


70) 트리 순회하기

- 전위 순회 함수 구현하기

현재 노드 -> 왼쪽 노드 -> 오른쪽 노드 순서대로 탐색

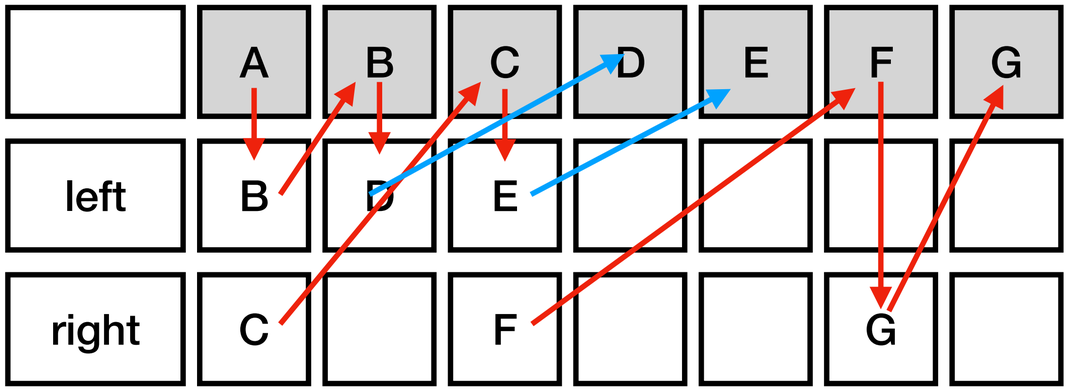




(A -> B -> D) -> C -> E -> F -> G

70) 트리 순회하기

- sudo 코드로 보기



preOrder

if 현재값 == '.':
return
1. 현재 노드 출력
2. 왼쪽 자식 노드 탐색
3. 오른쪽 자식 노드 탐색

inOrder

if 현재값 == '.':
return
1. 왼쪽 자식 노드 탐색
2. 현재 노드 출력
3. 오른쪽 자식 노드 탐색

postOrder

if 현재값 == '.':
return
1. 왼쪽 자식 노드 탐색
2. 오른쪽 자식 노드 탐색
3. 현재 노드 출력

70) 트리 순회하기

- 코드로 이해하기

데이터 구조

```
tree = {}
root, left, right = input().split()
tree[root] = [left,right]
```

preOrder

if 현재값 == '.':
return
1. 현재 노드 출력
2. 왼쪽 자식 노드 탐색
3. 오른쪽 자식 노드 탐색

```
def preOrder(now):
if now =='.':
    return
print(now,end=' ')
preOrder(tree[now][0]
preOrder(tree[now][1]
```

70) 트리 순회하기

- 코드로 이해하기

inOrder

if 현재값 == '.':
return
1. 왼쪽 자식 노드 탐색
2. 현재 노드 출력
3. 오른쪽 자식 노드 탐색

```
def inOrder(now):
if now ==‘.’:
    return
inOrder(tree[now][0]
print(now,end=‘ ‘)
inOrder(tree[now][1]
```

70) 트리 순회하기

- 코드로 이해하기

postOrder

if 현재값 == ‘.’:
return
1. 왼쪽 자식 노드 탐색
2. 오른쪽 자식 노드 탐색
3. 현재 노드 출력

```
def postOrder(now):
if now ==‘.’:
    return
postOrder(tree[now][0]
postOrder(tree[now][1]
print(now,end=‘ ‘)
```

최종 결과 출력

preOrder(‘A’)

print()

inOrder(‘A’)

print()

PostOrder(‘A)