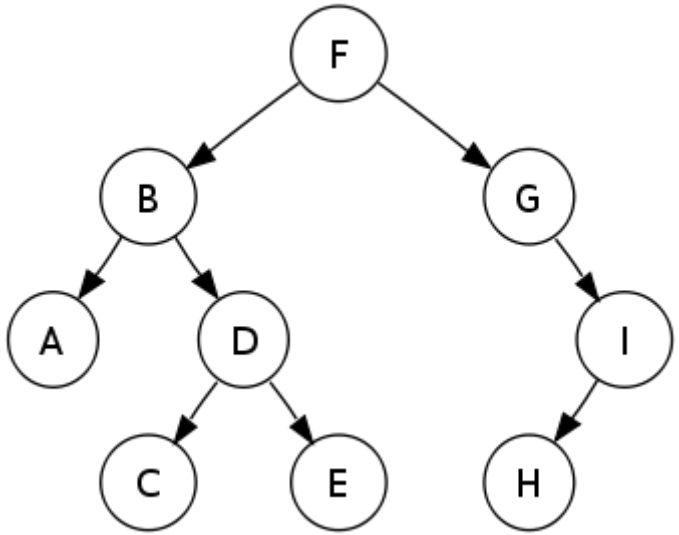
트리 I : 기초~순회

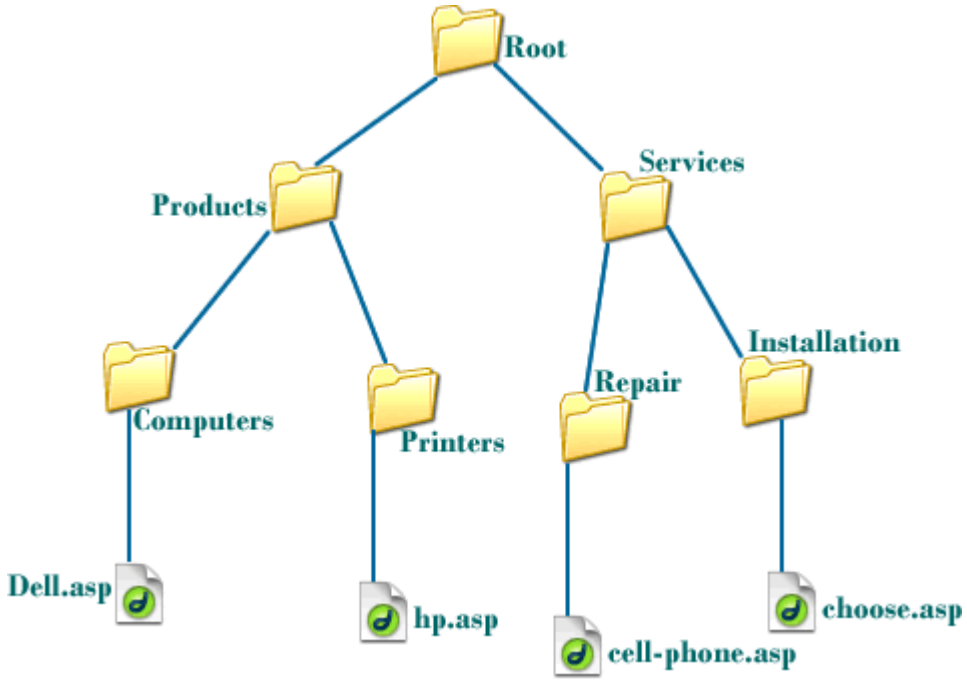
트리 정의

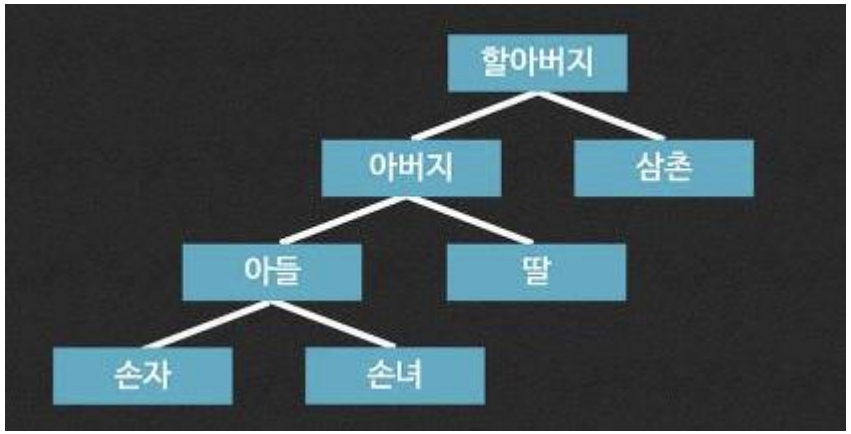
트리는 계층형 트리 구조를 시뮬레이션 하는 추상 자료형(ADT)으로, 루트 값과 부모-자식 관계의 서브트리로 구성되며, 서로 연결된 노드의 집합이다.(나무를 거꾸로 뒤집은 모양..)



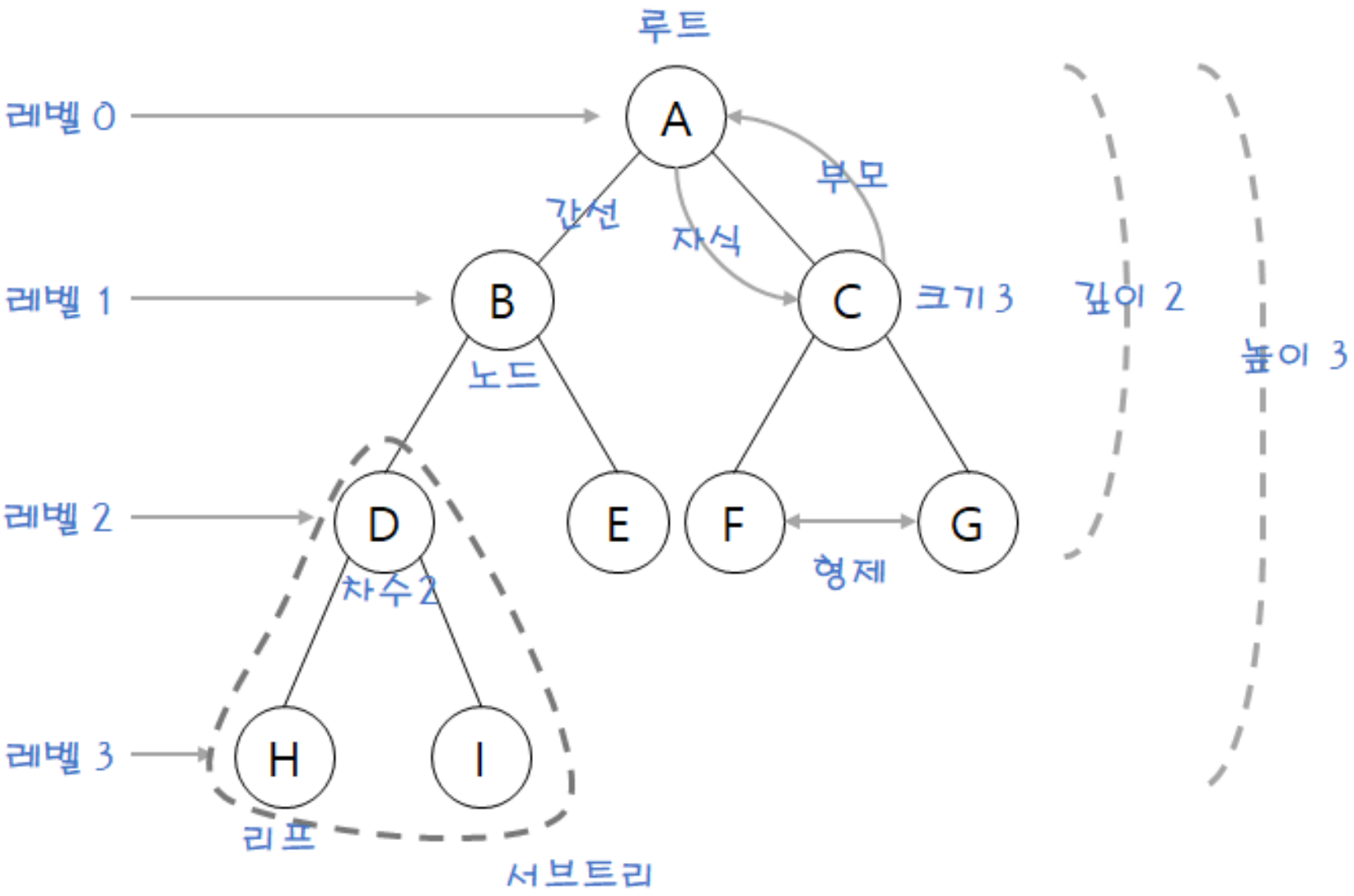
트리 정의(2)

Ex) 가계도, 회사의 조직도, 컴퓨터의 디렉토리





트리와 관련된 용어



트리는 항상 root 노드에서 시작.

Root 노드-> 부모가 없는 노드, 트리는 하나의 루트 노드만을 가진다.

Reaf 노드 -> 자식이 없는 노드.

간선(edge) -> 노드를 연결하는 선(단방향)

형제(sibling) -> 같은 부모를 가지는 노드

노드의 크기(size)
-> 자신을 포함한 모든 자식 노드의 개수

노드의 깊이(depth)
->루트에서 어떤 노드에 도달하기 위해 거쳐야 하는 간선의 수

노드의 레벨(level)
->트리의 특정 깊이를 가지는 노드의 집합노드의 차수(degree) ->하위 트리 개수 / 간선 수 (degree)
= 각 노드가 지닌 가지의 수

트리의 높이(height)-> 루트 노드에서 리프 노드까지의 거리

트리 자료구조에서 알면 좋은 점(1)

트리 자료구조에서 알면 좋은 속성

1. 재귀로 정의된 자기 참조형의 자료구조라는 점입니다.

트리는 자식도 트리, 또 그 자식도 트리로 구성되어 있고, 여러 개의 트리가 쌓아 올려져 큰 트리가 됩니다. 흔히 subtree로 구성된다고 표현합니다. 이러한 재귀적 특성 때문에 트리를 순회할 때에도 재귀 순회를 사용하는 편입니다.

트리 자료구조에서 알면 좋은 점(2)

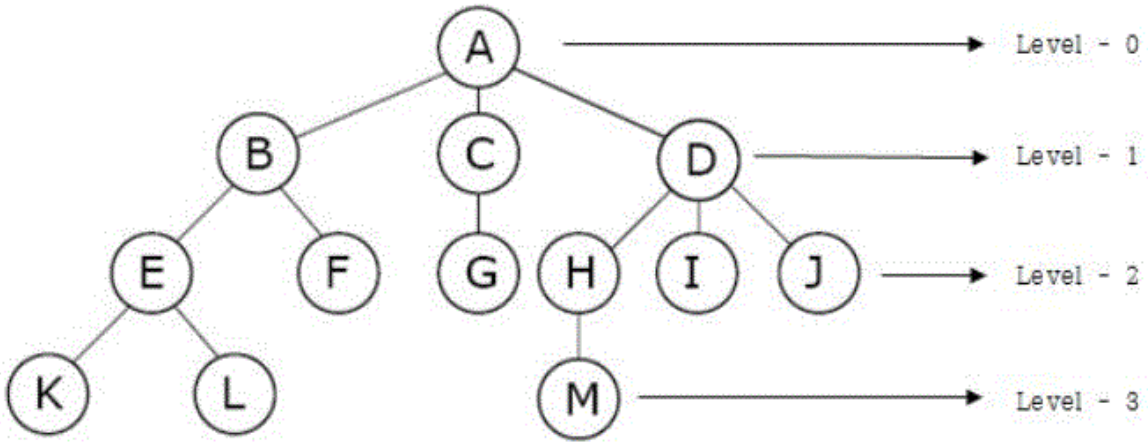
2. 노드가 N개인 트리는 항상 N-1개의 간선(edge)을 가진다.즉, 간선은 항상 (정점의 개수 - 1) 만큼을 가집니다.

3. 트리는 DAG(Directed Acyclic Graphs, 방향성이 있는 비순환 그래프)의 한 종류이다. 즉 '순환 구조를 갖지 않는 그래프'이다.
->그래프와 트리의 차이점

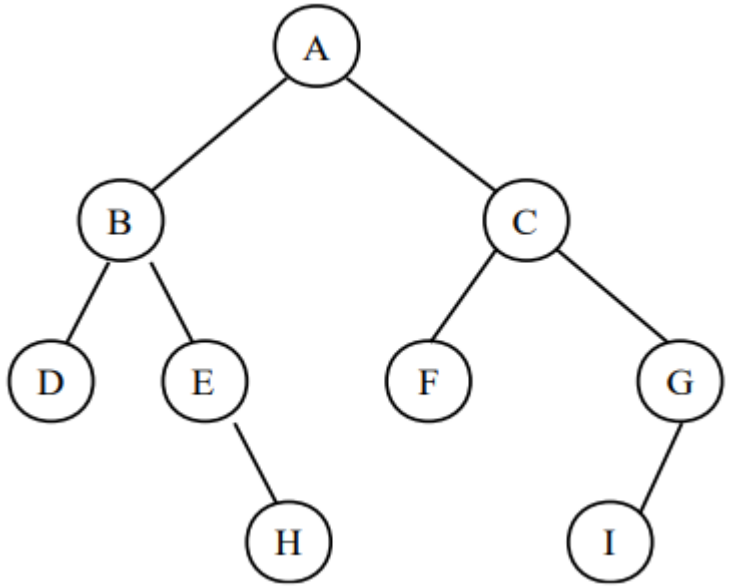
4. 루트에서 어떤 노드로 가는 경로는 유일하다.임의의 두 노드 간의 경로도 유일하다. 즉, 두 개의 정점 사이에 반드시 1개의 경로만을 가진다.

이진 트리

이진트리 ->모든 노드의 차수(degree)가 2이하인 트리



M-ary 트리(다항 트리)



이진트리(binary tree)

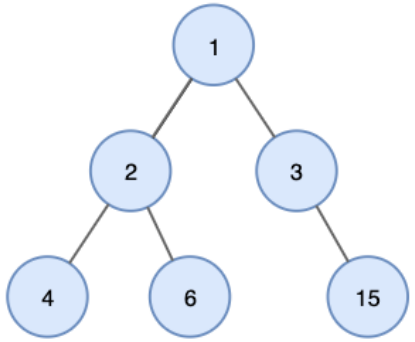
이진 트리의 유형

1. Full Binary Tree(정 이진 트리)
-> 모든 노드가 0개 or 2개의 자식 노드를 갖는다.

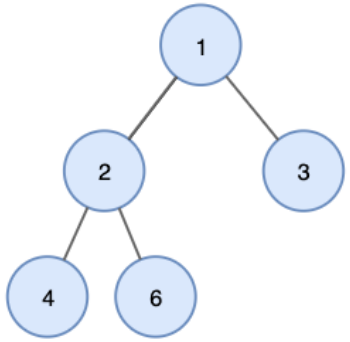
2. Complete Binary Tree(완전 이진 트리)
-> 마지막 레벨을 제외하고 모든 레벨이 완전히 채워져 있음. 또한 마지막 레벨의 모든 노드는 가장 왼쪽부터 채워져 있다.

3. Perfect Binary Tree(포화 이진 트리)
-> 모든 노드가 2개의 자식 노드를 갖고 있으며 모든 리프 노드가 동일한 깊이 또는 레벨을 갖는다. 문자 그대로 가장 완벽한 유형의 트리.

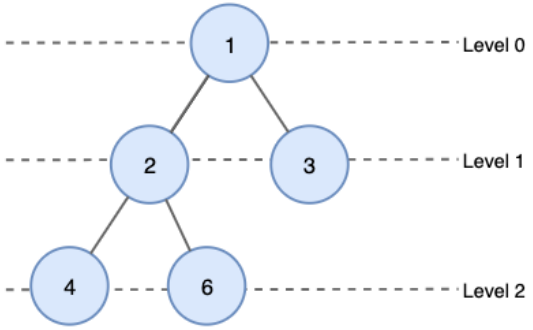
이진 트리의 유형(2)



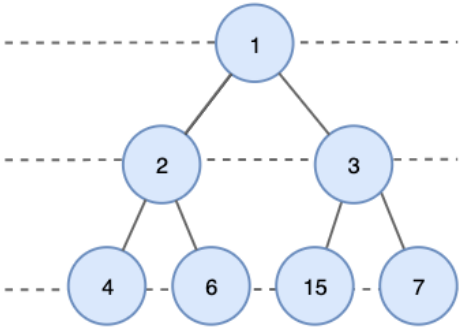
전 이진 트리가 아님
노드 3이 하나의 자식 노드를 갖기 때문



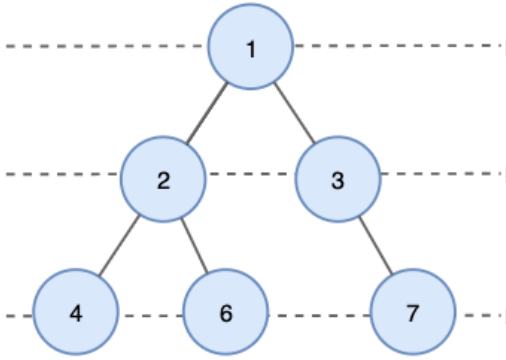
전 이진 트리가 맞음



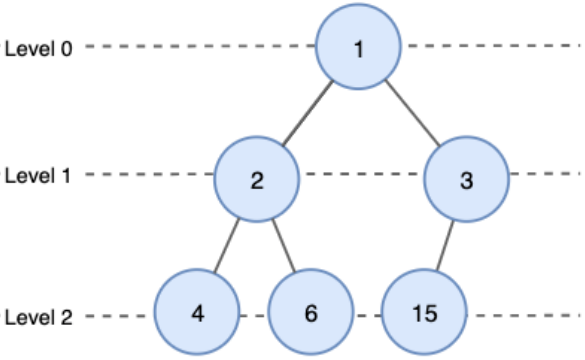
포화 이진 트리가 아님
노드 마다 두 명의 자식을 가지고 있지만 모든 잎 노드가 같은 Level 이 아님
잎 노드 3: Level 1
잎 노드 4, 6 : Level 2



포화 이진 트리가 맞음



완전 이진 트리가 아님
Level 2에서 왼쪽 노드부터 채워지지 않았기 때문



완전 이진 트리가 맞음

트리 순회

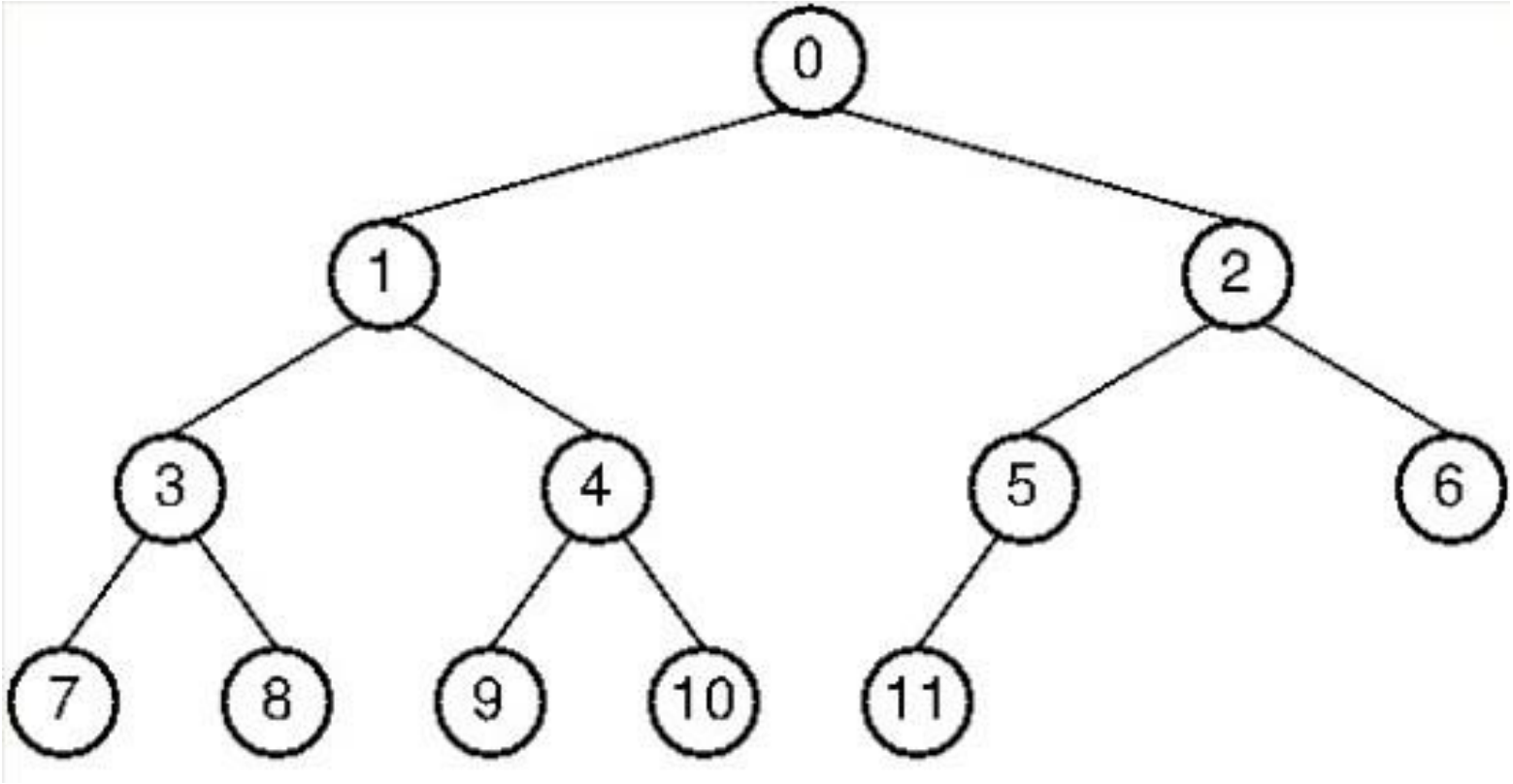
트리의 모든 노드들을 방문하는 과정을 트리 순회(TreeTraversal)라고 합니다.
선형 자료 구조(연결 리스트, 스택, 큐 등)는 순차적으로 요소에 접근하지만 트리 자료구조는 다른 방식을 사용해야 합니다.

일반적으로 트리 순회에는 다음과 같은 방법들이 있습니다.
- 전위 순회 (Preorder)
- 중위 순회 (Inorder)
- 후위 순회 (Postorder)

이러한 순회는 보통 재귀로 쉽게 구현할 수 있습니다.

트리 순회(1) - 전위 순회(preorder traversal)

뿌리를 먼저 방문한다. 뿌리->왼쪽 자식 -> 오른쪽 자식



0->1->3->7->8->4->9->10->2->5->11->6

트리 순회(1) - 전위 순회(preorder traversal) 구현

```
preorder(node)
  print node.value
  if node.left ≠ null then preorder(node.left)
  if node.right ≠ null then preorder(node.right)
```

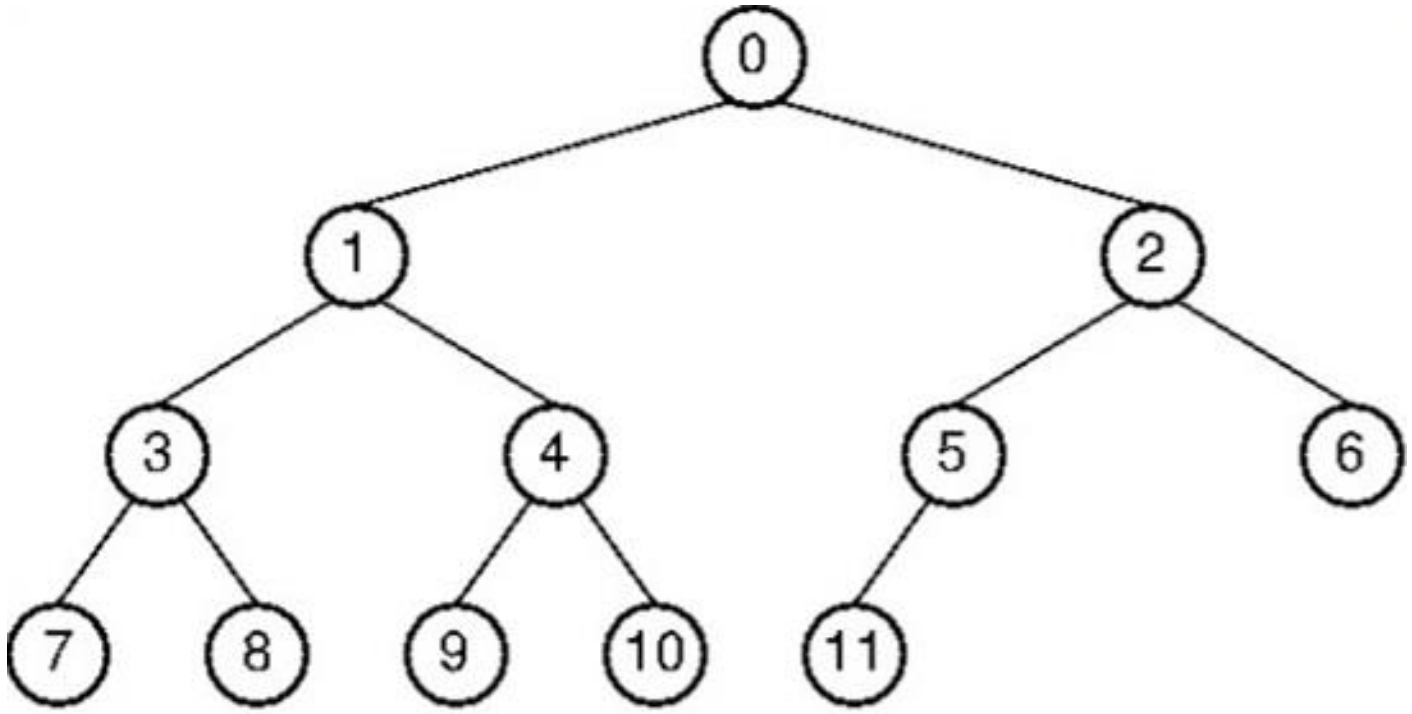
```java
void preOrder(TraversalNode node){
    if(node != null){
        System.out.println(node.data);
        preOrder(node.leftNode);
        preOrder(node.rightNode);
    }
}
```

```cpp
void preorder(nodePointer pointer){
    if(pointer){
        // pointer가 null이 아니라면
        cout << pointer->data << '';
        preorder(pointer->left);
        preorder(pointer->right);
    }
}
```

```python
def preorder_traverse(node):

    if node == None: return
    print(node.data, end = ' -> ')
    preorder_traverse(node.left)
    preorder_traverse(node.right)
```

트리 순회(2) - 중위 순회(in-order traversal)

왼쪽 자식을 먼저 방문한 후 뿌리를 방문.
왼쪽 자식 -> 뿌리 -> 오른쪽 자식



7->3->8->1->9->4->10->0->11->5->2->6

트리 순회(2) - 중위 순회(in-order traversal)

```
inorder(node)
  if node.left  ≠ null then inorder(node.left)
  print node.value
  if node.right ≠ null then inorder(node.right)
```

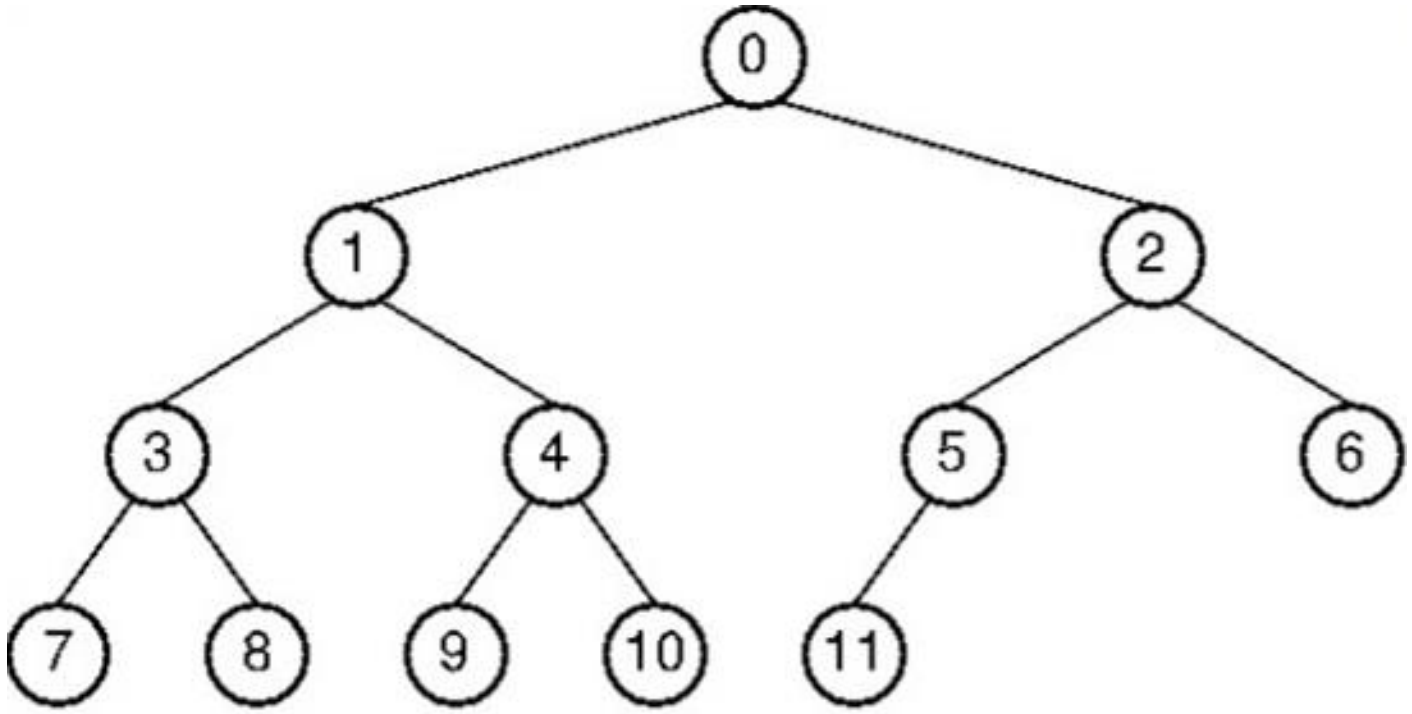
```java
void inOrder(TraversalNode node){

    if (node != null){
        inOrder(node.leftNode);
        System.out.println(node.data);
        inOrder(node.rightNode);
    }
}
```

```python
def inorder_traverse(node):
    if node == None: return
    inorder_traverse(node.left)
    print(node.data, end='->')
    inorder_traverse(node.right)
```

```cpp
void inorder(nodePointer pointer){
    if(pointer){
        // pointer가 null이 아니라면
        inorder(pointer->left);
        cout << pointer->data << '';
        inorder(pointer->right);
    }
}
```

트리 순회(3) - 후위 순회(postorder traversal)

왼쪽 자식과 오른쪽 자식 방문 후 뿌리를 방문.
왼쪽 자식 -> 오른쪽 자식 -> 뿌리



7 -> 8 -> 3 -> 9-
>10->4->1->11->
5->6->2->0

트리 순회(3) - 후위 순회(postorder traversal)

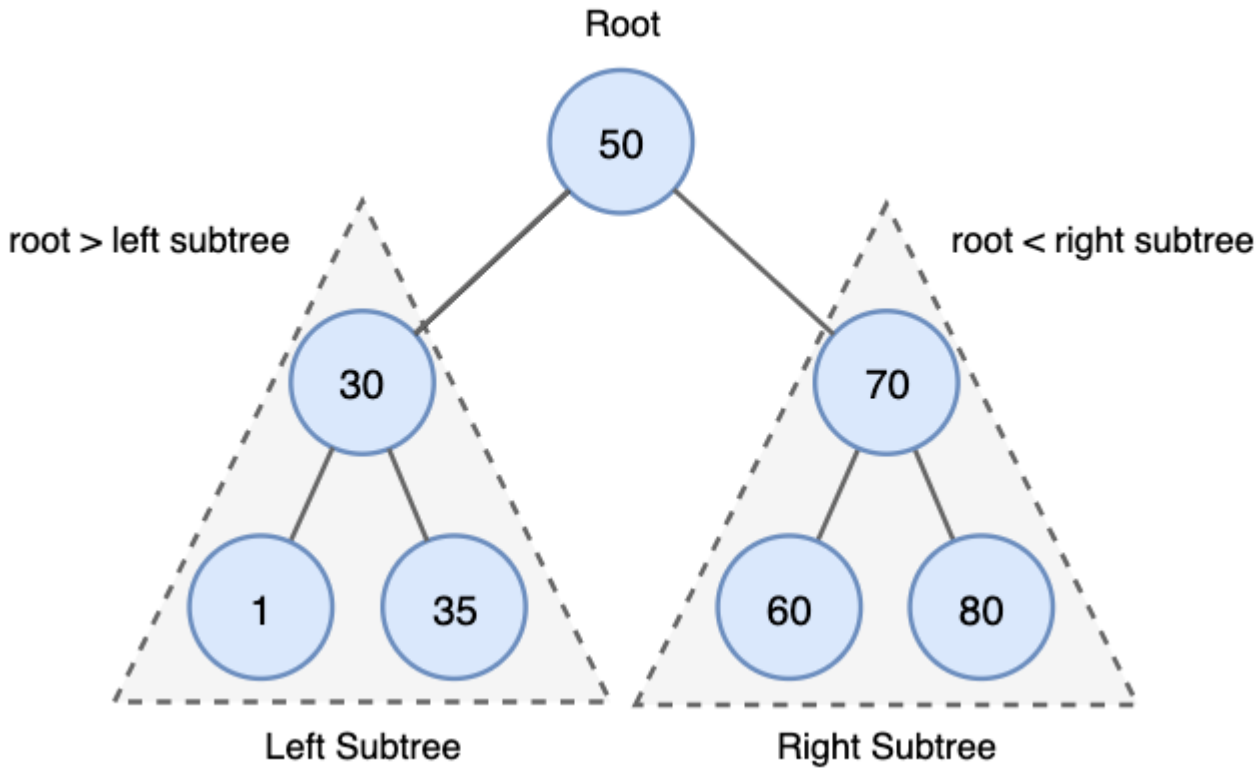
```
postorder(node)
  if node.left  ≠ null then postorder(node.left)
  if node.right ≠ null then postorder(node.right)
  print node.value
```

```java
void postOrder(TraversalNode node){
    if(node != null){
        postOrder(node.leftNode);
        postOrder(node.rightNode);
        System.out.println(node.data);
    }
}
```

```cpp
void postorder(nodePointer pointer){
    if(pointer){
        // pointer가 null이 아니라면
        postorder(pointer->left);
        postorder(pointer->right);
        cout << pointer->data << '';
    }
}
```

```python
def postorder_traverse(node):
    if node == None: return
    postorder_traverse(node.left)
    postorder_traverse(node.right)
    print(node.data, end = '->')
```

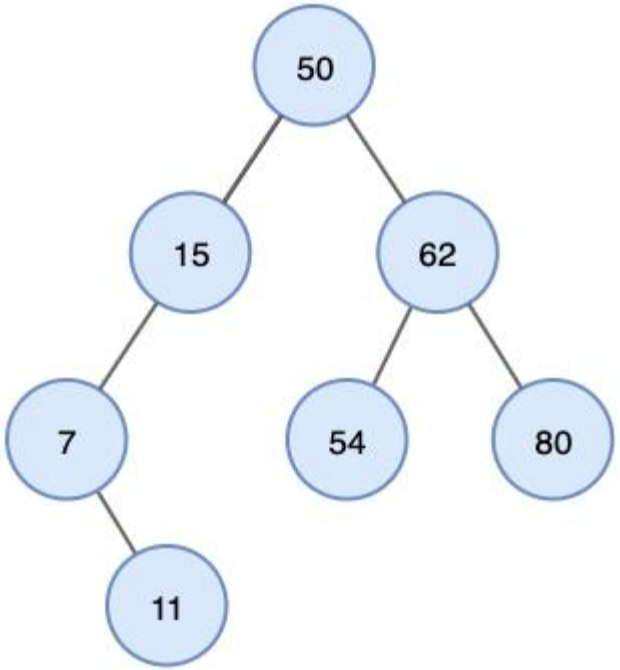
이진 탐색 트리

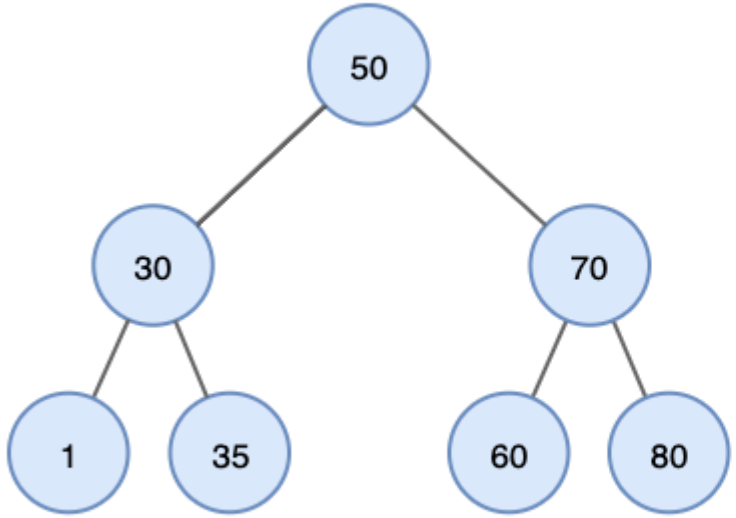


이진 탐색 트리란 정렬된 이진트리로써 다음과 같은 속성을 가지고 있습니다.

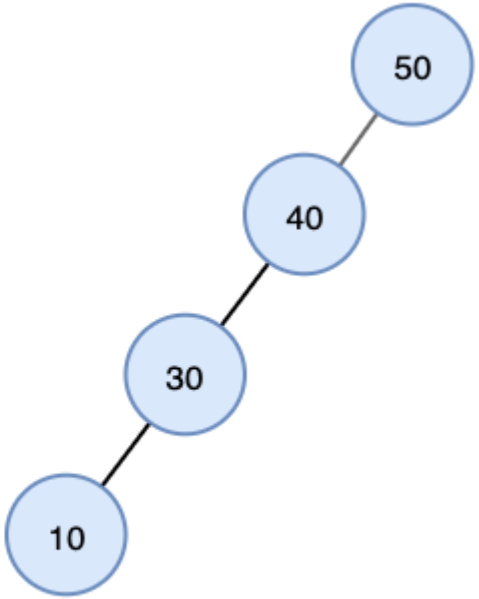
- 노드의 왼쪽 하위 트리에는 노드의 키보다 작은 키가있는 노드 만 포함됩니다
- 노드의 오른쪽 하위 트리에는 노드의 키보다 큰 키가있는 노드 만 포함됩니다.
- 왼쪽 및 오른쪽 하위 트리도 각각 이진 검색 트리 여야합니다.
- 중복된 키를 허용하지 않습니다.

이진 탐색 트리의 특징





Time Complexity = O(LogN)



Time Complexity = O(N)

1. BST의 Inorder Traversal을 수행하여 모든 키를 정렬된 순서로 가져올 수 있습니다.

2. BST의 검색에 대한 시간복잡도는 균형 상태이면 O(logN)의 시간이 걸리고 불균형 상태라면 최대 O(N) 시간이 걸립니다.

문제풀이

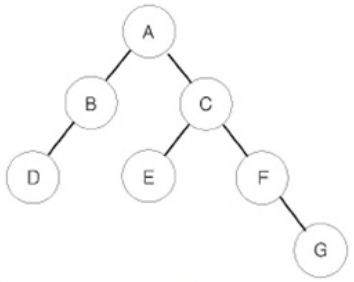
문제 1) 트리 순회(1991)

트리 순회

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	34708	22435	17073	65.962%

문제

이진 트리를 입력받아 전위 순회(preorder traversal), 중위 순회(inorder traversal), 후위 순회(postorder traversal)한 결과를 출력하는 프로그램을 작성하시오.



예를 들어 위와 같은 이진 트리가 입력되면,

- 전위 순회한 결과 : ABDCEFG // (루트) (왼쪽 자식) (오른쪽 자식)
- 중위 순회한 결과 : DBAECFG // (왼쪽 자식) (루트) (오른쪽 자식)
- 후위 순회한 결과 : DBEGFCA // (왼쪽 자식) (오른쪽 자식) (루트)

가 된다.

입력

첫째 줄에는 이진 트리의 노드의 개수 N(1 ≤ N ≤ 26)이 주어진다. 둘째 줄부터 N개의 줄에 걸쳐 각 노드와 그의 왼쪽 자식 노드, 오른쪽 자식 노드가 주어진다. 노드의 이름은 A부터 차례대로 알파벳 대문자로 매겨지며, 항상 A가 루트 노드가 된다. 자식 노드가 없는 경우에는 .으로 표현한다.

출력

첫째 줄에 전위 순회, 둘째 줄에 중위 순회, 셋째 줄에 후위 순회한 결과를 출력한다. 각 줄에 N개의 알파벳을 공백 없이 출력하면 된다.

예제 입력 1 복사

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

예제 출력 1 복사

```
ABDCEFG
DBAECFG
DBEGFCA
```

문제 1) 트리 순회(1991)(2)

문자열을 트리 ADT로 변형 -> 순회 함수 구현 in python

```
import sys

n = int(sys.stdin.readline())
lst = [list(sys.stdin.readline().split()) for _ in range(n)]


class node():
    def __init__(self, item, left, right):
        self.item = item
        self.left = left
        self.right = right
```

```
tree = {}
for item, left, right in lst:
    tree[item] = node(item, left, right)
```

```
def preorder(node):
    print(node.item, end='')
    if node.left != '.':
        preorder(tree[node.left])
    if node.right != '.':
        preorder(tree[node.right])
```

```
def inorder(node):
    if node.left != '.':
        inorder(tree[node.left])
    print(node.item, end='')
    if node.right != '.':
        inorder(tree[node.right])
```

```
def postorder(node):
    if node.left != '.':
        postorder(tree[node.left])
    if node.right != '.':
        postorder(tree[node.right])
    print(node.item, end='')
```

마지막 출력 부분

```
preorder(tree['A'])
print()
inorder(tree['A'])
print()
postorder(tree['A'])
```

문제 1) 트리 순회(1991)(3)

문자열을 트리 ADT로 변형 -> 순회 함수 구현 in java

```
import java.util.*;

class Node{
	char Data;
	Node Left, Right;
	public Node(char Data) {
		this.Data = Data;
	}
}

class Tree{
	Node Root;

	public void Add(char Data, char Left_Data, char Right_Data) {
		if(Root==null) {
			//Data가 . 이 아니라면 루트에 Data 값을 가진 노드 생성
			if(Data!='.') Root = new Node(Data);
			//왼쪽 자식 노드 데이터가 . 이 아니라면 Left_Data를 가진 왼쪽 자식 노드 생성
			if(Left_Data!='.') Root.Left = new Node(Left_Data);
			//오른쪽 자식 노드 데이터가 . 이 아니라면 Right_Data를 가진 오른쪽 자식 노드 생성
			if(Right_Data!='.') Root.Right = new Node(Right_Data);
		}
		//루트가 널이 아니라면 탐색
		else Search(Root, Data, Left_Data, Right_Data);
	}

	public void Search(Node Root, char Data, char Left_Data, char Right_Data) {
		//만약에 루트노드가 null이면 종료
		if(Root==null) return;

		//루트노드의 데이터가 Data라면
		else if(Root.Data==Data) {
			if(Left_Data!='.') Root.Left = new Node(Left_Data);
			if(Right_Data!='.') Root.Right = new Node(Right_Data);
		}
		//아니라면, 못찾았다면
		else {
			Search(Root.Left, Data, Left_Data, Right_Data);
			Search(Root.Right, Data, Left_Data, Right_Data);
		}
	}
```

```
public void PreOrder(Node Root) {
	//루트 -> 왼쪽 자식 -> 오른쪽 자식
	//먼저 루트노드 출력
	System.out.print(Root.Data);

	if(Root.Left!=null) PreOrder(Root.Left);

	if(Root.Right!=null) PreOrder(Root.Right);
}

public void InOrder(Node Root) {
	//왼쪽 자식 -> 루트 -> 오른쪽 자식

	if(Root.Left!=null) InOrder(Root.Left);
	//루트노드 출력
	System.out.print(Root.Data);

	if(Root.Right!=null) InOrder(Root.Right);
}

public void PostOrder(Node Root) {
	//왼쪽 자식 -> 오른쪽 자식 -> 루트

	if(Root.Left!=null) PostOrder(Root.Left);

	if(Root.Right!=null) PostOrder(Root.Right);

	//루트노드 출력
	System.out.print(Root.Data);


}
```

```
public class Main {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int N = sc.nextInt(); //트리노드의 개수

		Tree tree = new Tree();
		for(int i=0;i<N;i++) {
			tree.Add(sc.next().charAt(0),sc.next().charAt(0),sc.next().charAt
		}

		tree.PreOrder(tree.Root);
		System.out.println();
		tree.InOrder(tree.Root);
		System.out.println();
		tree.PostOrder(tree.Root);
	}
}
```

문제 1) 트리 순회(1991)(4)

문자열을 트리 ADT로 변형 -> 순회 함수 구현 in c++

```
void preOrder(char root)
{
    if (root == '.')
        return;
    else {
        cout << root;
        preOrder(arr[root].left);
        preOrder(arr[root].right);
    }
}

void inOrder(char root)
{
    if (root == '.')
        return;
    else {
        inOrder(arr[root].left);
        cout << root;
        inOrder(arr[root].right);
    }
}

void postOrder(char root)
{
    if (root == '.')
        return;
    else {
        postOrder(arr[root].left);
        postOrder(arr[root].right);
        cout << root;
    }
}
```

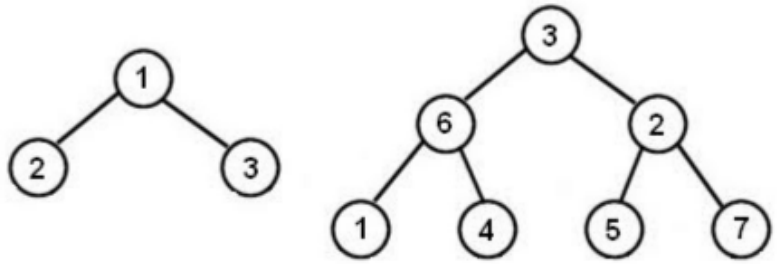
```
struct node
{
    char left;
    char right;
};

struct node arr[100];
```

```
int main() {
    std::ios::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);

    cin >> n;

    char t1, t2, t3;

    for (int i = 1; i <= n; i++)
    {
        cin >> t1 >> t2 >> t3;
        arr[t1].left = t2;
        arr[t1].right = t3;
    }

    preOrder('A');
    cout << "\n";
    inOrder('A');
    cout << "\n";
    postOrder('A');
    cout << "\n";

}
```

문제 2) 완전 이진 트리(9934)

문제

상근이는 슬로베니아의 도시 Donji Andrijevci를 여행하고 있다. 이 도시의 도로는 깊이가 K인 완전 이진 트리를 이루고 있다. 깊이가 K인 완전 이진 트리는 총 2^K-1개의 노드로 이루어져 있다. (아래 그림) 각 노드에는 그 곳에 위치한 빌딩의 번호가 붙여져 있다. 또, 가장 마지막 레벨을 제외한 모든 집은 왼쪽 자식과 오른쪽 자식을 갖는다.



깊이가 2와 3인 완전 이진 트리

상근이는 도시에 있는 모든 빌딩에 들어갔고, 들어간 순서대로 번호를 종이에 적어 놓았다. 한국으로 돌아온 상근이는 도시가 어떻게 생겼는지 그림을 그려보려고 하였으나, 정확하게 기억이 나지 않아 실패했다. 하지만, 어떤 순서로 도시를 방문했는지 기억해냈다.

1. 가장 처음에 상근이는 트리의 루트에 있는 빌딩 앞에 서있다.
2. 현재 빌딩의 왼쪽 자식에 있는 빌딩에 아직 들어가지 않았다면, 왼쪽 자식으로 이동한다.
3. 현재 있는 노드가 왼쪽 자식을 가지고 있지 않거나 왼쪽 자식에 있는 빌딩을 이미 들어갔다면, 현재 노드에 있는 빌딩을 들어가고 종이에 번호를 적는다.
4. 현재 빌딩을 이미 들어갔다 온 상태이고, 오른쪽 자식을 가지고 있는 경우에는 오른쪽 자식으로 이동한다.
5. 현재 빌딩과 왼쪽, 오른쪽 자식에 있는 빌딩을 모두 방문했다면, 부모 노드로 이동한다.

왼쪽 그림에 나와있는 마을이라면, 상근이는 2-1-3 순서대로 빌딩을 들어갔을 것이고, 오른쪽 그림의 경우에는 1-6-4-3-5-2-7 순서로 들어갔을 것이다. 상근이가 종이에 적은 순서가 모두 주어졌을 때, 각 레벨에 있는 빌딩의 번호를 구하는 프로그램을 작성하시오.

문제 2) 완전 이진 트리(9934)

완전 이진 트리 성공 다국어

☆ 한국어

실버 I

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	3574	2401	1941	69.795%

입력

첫째 줄에 K ($1 \le K \le 10$)가 주어진다.

둘째 줄에는 상근이가 방문한 빌딩의 번호가 들어간 순서대로 주어진다. 모든 빌딩의 번호는 중복되지 않으며, 구간 $[1,2^K)$에 포함된다.

출력

총 K개의 줄에 걸쳐서 정답을 출력한다. i번째 줄에는 레벨이 i인 빌딩의 번호를 출력한다. 출력은 왼쪽에서부터 오른쪽 순서대로 출력한다.

예제 입력 1 복사

```
2
2 1 3
```

예제 출력 1 복사

```
1
2 3
```

예제 입력 2 복사

```
3
1 6 4 3 5 2 7
```

예제 출력 2 복사

```
3
6 2
1 4 5 7
```

문제 2) 완전 이진 트리(9934)

중위 순회임을 알아채고, 중위 순회가 주어졌을 때 개형을 복원하는 문제.
문제 조건을 잘 읽어보면 '완전 이진 트리'이면서 포화이진트리임을 알 수 있음.
이 때 중위 순회를 하게되면 재귀적으로 잘랐을 때 가운데 값이 루트 노드임을 알 수 있당.

문제 2) 완전 이진 트리(9934)

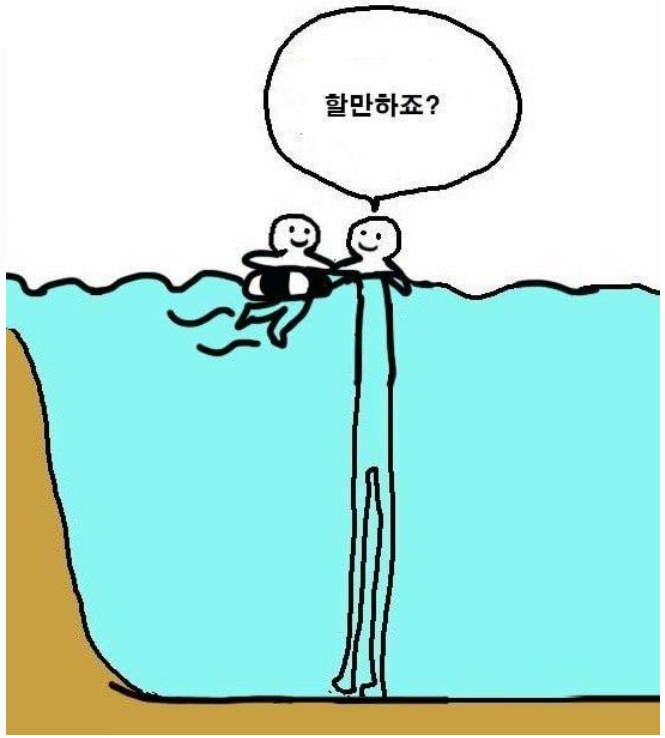
```python
from sys import stdin

input = stdin.readline

n = int(input())
nums = list(map(int, input().split()))
num_idx = 0
tree = [[] for _ in range(n)]

def inorder(depth):
    global tree, num_idx
    if depth == n:
        return

    inorder(depth+1)
    tree[depth].append(nums[num_idx])
    num_idx += 1
    inorder(depth+1)

inorder(0)
for x in tree:
    print(*x)
```

와따시의 풀이..
그냥 inorder을 실행해서 풀었다.
완전 이진 트리와 중위 순회의 특징을
알지 못하고..

문제 2) 완전 이진 트리(9934)

고수의 풀이 in python

```python
def sol(start, end , level):
    if start == end:
        ans[level].append(tree[start])
        return
    center = (start + end) // 2
    ans[level].append(tree[center])
    sol(start, center - 1, level + 1)
    sol(center + 1, end, level + 1)


Level = int(input())
tree = list(map(int, input().split()))
l = len(tree)
ans = [[] for _ in range(Level)]

sol(0, l - 1, 0)
for a in ans:
    print(*a)
```



문제 2) 완전 이진 트리(9934)

고수의 풀이(?) in java

```
public static void solve(int s, int e, int floor) {

	if (floor == K)
		return;

	int m = (s + e) / 2;
	ans[floor].append(arr[m] + " ");

	solve(s, m - 1, floor + 1);
	solve(m + 1, e, floor + 1);
}
```

```
import java.io.*;

public class n09934 {

	static int K;
	static int[] arr;
	static StringBuffer[] ans;

	public static void main(String[] args) throws Exception {

		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));

		K = Integer.parseInt(br.readLine());
		arr = new int[(int) Math.pow(2, K) - 1];

		String[] input = br.readLine().split(" ");
		for (int i = 0; i < arr.length; i++)
			arr[i] = Integer.parseInt(input[i]);

		ans = new StringBuffer[K];
		for (int i = 0; i < K; i++)
			ans[i] = new StringBuffer();

		solve(0, arr.length - 1, 0);

		for (int i = 0; i < K; i++)
			bw.write(ans[i].toString() + "\n");
		bw.flush();

	}
```

문제 2) 완전 이진 트리(9934)

고수의 풀이(?) in java

```cpp
#include <iostream>
#include <vector>
using namespace std;

//1<=k<=10
//2^k
int input[1024];
int k;
vector<int> arr[10];
void insertTree(int depth,int start, int end) {

	if (start >= end) {
		return;
	}
	int mid = (start + end) / 2;
	arr[depth].push_back(input[mid]);
	insertTree(depth + 1, start, mid);
	insertTree(depth + 1, mid+1, end);

}

int main() {
	ios::sync_with_stdio(false);
	cin.tie(0);
	cout.tie(0);

	cin >> k;


	int num;
	int idx = 0;
	while (cin >> num) {
		input[idx++] = num;
	}
	insertTree(0,0, idx);
	for (int i = 0; i < k; i++) {
		for (auto o : arr[i]) {
			cout << o << ' ';
		}
		cout << '#n';
	}
	return 0;
}
```

고생하셨습니다…