

1. Baseball Game

2. 유효한 괄호 (Valid Parentheses)

3. BinaryTree Level Order

Baseball Game

Problem

Input: ["5","-2","4","C","D","9","+","+"]

Output: 27

1: The sum is: 5.

2: -2 points. The sum is: 3.

3: 4 points. The sum is: 7.

C: 3번 데이터 삭제. The sum is: 3.

4: 2번의 -2 값 더블 -4을 얻는다 $-4+3=-1$. The sum is: -1.

5: 9 points. The sum is: 8.

6: $-4 + 9 = 5$ points. The sum is 13.

7: $9 + 5 = 14$ points. The sum is 27.

Soution

+

C

D

14

5

9

-4

-2

5

Baseball Game

문제 Format

```
class Solution {  
    public int solve(String[] strs) { { }  
}
```

Solution

BFS

1. 문제를 정확히 이해
2. 알고리즘 정하고 답을 그릇 정한다
3. for 문 돌리기
4. 생각->프로그램(한국말로 생각하고->Java)
결과를 해석하여 이미지화시킨다

I Can Image

유효한 괄호 (Valid Parentheses)

설명

String `s`가 주어집니다. `s`는 '(', ')', '{', '}', '[', ']' 로 이루어집니다.
유효한 괄호인지 체크하여 boolean값으로 리턴하세요.

입력 문자열은 다음과 같은 경우에 유효합니다.

1. 열린 괄호는 동일한 유형의 괄호로 닫아야합니다.
2. 열린 괄호는 올바른 순서로 닫아야합니다.

입출력

Input: `s = "()"`

Output: true

Input: `s = "()[]{}"`

Output: true

Input: `s = "([)]"`

Output: false

Input: `s = "{[]}"`

Output: true

문제 Format

```
class Solution {  
    public boolean solve(String str) { }  
}
```

제한사항

$1 \leq s.length \leq 10^4$
`s` consists of parentheses only '()[]{}'.

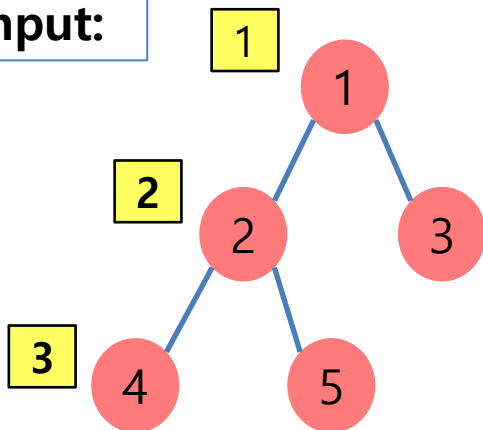
이진 트리 레벨 순서 순회 (BinaryTree Level Order)

설명

이진 트리가 주어지면 노드 값의 레벨 순서 순회를 반환 합니다 .
(즉, 왼쪽에서 오른쪽으로, 레벨별로)

입출력

Input:



Output: [[1],[2,3],[4,5]]

참고) class TreeNode{
 int val;
 TreeNode left, right;
 TreeNode(int x){
 this.val = x;
 }
}

제한사항

The number of nodes in the tree is in the range [0, 2000].
-1000 <= Node.val <= 1000

이진 트리 레벨 순서 순회 (BinaryTree Level Order)

문제 Format

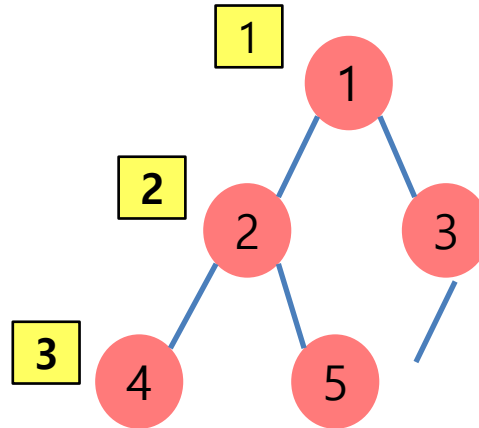
```
class TreeNode{
    int val;
    TreeNode left, right;
    TreeNode(int x){
        this.val = x;
    }
}

class Solution {
    public List<List<Integer>> solve(TreeNode tree) { { }
}
```

BinaryTree Level Order

Problem

Input:

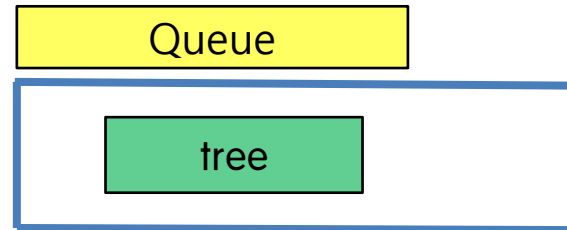
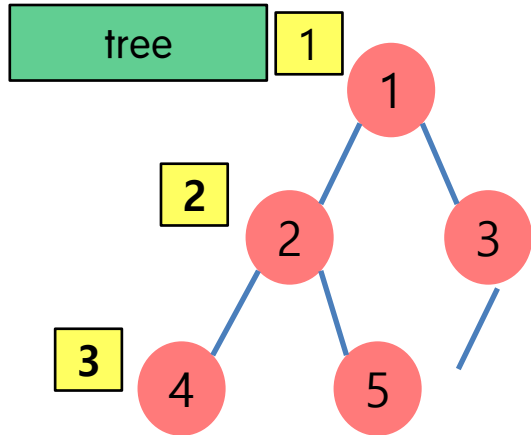


Output: `[[1],[2,3],[4,5]]`

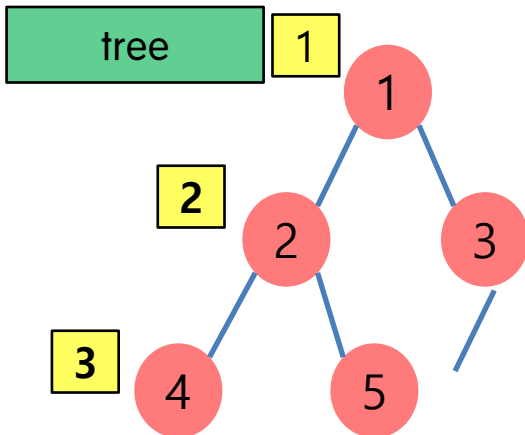
Queue bfs

Stack dfs

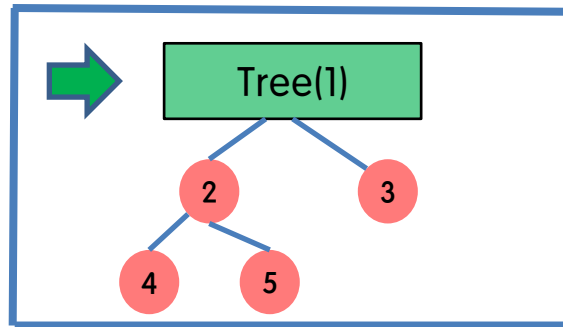
BinaryTree Level Order



BinaryTree Level Order

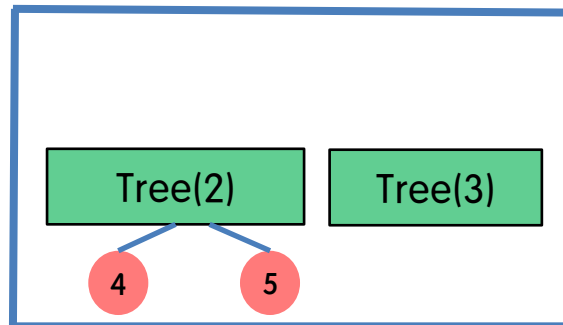


Queue



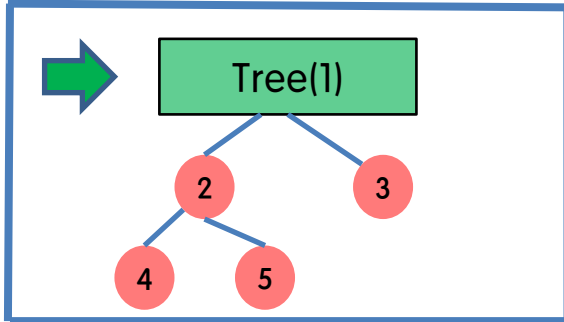
→ List [1]

Queue



→ List [2,3]

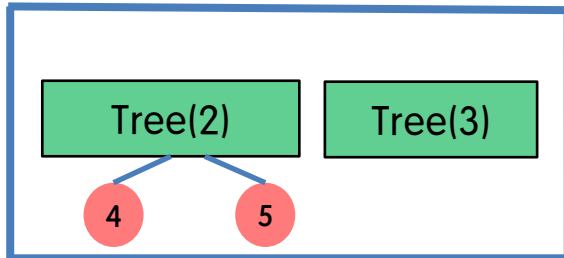
Queue



→ List [1]

List<List [1]>

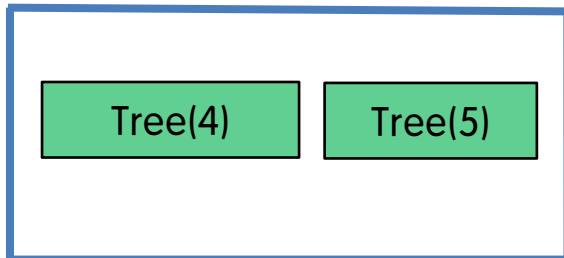
Queue



→ List [2,3]

List<List [1], [2,3]>

Queue



→ List [4,5]

List<List [1], [2,3], [4,5]> >