문제 1,2) NumberOfIslands

Given a 2d grid map of '1's (land) and '0's (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:
Input:
11110
11010
11000
00000
Output: 1

Example 2:
Input:
11000
11000
00100
00011
Output: 3

1. NumberOfIsland

Problem

- {'1','1','1','0','1'},
  {'1','1','0','0','0'},
  {'1','1','0','0','1'},
  {'0','0','0','0','1'}

1은 육지, 0 은 바다

Example

Input: J = "aA", S = "aAAbbbb" Output: 3

Note

S and J will consist of letters and have length at most 50.
The characters in J are distinct

NumberOfIslands_BFS(바비큐)

{'1','1','1','0','1'},   1은 육지, 0 은 바다
{'1','1','0','0','0'},   Output :3
{0',0','0','0','1'},
{'0','0','0','0','1'}

{00, 01, 02, 03, 04},
{10, 11, 12, 13, 14},
{20, 21, 22, 23, 24},
{30, 31, 32, 33, 34},

queue(0,0)   0-1, 01, 10, -10

문제 3,4,5) Maximum Depth Of BinaryTree
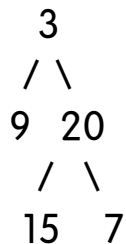
Problem

Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

Note: A leaf is a node with no children.
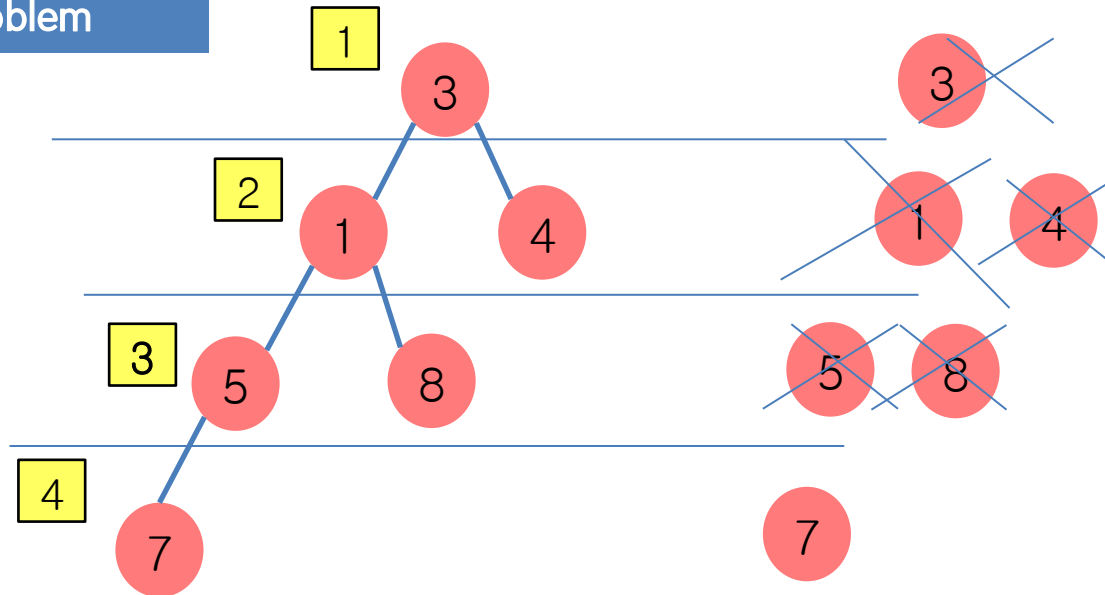
Example:

Given binary tree [3,9,20,null,null,15,7],

```
    3
   / \
  9  20
    /  \
   15   7
```
return its depth = 3.

Maximum Depth Of BinaryTree

Problem

1

3

2
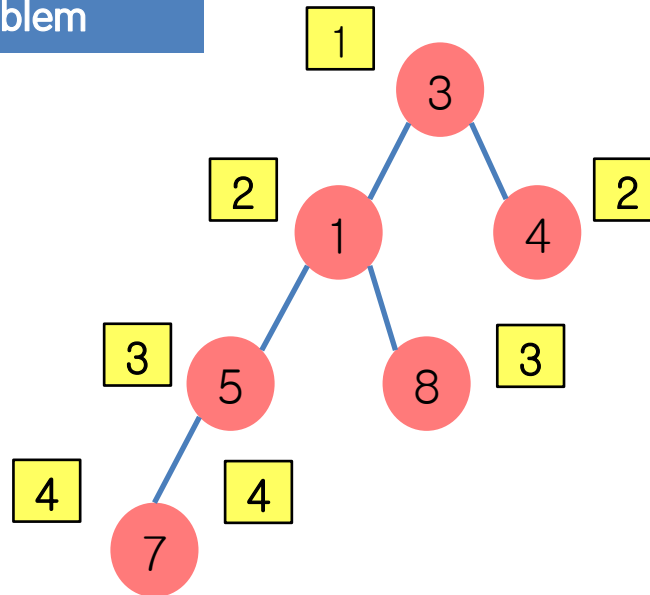
1   4

3

5   8

4

7

3

1   4

5   8

7

Example

1. Queue 방식, 근접해서 한칸씩
2.

Maximum Depth Of BinaryTree
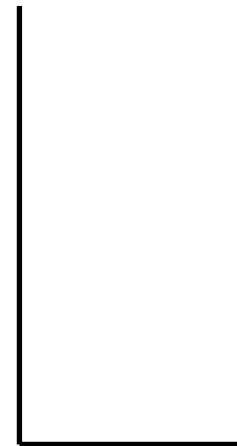
Problem

Example

1. DFS방식 스택을 이용한다.
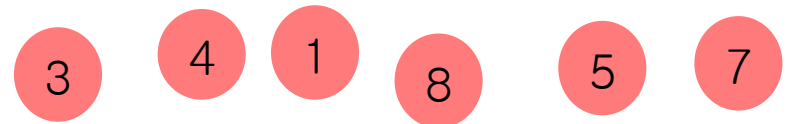
Maximum Depth Of BinaryTree

BFS-queue

Problem

1

3

2

1   4

3

5   8

4

7

3

1   4

5   8

7
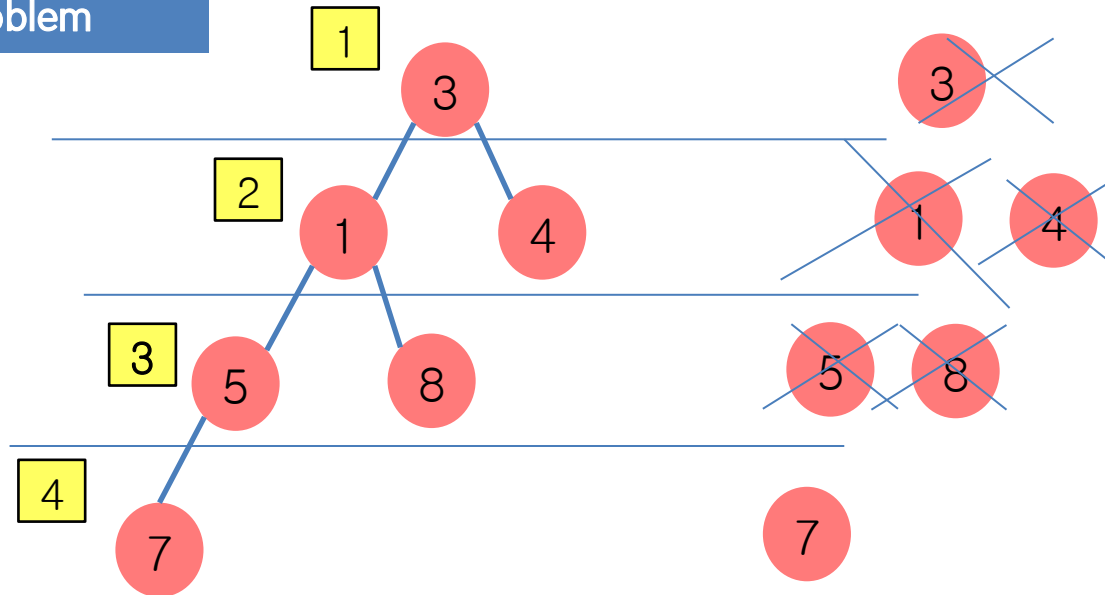
Example

1. Queue 방식, 근접해서 한칸씩
2.

문제 6) Max Of Islands

Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

Note: A leaf is a node with no children.

Example:

Given binary tree [3,9,20,null,null,15,7],

```
    3
   / \
  9  20
    /  \
   15   7
```
return its depth = 3.

Max Of Islands

{1, 1, 0, 1, 1},
{0, 1, 1, 0, 0}
{0, 0, 0, 0, 0}
{1, 1, 0, 1, 1}
{1, 0, 1, 1, 1}
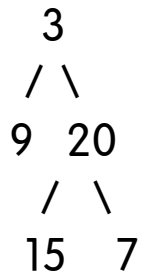{1, 0, 1, 1, 1}

1은 육지, 0 은 바다
Output :8

1. Number of island
섬의 개수를 구한다.

2. Count 변수를 두고
4,2,4,8
max= Math.max(max, area);

Example

{00, 01, 02, 03, 04},
{10, 11, 12, 13, 14},
{20, 21, 22, 23, 24},
{30, 31, 32, 33, 34},
{40, 41, 42, 43, 44},
{50, 51, 52, 53, 54},

문제 7) Word Ladder

Given two words (*beginWord* and *endWord*), and a dictionary's word list, find the length of shortest transformation sequence from *beginWord* to *endWord*, such that:

Only one letter can be changed at a time.

Each transformed word must exist in the word list. Note that *beginWord* is *not* a transformed word.

Note:

Return 0 if there is no such transformation sequence.

All words have the same length.

All words contain only lowercase alphabetic characters.

You may assume no duplicates in the word list.

You may assume *beginWord* and *endWord* are non-empty and are not the same.

Example 1:

Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]

 Output: 5

 Explanation: As one shortest transformation is

"hit" -> "hot" -> "dot" -> "dog" -> "cog", return its length 5.

Word Ladder

Input: beginWord = "hit", endWord = "cog",
wordList = ["hot","dot","dog","lot","log","cog"]

Output: 5
"hit" -> "hot" -> "dot" -> "dog" -> "cog",

Example

1.	hit	Queue size 1

1.   hit
ait, hat , hia
zit, hzt,   hiz

Queue size 1
hot

2. hot
aot   hat   hoa
zot   hzt   hoz

Queue size 3
Dot,lot,hot

3. dot ,lot, hot
       log
4. 10개

Queue size 10
Dot,lot.hot,log..
33개

1
hit
2
hot
3
dot
4
dog
5
cog
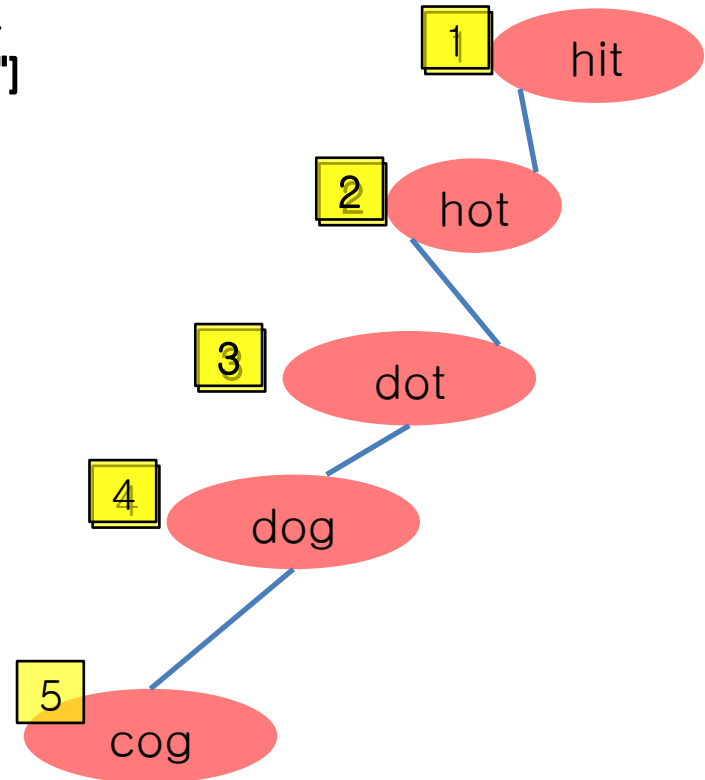
문제 8) Word Search

Given a 2D board and a word, find if the word exists in the grid.

The word can be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The same letter cell may not be used more than once.

Example:

board =
[
  ['A','B','C','E'],
  ['S','F','C','S'],
  ['A','D','E','E']
]

Given word = "ABCCED", return true.
Given word = "SEE", return true.
Given word = "ABCB", return false.

WordSearch

{'A','B', C','E'},
{'S','F', 'C','S'},
{'A','D','E','E'}
String word = "ABCCED";

Output : true;

Example

1. Dfs 문제
2. 제한 조건 찾기
   'A' 를 찾았다면

   {00, 01, 02, 03},
   {10, 11, 12, 13},
   {20, 21, 22, 23},

   좌표, visited, word

문제 9) Remove Invalid Parentheses

Remove the minimum number of invalid parentheses in order to make the input string valid. Return all possible results.

Note: The input string may contain letters other than the parentheses (and).

Example 1:

Input: "()())()"
Output: ["()()()", "(())()"]
Example 2:

Input: "(a)())()"
Output: ["(a)()()", "(a())()"]
Example 3:

Input: ")("
Output: [""]

Remove Invalid Parentheses

Input: "(a)())()"
Output: ["(a)()()", "(a())()"]

Input: "()())()"
Output: ["()()()", "(())()"]

Ex) 012345
substring(0,3): 012
substring(3) : 345

Solution

0	**1**	2	3	**4**	5	**6**	**7**

(a)( )) ( )

1. Queue에 저장
2. 케이스별로 valid 체크
3. Brace 체크, 쌍으로 존재
4. newStr 큐에 저장

i=0   + a) ( )) ( )

i=2   (a   + ( )) ( )

i=5   (a) ( )   + ( )

Remove Invalid Parentheses

0	1	2	3	4	5	6	7

(a) ( )) ( ) ➡ a) ( )) ( )

(a) ( )) ( ) ➡

(a) ( )) ( ) ➡ (a   ( )) ( )

(a) ( )) ( ) ➡ (a)   )) ( )

(a) ( )) ( ) ➡ (a) (   ) ( )

(a) ( )) ( ) ➡ (a) ( )   ( )

(a) ( )) ( ) ➡ (a) ( ))   )

(a) ( )) ( ) ➡ (a) ( )) (

Remove Invalid Parentheses

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

a) ( )) ( )

a) ( )) ( )   a   ( )) ( )

a) ( )) ( )   a)   )) ( )

(a ( )) ( )

문제10) Maze

There is a ball in a maze with empty spaces and walls. The ball can go through empty spaces by rolling up, down, left or right, but it won't stop rolling until hitting a wall. When the ball stops, it could choose the next direction.

Given the ball's start position, the destination and the maze, determine whether the ball could stop at the destination.

The maze is represented by a binary 2D array. 1 means the wall and 0 means the empty space. You may assume that the borders of the maze are all walls. The start and destination coordinates are represented by row and column indexes.

Example 1:
Input 1: a maze represented by a 2D array
0 0 1 0 0
0 0 0 0 0
0 0 0 1 0
1 1 0 1 1
0 0 0 0 0
Input 2: start coordinate (rowStart, colStart) = (0, 4)
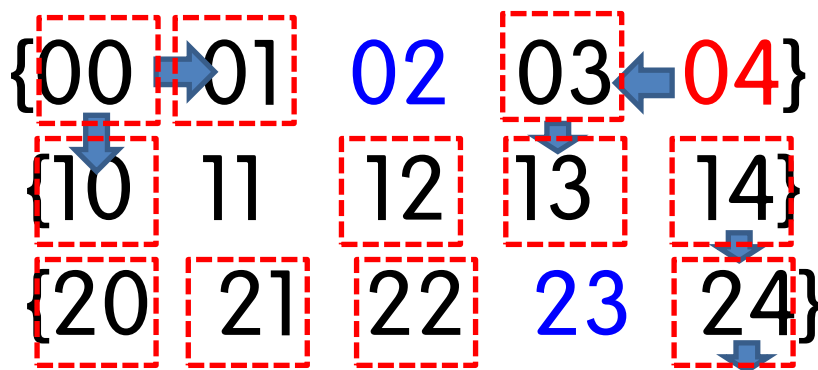Input 3: destination coordinate (rowDest, colDest) = (4, 4)

Output: true

Maze1_BFS

1 0 2 0 1
0 0 0 0 0
0 0 1 0 0

Input : start = (0, 4)
Input : destination = (4, 4)

결과값 : true

left -> down -> left -> down -> right -> down -> right.

Example

{00 ➡ 01   02   03 ⬅ 04}

{10   11   12   13   14}

{20   21   22   23   24}

1.벽면 인식후, 벽 앞 위치

Maze 1_DFS

0 0 1 0 0
0 0 0 0 0
0 0 0 1 0
1 1 0 1 1
0 0 0 0 0

Input : start = (0, 4)
Input : destination = (4, 4)

결과값 : true

Example

{00   01   02   03 ← 04}   1.벽면 인식후, 벽 앞 위치
{10   11   12   13   14}
{20   21   22   23   24}
{30   31   32   33   34}
{40   41   42   43   44}