

BFS - Breadth First Search

宽度优先搜索

When

Level Order Traversal
Connected Component
Topological Sorting

1 \Rightarrow BFS in Binary Tree
2 \Rightarrow BFS in Graph (Topological Sorting)
3 \Rightarrow BFS in matrix

If one problem can be solved with BFS, do not use DFS !

1. BFS in Binary Tree

129. Sum Root to Leaf Numbers && 1022. Sum of Root To Leaf Binary Numbers

101. Symmetric Tree

662. Maximum Width of Binary Tree

863. All Nodes Distance K in Binary Tree

987. Vertical Order Traversal of a Binary Tree

2. BFS in Graph (Topological Sorting)

127. Word Ladder

133. Clone Graph

210. Course Schedule II

310. Minimum Height Trees

743. Network Delay Time

1129. Shortest Path with Alternating Colors

1311. Get Watched Videos by Your Friends

847. Shortest Path Visiting All Nodes

1345. Jump Game IV

3. BFS in matrix

200. Number of Islands

407. Trapping Rain Water II

417. Pacific Atlantic Water Flow

529. Minesweeper

542. 01 Matrix

Zombie in Matrix (994. Rotting Oranges)

909. Snakes and Ladders

934. Shortest Bridge

773. Sliding Puzzle

815. Bus Routes

752. Open the Lock

854. K-Similar Strings

864. Shortest Path to Get All Keys

1298. Maximum Candies You Can Get from Boxes

1036. Escape a Large Maze

1091. Shortest Path in Binary Matrix

1391. Check if There is a Valid Path in a Grid

1210. Minimum Moves to Reach Target with Rotations

1263. Minimum Moves to Move a Box to Their Target Location

1284. Minimum Number of Flips to Convert Binary Matrix to Zero Matrix

1293. Shortest Path in a Grid with Obstacles Elimination

1368. Minimum Cost to Make at Least One Valid Path in a Grid

Template - 102. Binary Tree Level Order Traversal

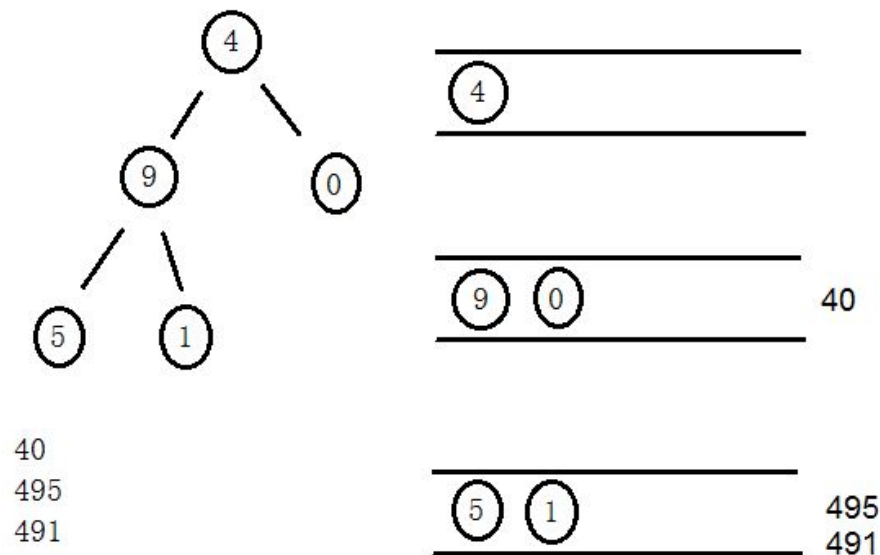
```
final List<List<Integer>> result = new ArrayList<>();
if (root == null) {
    return result;
}
// 1. create a queue FIFO, then put root in it
Queue<TreeNode> treeNodeQueue = new LinkedList<>();
treeNodeQueue.offer(root);
// 2. while queue is not null, take itself add its children
while (!treeNodeQueue.isEmpty()) {
    final int currentSize = treeNodeQueue.size();
    final List<Integer> currentLevel = new ArrayList<>();
    for (int i = 0; i < currentSize; i++) {
        TreeNode treeNode = treeNodeQueue.poll();
        currentLevel.add(treeNode.val);
        if (treeNode.left != null) {
            treeNodeQueue.offer(treeNode.left);
        }
        if (treeNode.right != null) {
            treeNodeQueue.offer(treeNode.right);
        }
    }
    result.add(currentLevel);
}
return result;
```

```
val result = mutableListOf<List<Int>>()
if (root == null) {
    return result
}
val treeNodeQueue: Queue<TreeNode> = LinkedList()
treeNodeQueue.offer(root)
while (treeNodeQueue.isNotEmpty()) {
    val currentLevel = mutableListOf<Int>()
    for (i in 0 until treeNodeQueue.size) {
        val current = treeNodeQueue.poll()
        currentLevel.add(current.`val`)
        current.left?.let { treeNodeQueue.offer(it) }
        current.right?.let { treeNodeQueue.offer(it) }
    }
    result.add(currentLevel)
}
return result
```

129. Sum Root to Leaf Numbers & 1022. Sum of Root To Leaf Binary Numbers

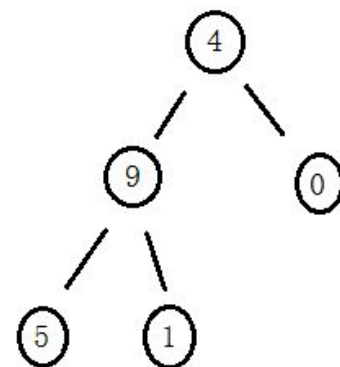
<https://leetcode.com/problems/sum-root-to-leaf-numbers/>

<https://leetcode.com/problems/sum-of-root-to-leaf-binary-numbers/>



40
495
491

BFS

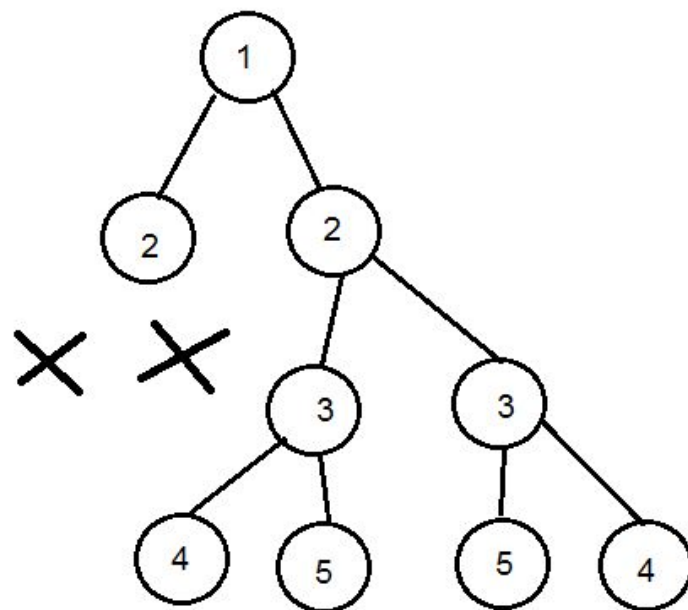
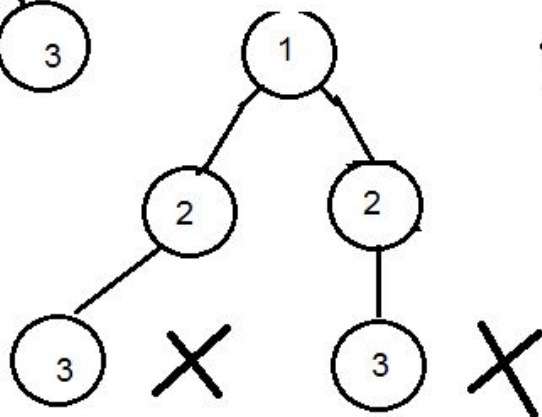
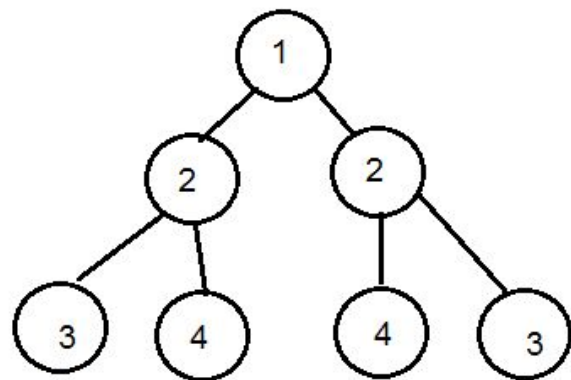


495
491
40

DFS

101. Symmetric Tree

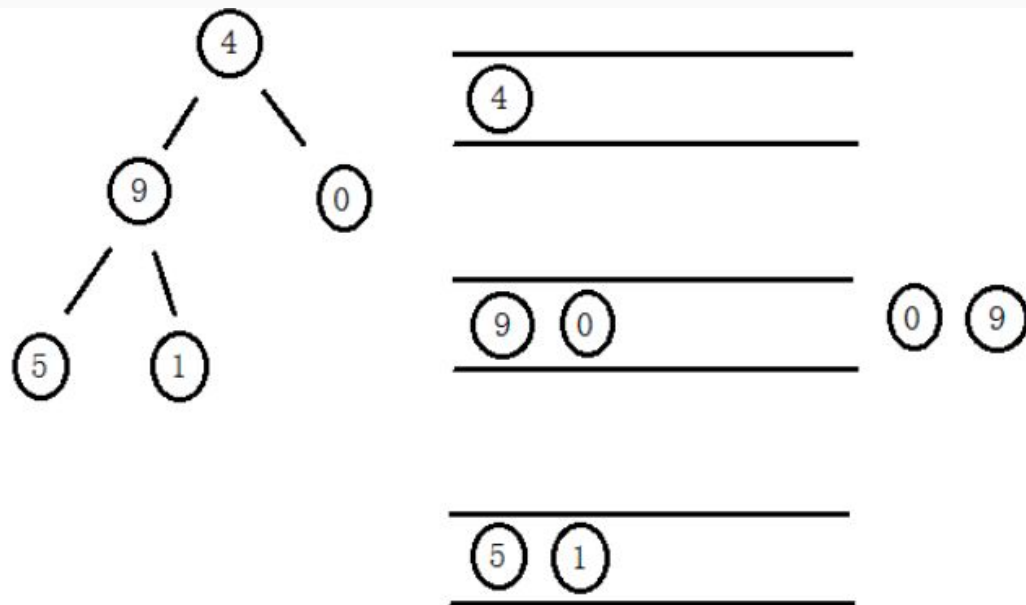
<https://leetcode.com/problems/symmetric-tree/>



103. Binary Tree Zigzag Level Order Traversal & 111. Minimum Depth of Binary Tree

<https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/>

<https://leetcode.com/problems/minimum-depth-of-binary-tree/>

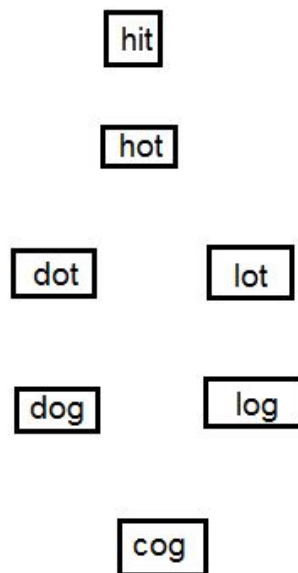


127. Word Ladder

<https://leetcode.com/problems/word-ladder/>

```
begin = "hit"
end = "cog"
wordList = ["hot","dot","dog","lot","log","cog"]

final int currentSize = queue.size();
for (int k = 0; k < currentSize; k++) {
    final String currentString = queue.poll();
    if (currentString.equals(endWord)) {
        return count;
    }
    for (int index = 0; index < currentString.length(); index++) {
        final StringBuilder str = new StringBuilder(currentString);
        for (char c = 'a'; c <= 'z'; c++) {
            str.setCharAt(index, c);
            final String newStr = str.toString();
            if (wordSet.contains(newStr) && !newStr.equals(currentString)) {
                queue.offer(newStr);
                wordSet.remove(newStr);
            }
        }
    }
    ++count;
}
```



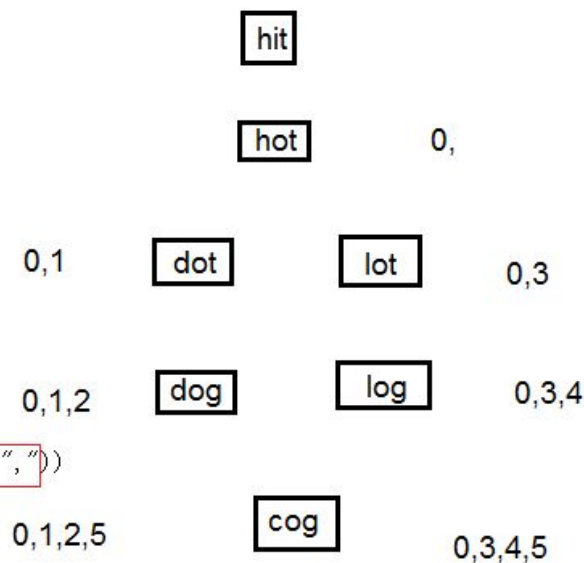
126. Word Ladder II

<https://leetcode.com/problems/word-ladder-ii/>

```
begin = "hit"  
end = "cog"  
wordList = ["hot","dot","dog","lot","log","cog"]
```

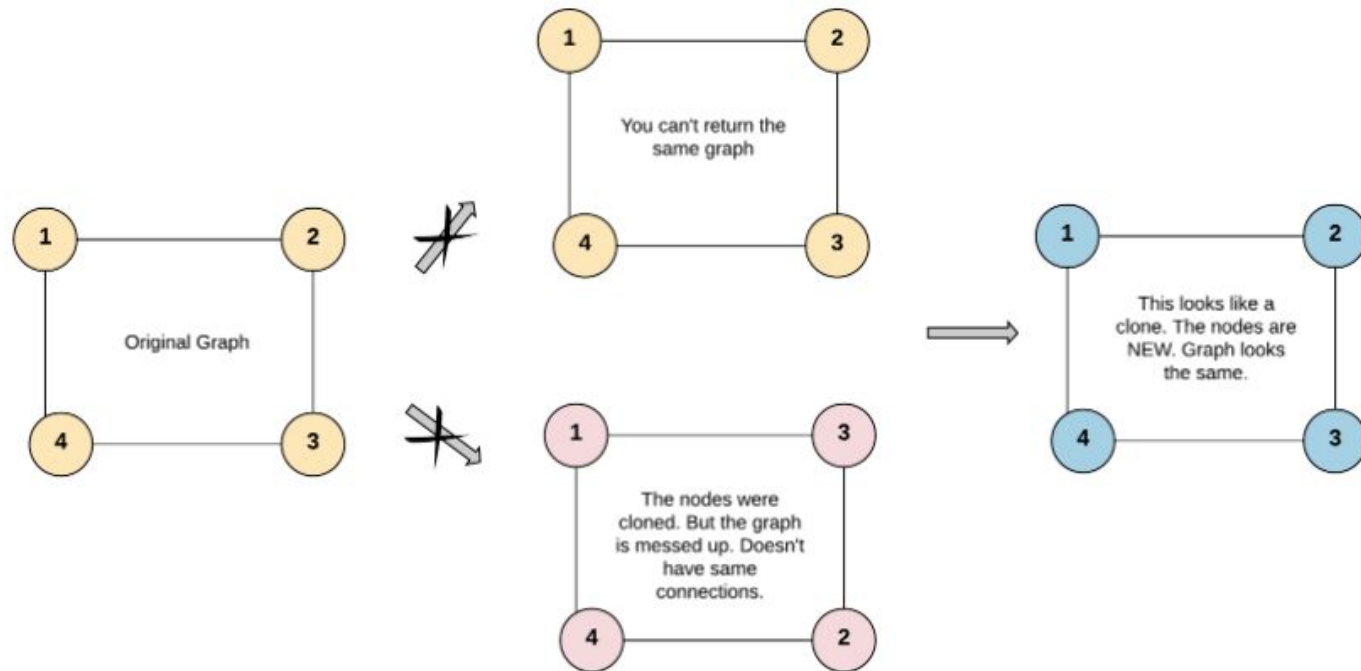
```
val currentLevelUsedWord: MutableSet<String> = HashSet()
```

```
if (wordMap.contains(new) && new != currentPair.first) {  
    if (new == endWord) {  
        queue.offer(Pair(new, currentPair.second + wordMap.getValue(new)))  
    } else {  
        currentLevelUsedWord.add(new)  
        queue.offer(Pair(new, currentPair.second + wordMap.getValue(new) + ","))  
    }  
}
```



133. Clone Graph

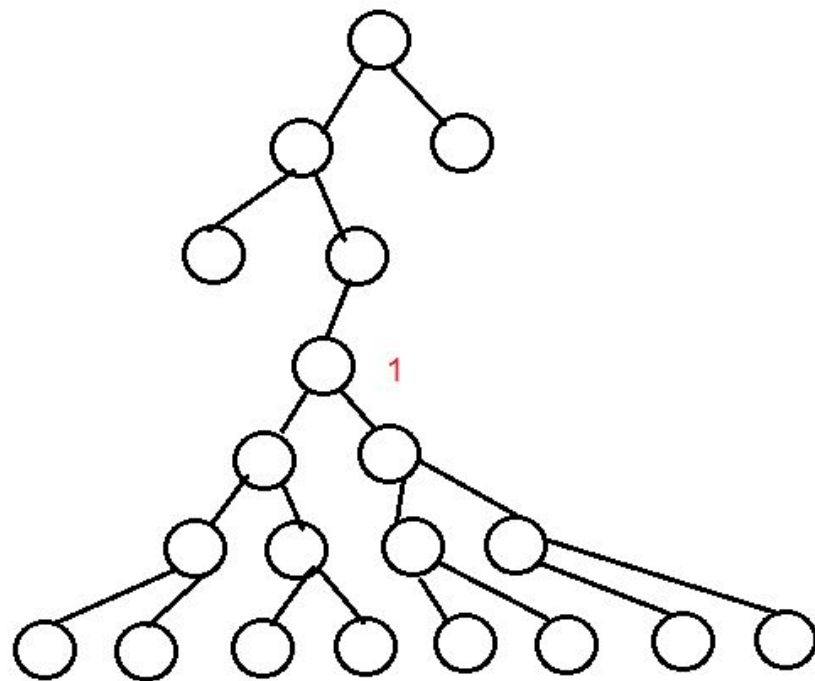
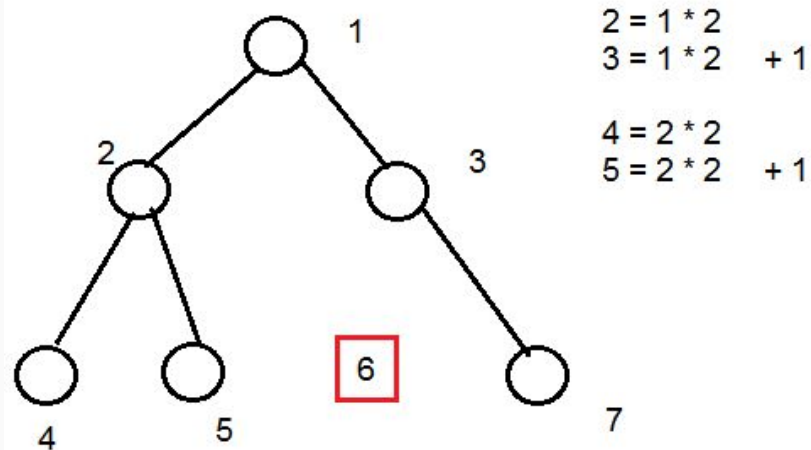
<https://leetcode.com/problems/clone-graph/>



1. BFS get all nodes
2. copy all nodes then mapping
3. cope neighbors

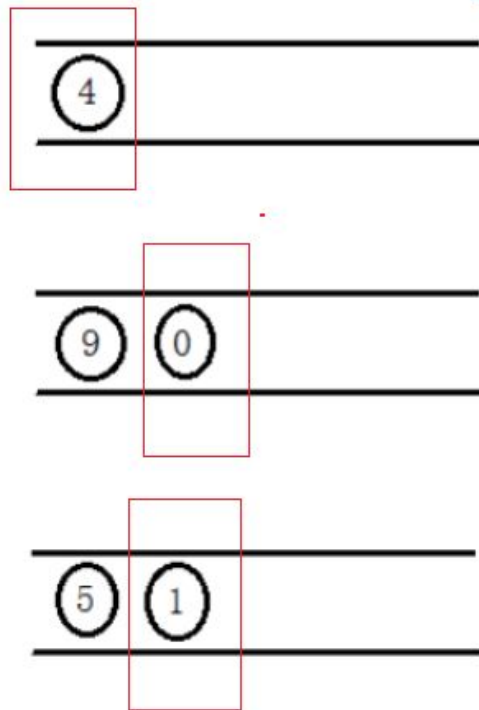
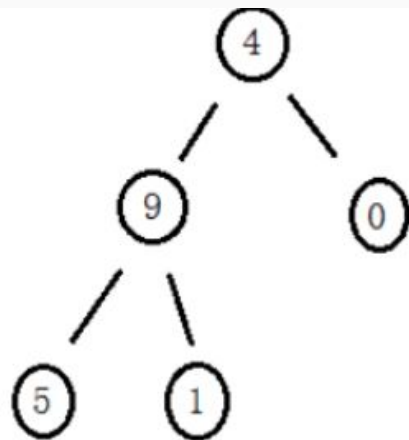
662. Maximum Width of Binary Tree

<https://leetcode.com/problems/maximum-width-of-binary-tree/>



199. Binary Tree Right Side View

<https://leetcode.com/problems/binary-tree-right-side-view/>



template review!

```
val result: MutableList<Int> = LinkedList()
if (root == null) {
    return result
}
val queue: Queue<TreeNode> = LinkedList()
queue.offer(root)
while (queue.isNotEmpty()) {
    var levelValue = 0
    for (time in 0 until queue.size) {
        val current = queue.poll()
        levelValue = current.`val`
        current.left?.let { queue.offer(it) }
        current.right?.let { queue.offer(it) }
    }
    result.add(levelValue)
}
return result
```

200. Number of Islands

<https://leetcode.com/problems/number-of-islands/>

```
[["1","1","0","0","0"],  
["1","1","0","0","0"],  
["0","0","1","0","0"],  
["0","0","0","1","1"]]
```

```
1 1 0  
1 1
```

4 directions:

nextX
nextY

1 ==> inBound

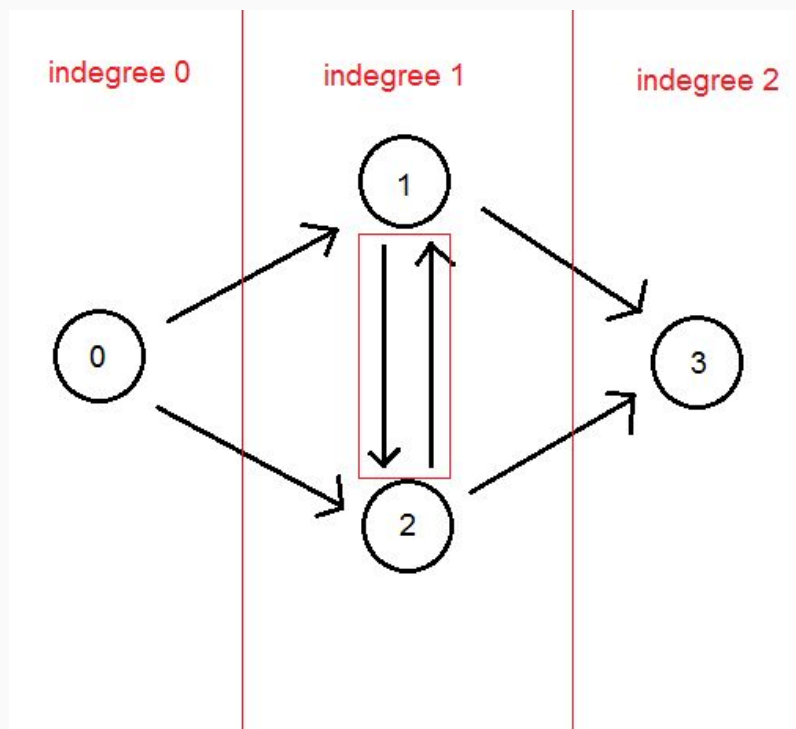
2 ==> grid[nextX][nextY] == '1'

3 ==> visited[nextX][nextY] == 0

```
private void dfs(char[][] grid, int x, int y, int[][] visited) {  
    visited[x][y] = 7;  
    for (int index = 0; index < deltaX.length; index++) {  
        final int nextX = x + deltaX[index];  
        final int nextY = y + deltaY[index];  
        if (inBound(grid, nextX, nextY) &&  
            visited[nextX][nextY] == 0 &&  
            grid[nextX][nextY] == '1') {  
            visited[nextX][nextY] = 7;  
            dfs(grid, nextX, nextY, visited);  
        }  
    }  
}
```

210. Course Schedule II

<https://leetcode.com/problems/course-schedule-ii/>



```
while (queue.isNotEmpty()) {  
    val current = queue.poll()  
    result.add(current)  
    graph[current]?.forEach {  
        if (--indegrees[it] == 0) {  
            queue.offer(it)  
        }  
    }  
}
```

1. add value to result
2. nextCourse indegree --
 if == 0
 add it in queue

if it is possible to take all courses, result.size should equal to n

301. Remove Invalid Parentheses

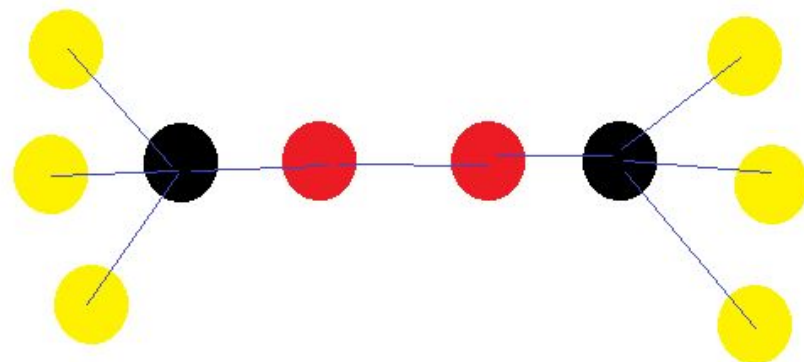
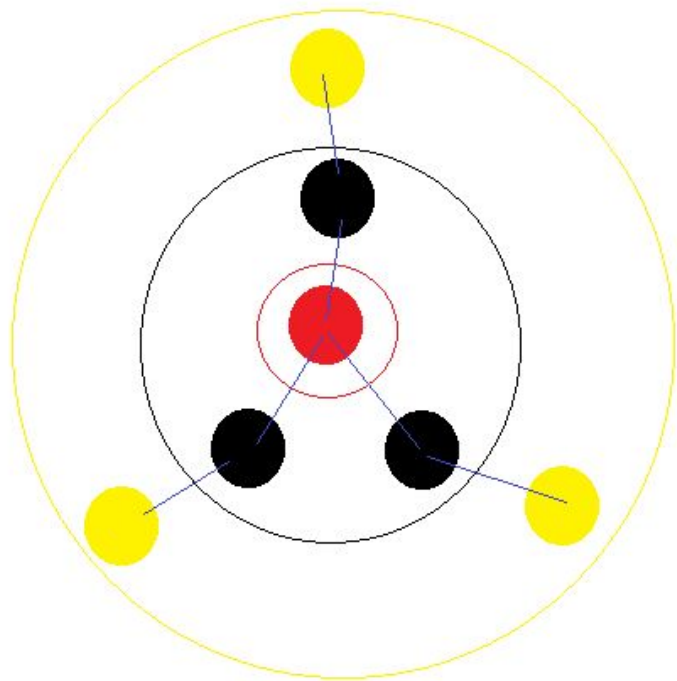
<https://leetcode.com/problems/remove-invalid-parentheses/>

level 0	1	2
(a) () ()	a) () ()	
	(a ()) ()	
	(a))) ()	
	(a) () ()	
	(a) () ()	
	(a) ())	
	(a) () (

```
private fun isValid(str: String): Boolean {  
    var count = 0  
    str.forEach {  
        if (it == '(') {  
            ++count  
        }  
        /*  
        if (it == ')') {  
            --count  
        }  
        */  
        if (it == ')' && --count < 0) {  
            return false  
        }  
    }  
    return count == 0  
}
```

310. Minimum Height Trees

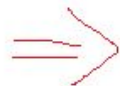
<https://leetcode.com/problems/minimum-height-trees/>



407. Trapping Rain Water II

<https://leetcode.com/problems/trapping-rain-water-ii/>

1	4	3	1	3	2
3	2	1	3	2	4
2	3	3	2	3	1



1	4	3	1	3	2
3	2	1	3	2	4
2	3	3	2	3	1

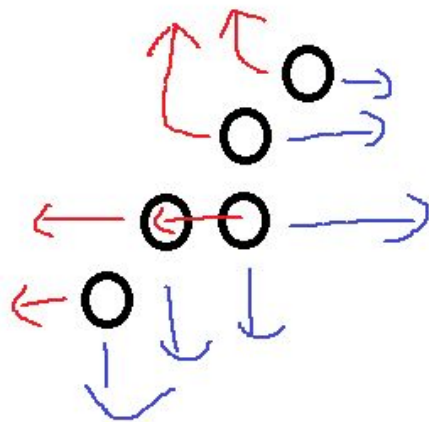
1	4	3	1	3	2
3	2	1	3	2	4
2	3	3	2	3	1

1	4	3	1	3	2
3	2	1	3	2	4
2	3	3	2	3	1

417. Pacific Atlantic Water Flow

<https://leetcode.com/problems/pacific-atlantic-water-flow/>

```
Pacific ~ ~ ~ ~ ~  
~ 1 2 2 3 (5) *  
~ 3 2 3 (4) (4) *  
~ 2 4 (5) 3 1 *  
~ (6) (7) 1 4 5 *  
~ (5) 1 1 2 4 *  
  * * * * * Atlantic
```



idea: start with the node on the board of matrix, then search where it can reach . it's a treaceback.

1. visited1 visited2 queue1 queue2
2. add board node in queue and set visited
3. bfs
4. visited1 && visited2

529. Minesweeper

<https://leetcode.com/problems/minesweeper/>

Input:

```
[['E', 'E', 'E', 'E', 'E'],  
 ['E', 'E', 'M', 'E', 'E'],  
 ['E', 'E', 'E', 'E', 'E'],  
 ['E', 'E', 'E', 'E', 'E']]
```

Click : [3,0]

Output:

```
[['B', '1', 'E', '1', 'B'],  
 ['B', '1', 'M', '1', 'B'],  
 ['B', '1', '1', '1', 'B'],  
 ['B', 'B', 'B', 'B', 'B']]
```

Explanation:



1. for a given position:

- > with no surrounded mine
mark as 'B', **add all neighbors**
- > with more than one mine
mark the total numbers

542. 01 Matrix

<https://leetcode.com/problems/01-matrix/>

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

idea: use BFS to set the level value to cell

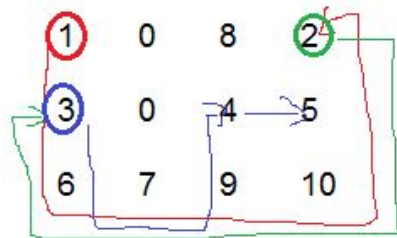
1. add all cells value 1 surrounded with 0, mark level 1

2. add adjacent cells value 1, mark level 2

3. do the same things until queue is empty

675. Cut Off Trees for Golf Event

<https://leetcode.com/problems/cut-off-trees-for-golf-event/>



1. find all cells where the value > 1, and add them into a list (value,x,y)
2. sort list by value
3. bfs find the distance between each item, if can not find a path, return -1

bfs

level iterate in matrix

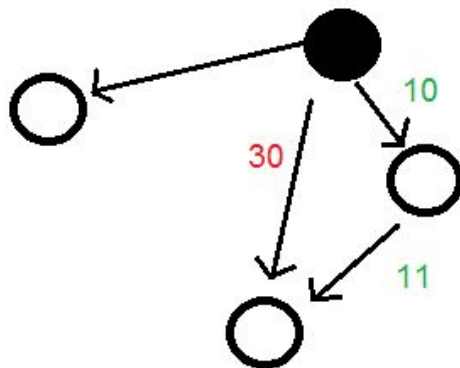
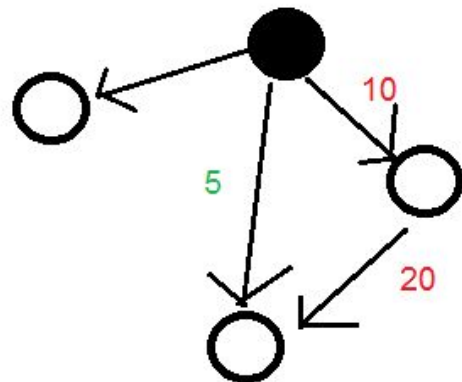
given a start point (x,y)

after each size of queue, ++level

if found end point, return level, else -1

743. Network Delay Time

<https://leetcode.com/problems/network-delay-time/>



1. transform times into a graph
2. build a map, priorityQueue
3. bfs
4. if map.size != N, return -1

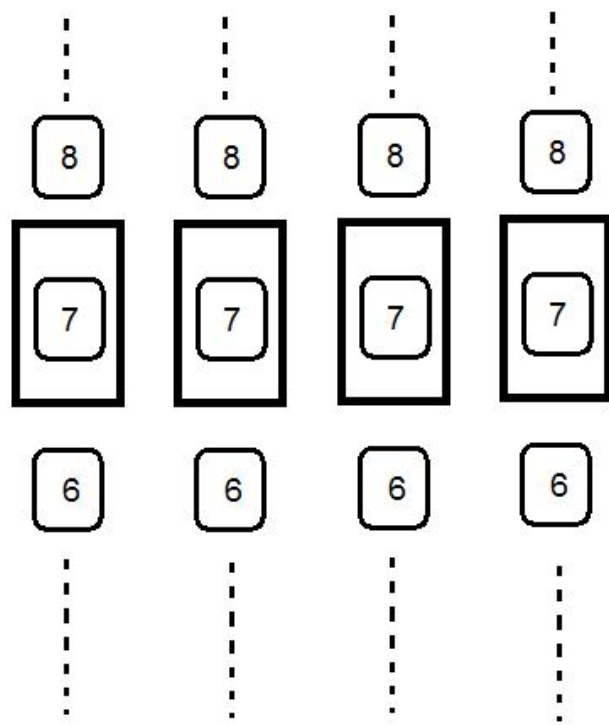
no need to iterate by level because of priority queue

if map not contains current node -> do

if map not contains next node -> add it in queue

752. Open the Lock

<https://leetcode.com/problems/open-the-lock/>



idea: to search minimum move to unlock use BFS

1. if deadEnds contains "0000" or deadEnds contains all nextMoves of target, return -1

2. create visited, queue, level

3. iterate queue size by level, if value == target, return level

for each string, it will have 8 next moves,
if not visited, not deadEnds contains => add it into queue, visited true

Zombie in Matrix

<https://www.lintcode.com/problem/zombie-in-matrix/description>

Input:

```
[[0,1,2,0,0],  
 [1,0,0,2,1],  
 [0,1,0,0,0]]
```

Output:

2

1. create visited, queue

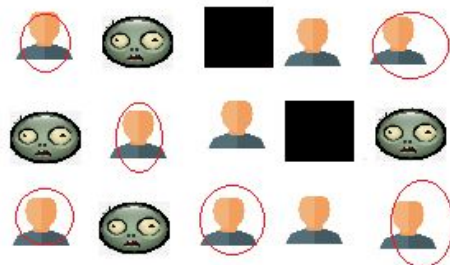
2. add all 1 (zombie) into the queue, and count the number of peoples

3. bfs

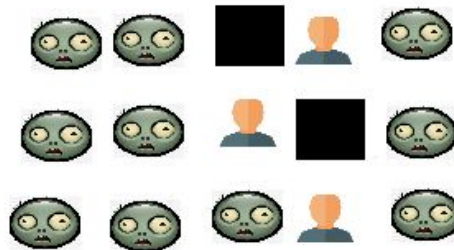
when add to queue, count++

compare count == peoples

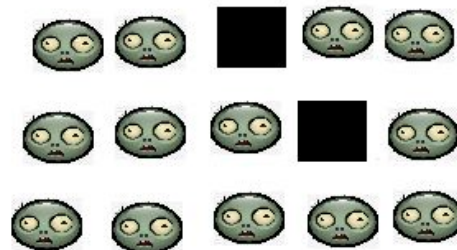
day 0



day 1

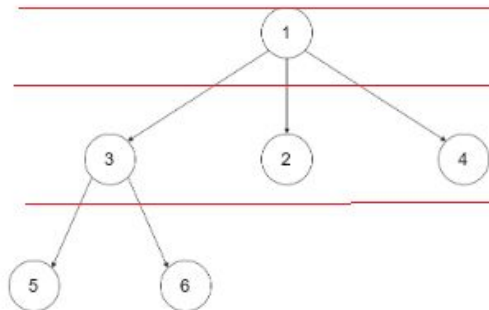


day 2



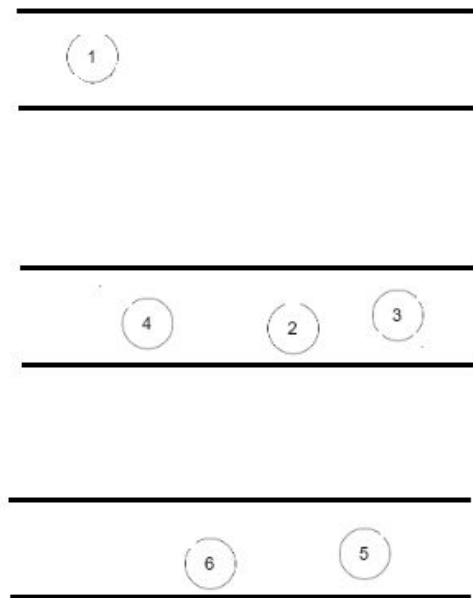
429. N-ary Tree Level Order Traversal

<https://leetcode.com/problems/n-ary-tree-level-order-traversal/>



Input: root = [1,null,3,2,4,null,5,6]
Output: [[1],[3,2,4],[5,6]]

use breadth first search



773. Sliding Puzzle

<https://leetcode.com/problems/sliding-puzzle/>

1	2	3
4	0	5

1 2 3 4 0 5

use Breadth First Search to find the minimum steps :

1. visited queue

2. bfs

1 0 3 4 2 5

1 2 3 0 4 5

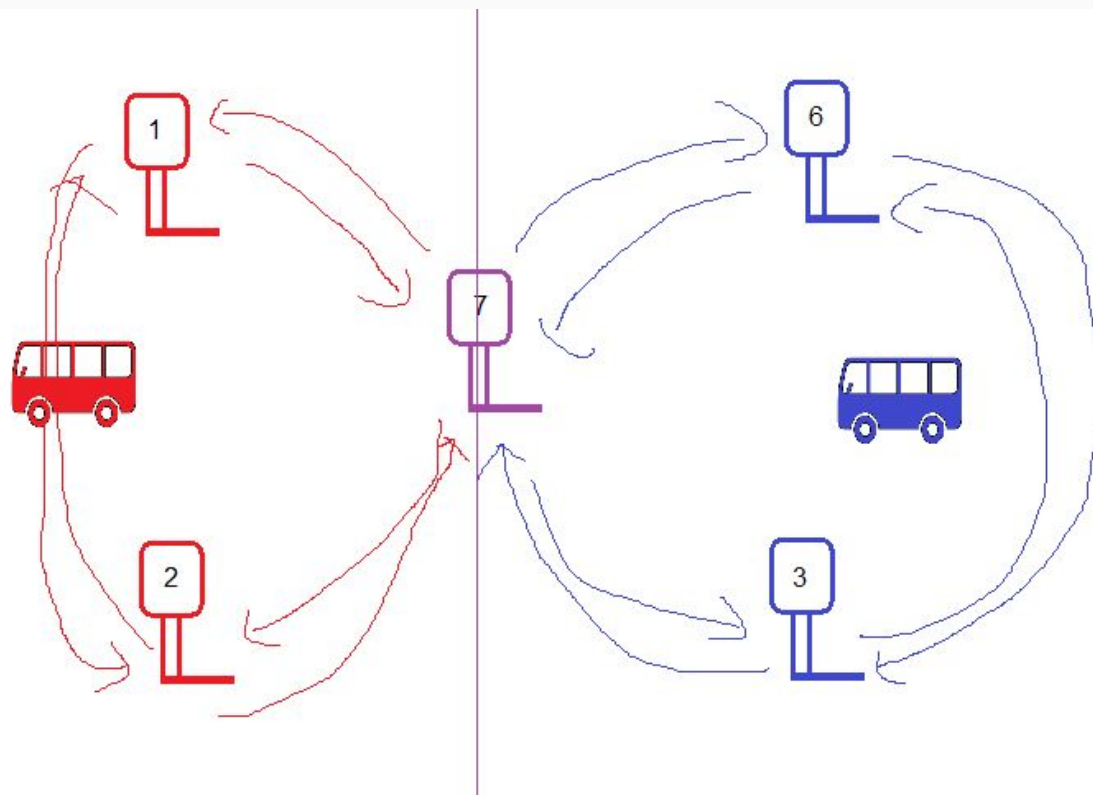
1 2 3 4 5 0

```
0, List.of(
    new int[] {0, 1},
    new int[] {0, 3}
),
1, List.of(
    new int[] {0, 1},
    new int[] {1, 2},
    new int[] {1, 4}
),
2, List.of(
    new int[] {1, 2},
    new int[] {2, 5}
),
```

```
3, List.of(
    new int[] {3, 4},
    new int[] {0, 3}
),
4, List.of(
    new int[] {5, 4},
    new int[] {3, 4},
    new int[] {1, 4}
),
5, List.of(
    new int[] {4, 5},
    new int[] {2, 5}
)
```

815. Bus Routes

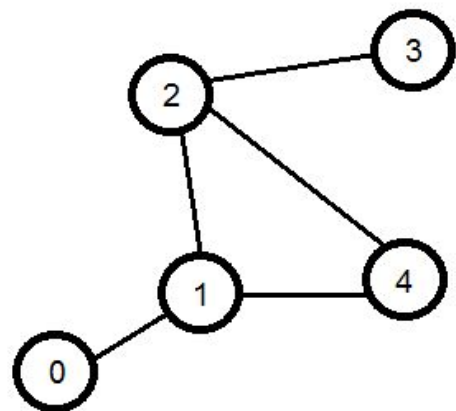
<https://leetcode.com/problems/bus-routes/>



1. `List<Set<Int>> buses` : index \Rightarrow stops
2. build the graph, bus connect to bus
3. visited, queue, endSet
(this steps int means bus number)
4. add start to queue and visited,
add end to endSet
5. bfs

847. Shortest Path Visiting All Nodes

<https://leetcode.com/problems/shortest-path-visiting-all-nodes/>



$N = 5$

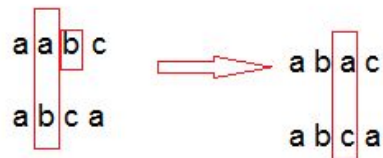
$11111 = 31 = (1 \ll 5) - 1$ endKey

0, (00001) -> 1, (00011)
1, (00010) -> 0, (00011) ~~-> 1, (00011)~~
2, (00100)
3, (01000)
4, (10000)

1. endKey, visited, queue
2. put all nodes into the queue
3. bfs

854. K-Similar Strings

<https://leetcode.com/problems/k-similar-strings/>

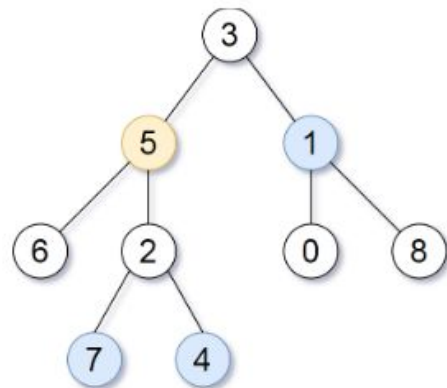


1. find the first index that current string char != target string char
2. find next char in current string that equals target char
3. exchange

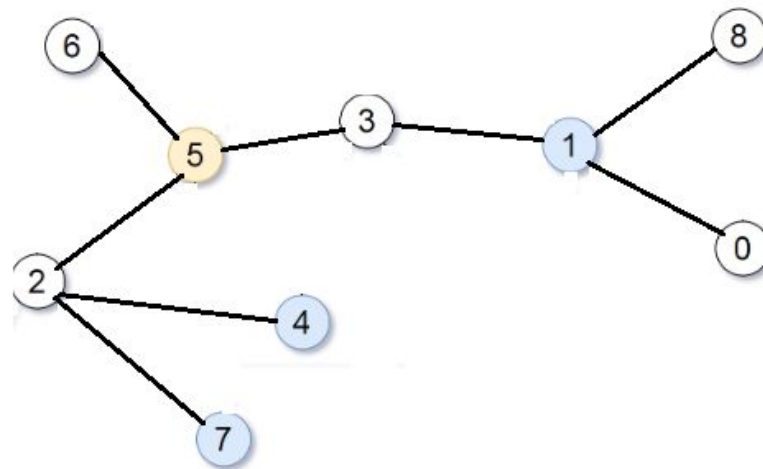
```
private List<String> getNexts(String current, String target) {  
    final List<String> results = new ArrayList<>();  
    int i = 0;  
    while (i < target.length() &&  
           current.charAt(i) == target.charAt(i)) {  
        ++i;  
    }  
    for (int j = i + 1; j < target.length(); j++) {  
        if (target.charAt(j) == current.charAt(i)) {  
            final StringBuilder newSb = new StringBuilder(current);  
            final char temp = current.charAt(i);  
            newSb.setCharAt(i, current.charAt(j));  
            newSb.setCharAt(j, temp);  
            results.add(newSb.toString());  
        }  
    }  
    return results;  
}
```

863. All Nodes Distance K in Binary Tree

<https://leetcode.com/problems/all-nodes-distance-k-in-binary-tree/>

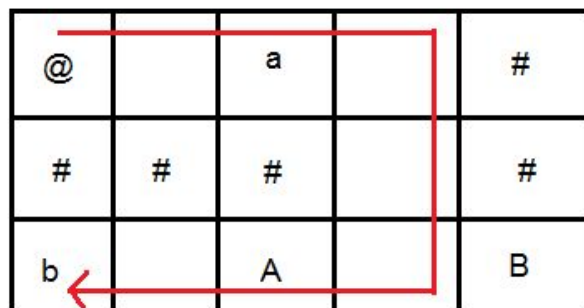


1. build graph
2. visited queue
3. breadth first search



864. Shortest Path to Get All Keys

<https://leetcode.com/problems/shortest-path-to-get-all-keys/>



a b c d e f

1 1 1 1 1 1 endKey

.

1. calculate count,start,x,y endKey

2. visited, queue

visited[startX][startY][0] = true

3. bfs

```
if (inBound(myGrid, nextX, nextY) && myGrid[nextX][nextY] != '#')
```

nextChar in 'a' .. 'f' -> calculate nextstate

nextChar in 'A' .. 'F' ->

```
val currentLockIndex = nextChar - 'A'
```

```
val unlock = 1.and(current[2].shr(currentLockIndex))
```

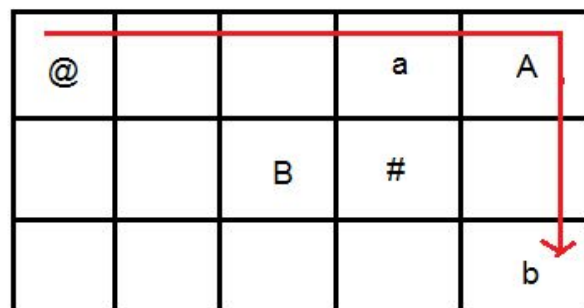
nextChar '.' ->

1 1 0 0
d c b a

c ? == 2

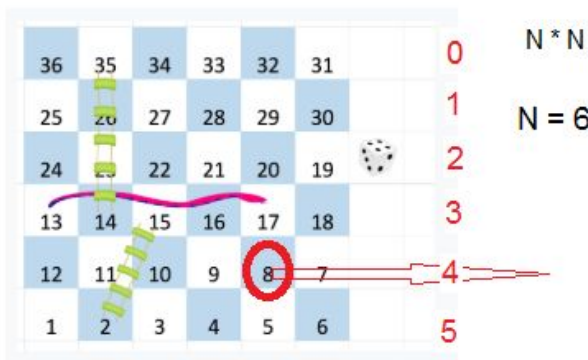
1 1 0 0 >> 2
1 1
& 1

= 1



909. Snakes and Ladders

<https://leetcode.com/problems/snakes-and-ladders/>


$$N^*N$$
$$N = 6$$

row = 4

column = 4

$$8^{-1} / 6 = 1$$

row % 2 == N % 2 ?

$$N - 1 - 1 = 4$$

true N - 1 - (8 - 1) % 6

$$\text{row} = N - (\text{number} - 1) / N - 1$$

false (8 - 1) % 6

1. visited, queue

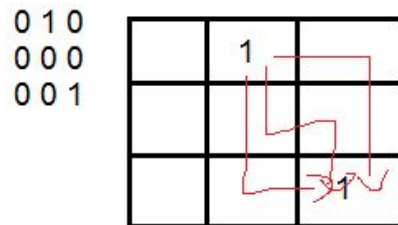
each time, we can move current + 6

- ## 2. BFS

if have a ladder or a snake (next != -1), then go to destination.

934. Shortest Bridge

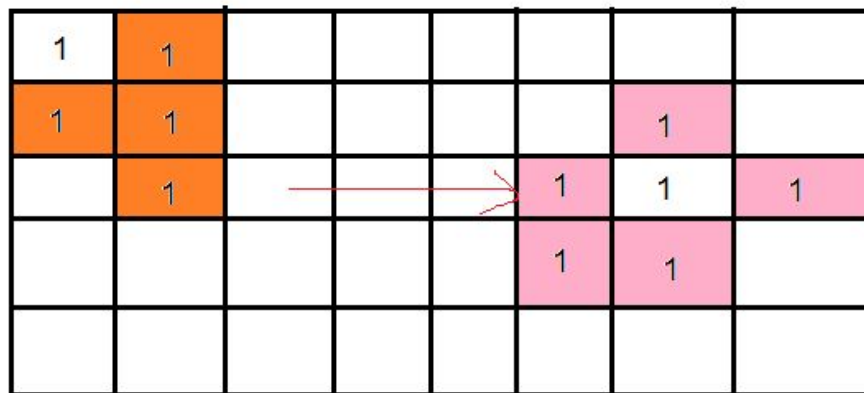
<https://leetcode.com/problems/shortest-bridge/>



1. find one island, then mark whole island visited, add ~~those cells that~~
~~surrounded with 0~~ into the queue all

2. BFS until a level that reached a cell '1', return level

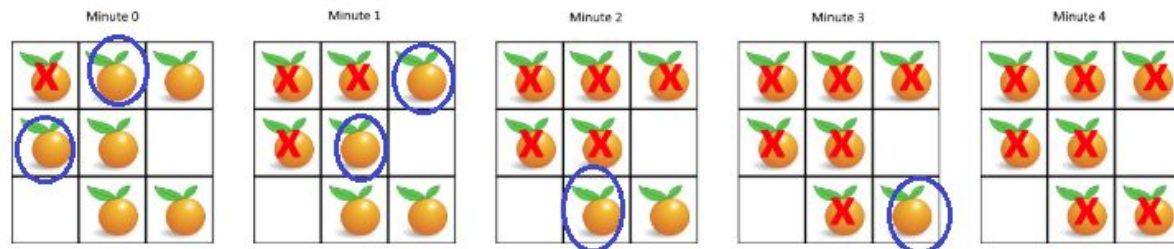
1 1 1 1 1
1 0 0 0 1
1 0 1 0 1
1 0 0 0 1
1 1 1 1 1



994. Rotting Oranges

<https://leetcode.com/problems/rotting-oranges/>

Example 1:



count fresh oranges

1. visited, queue add rotten oranges

2. bfs

if in bound, not visited, is a fresh orange, add it into the queue

if transform numbers == fresh oranges

return level

else return -1

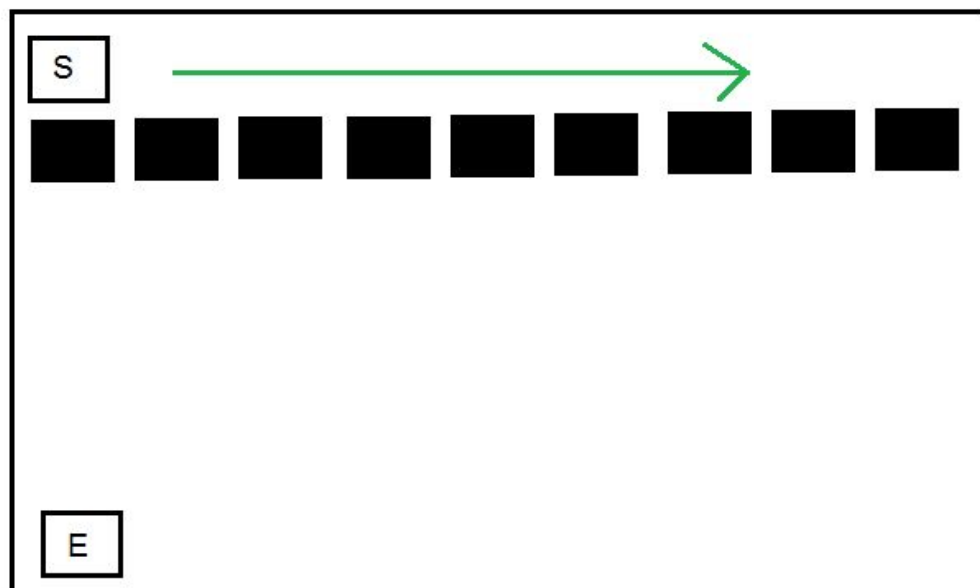
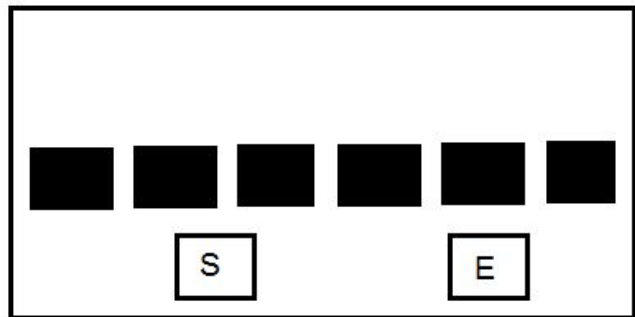
1036. Escape a Large Maze

<https://leetcode.com/problems/escape-a-large-maze/>

main idea is to check if source or target is surrounded by blocked cells.

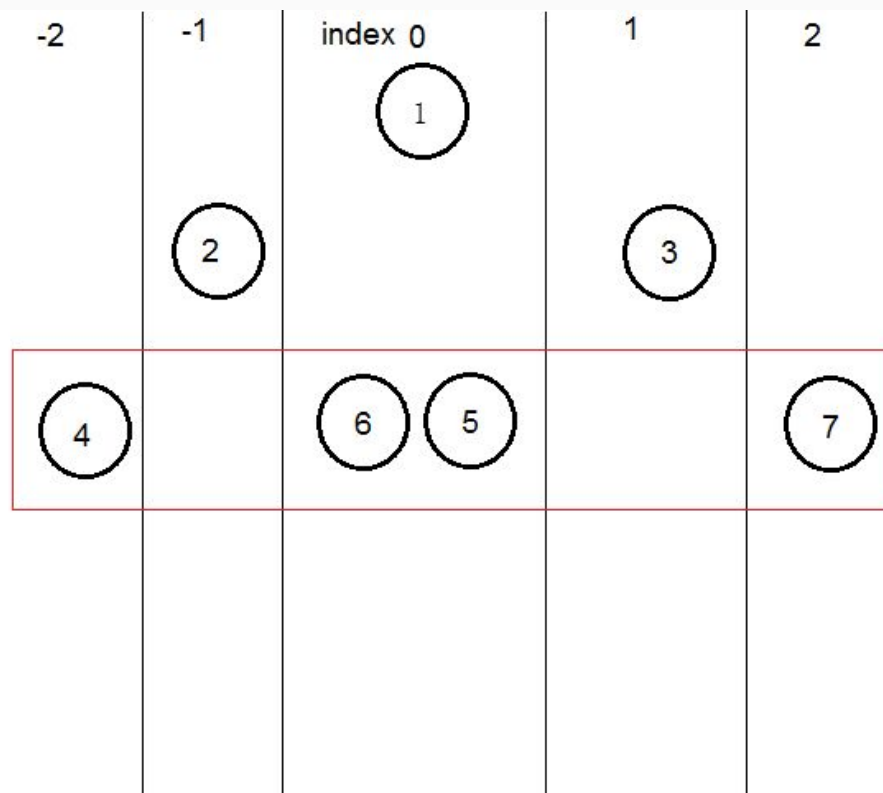
start with (source or target), can move blocked.length ?

notice that if source and target are surrounded with blocked cells but they are connected, return true



987. Vertical Order Traversal of a Binary Tree

<https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/>



idea: use breadth first search,
for every layer
maintain a map index -> list

at the end of each level, sort the list,
add to another map

at the end, sort the map by index, map to results.

-2 -> 4
0 -> 6,5 ==sort()==> 0 -> 5,6
2 -> 7

at the end of level,
add to map

-2 -> 4
0 -> 1,5,6
2 -> 7

1091. Shortest Path in Binary Matrix

<https://leetcode.com/problems/shortest-path-in-binary-matrix/>

Output: 4

0	→ 0	0
1	1	0
1	1	0

breadth first search

1. queue visited, if[0][0] == 1 return -1 else add it in queue and in visited

2. for each level,

for each cell, we have 8 next cells to check

if inBound
not visited
value == 0

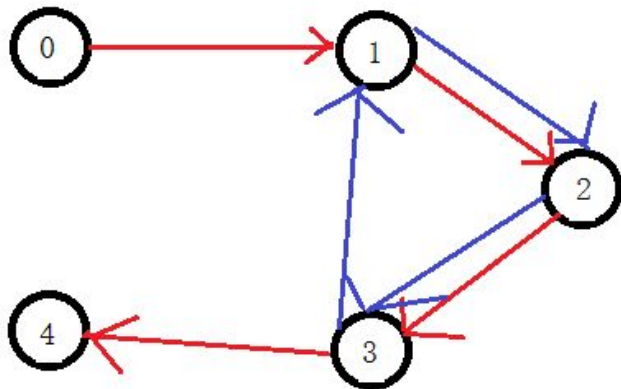
add next in queue and in visited

level start at 1

1129. Shortest Path with Alternating Colors

<https://leetcode.com/problems/shortest-path-with-alternating-colors/>

0 -> 1 1 -> 2 2 -> 3 3 -> 1 1 -> 2 2 -> 3 3 -> 4



1. convert array to two graphs

- BLUE, RED constant variables, why 0 1? visited is a 2 dimensions array so we can use it like `visited[index][color]`
- `resultArray`, set default value to -1, if we could access the index, we would update the value.

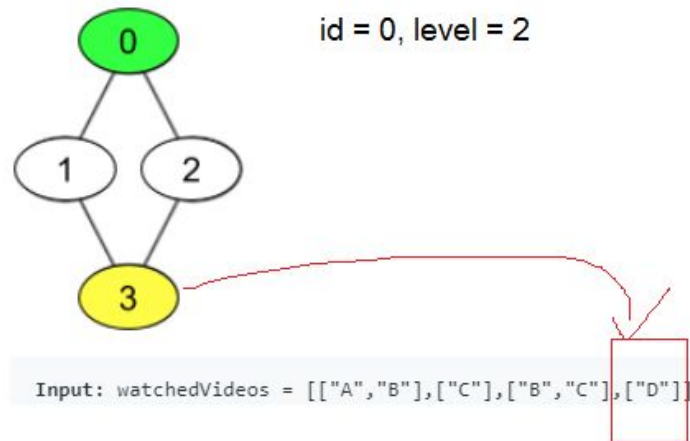
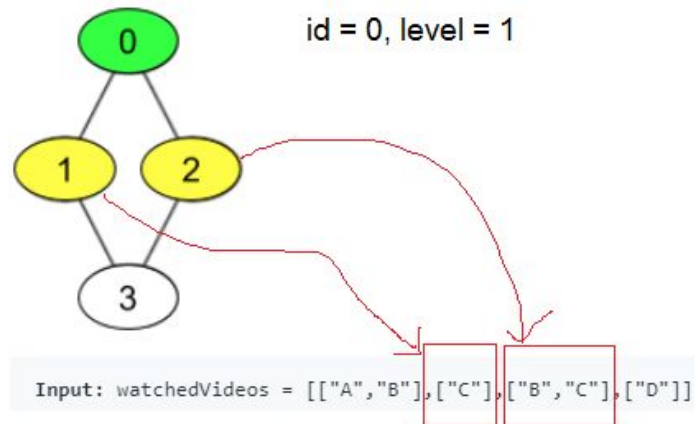
2. BFS template

- our state is `(index,color)`
- for each level, for each state, we update `resultArray` with current level,
- then we change color and check if we have met the state `(nextIndex, nextColor)` before

`resultArray = [0 1 2 3 7]`

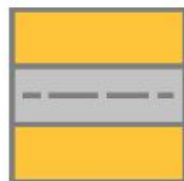
1311. Get Watched Videos by Your Friends

<https://leetcode.com/problems/get-watched-videos-by-your-friends/>

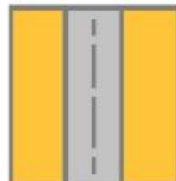


1391. Check if There is a Valid Path in a Grid

<https://leetcode.com/problems/check-if-there-is-a-valid-path-in-a-grid/>



Street 1



Street 2



Street 3



Street 4



Street 5



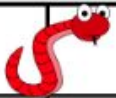
Street 6

- 1 which means a street connecting the left cell and the right cell.
- 2 which means a street connecting the upper cell and the lower cell.
- 3 which means a street connecting the left cell and the lower cell.
- 4 which means a street connecting the right cell and the lower cell.
- 5 which means a street connecting the left cell and the upper cell.
- 6 which means a street connecting the right cell and the upper cell.

```
private void next() {  
    final int nextX = x + deltaX[direction];  
    final int nextY = y + deltaY[direction];  
    if (inBound(grid, nextX, nextY) &&  
        !visited[nextX][nextY] &&  
        map.get(direction).contains(grid[nextX][nextY])  
    ) {  
        queue.offer(new int[]{nextX, nextY});  
        visited[nextX][nextY] = true;  
    }  
}
```

1210. Minimum Moves to Reach Target with Rotations

<https://leetcode.com/problems/minimum-moves-to-reach-target-with-rotations/>

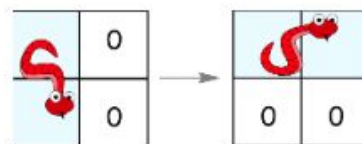
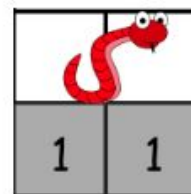
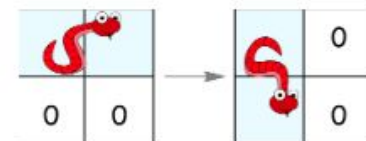
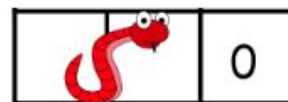
START					
		0	0	0	1
1	1	0	0	1	0
0	0	0	0	1	1
0	0	1	0	1	0
0	1	1	0	0	0
0	1	1	0	0	0
		FINISH			

for HORIZONTAL:

- go HORIZONTAL next
- go VERTICAL next
- rotate clockwise

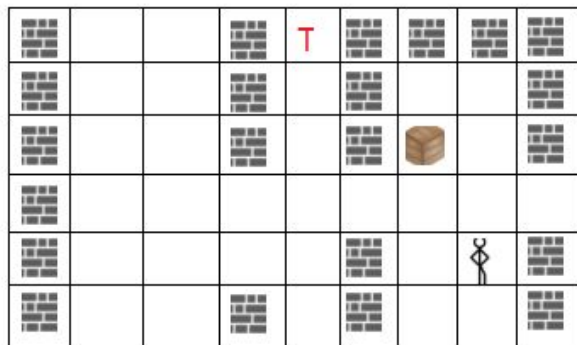
for VERTICAL:

- go HORIZONTAL next
- go VERTICAL next
- rotate counterclockwise

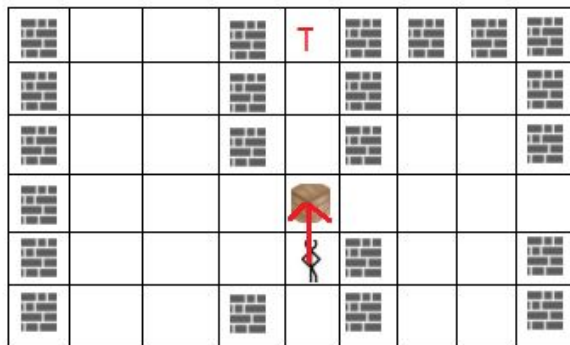
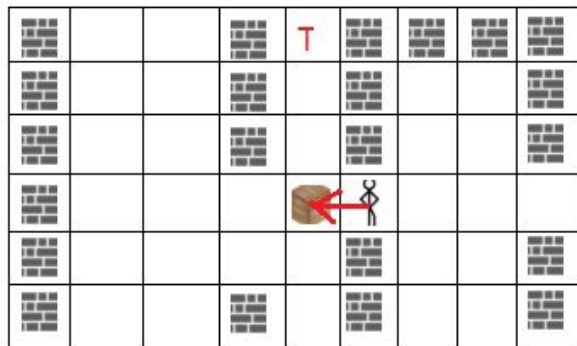


1263. Minimum Moves to Move a Box to Their Target Location

<https://leetcode.com/problems/minimum-moves-to-move-a-box-to-their-target-location/>



1. visited array is 3 dimensional, box x,y index and last move direction.
2. as we have 4 directions, when iterate, just use index as direction.
3. for each next move, we need to check next box position and next player position is in bound.
4. an additional bfs function to check the player can move the target player position or not.



1284. Minimum Number of Flips to Convert Binary Matrix to Zero Matrix

<https://leetcode.com/problems/minimum-number-of-flips-to-convert-binary-matrix-to-zero-matrix/>

[0 0]	[1 0]	[0 1]	[0 0]
[0 1]	[1 0]	[1 1]	[0 0]
0001	1010	0111	0000

```
val currentX = aimIndex / n  
val currentY = aimIndex % n
```

[0 1]
[2 3]
[4 5]

for element at bottom right, index in string = 5
so $x = 5/2 = 2$
 $y = 5\%2 = 1$

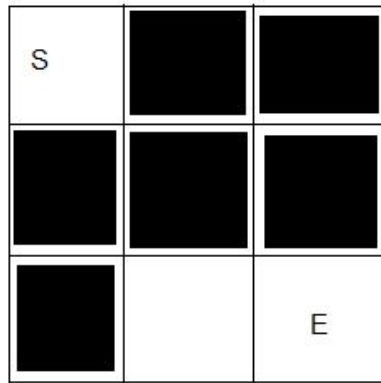
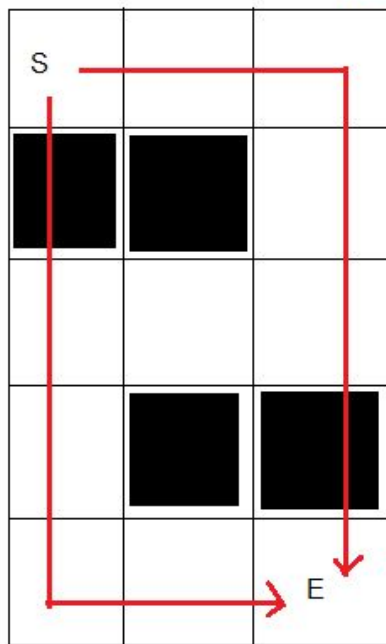
vice versa,
given a coordinate (2,1), index in string = $2*2 + 1 = 5$

```
private void change(StringBuilder stringBuilder, int i) {  
    if (stringBuilder.charAt(i) == '0') {  
        stringBuilder.setCharAt(i, '1');  
    } else {  
        stringBuilder.setCharAt(i, '0');  
    }  
}
```

1293. Shortest Path in a Grid with Obstacles Elimination

<https://leetcode.com/problems/shortest-path-in-a-grid-with-obstacles-elimination/>

K = 1



```
final boolean[][] visited =  
    new boolean[grid.length][grid[0].length][k + 1];
```

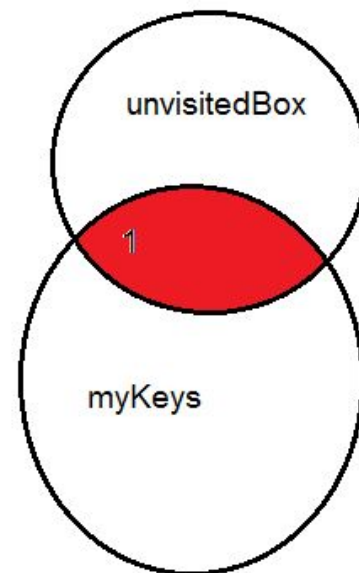
for each next cell, it should be in bound,

1. if next cell is 0, do a normal check visited
2. if next cell is 1, check if current $k \geq 1$, then check visited with $k-1$

1298. Maximum Candies You Can Get from Boxes

<https://leetcode.com/problems/maximum-candies-you-can-get-from-boxes/>

status	open		open	
candies	7	5	4	100
keys			1	
containedBoxes	1,2	3		



1345. Jump Game IV

<https://leetcode.com/problems/jump-game-iv/>

[illegible]

```
final Map<Integer, Set<Integer>> graph = new HashMap<>();
```

7 -> ...

last -> length -1

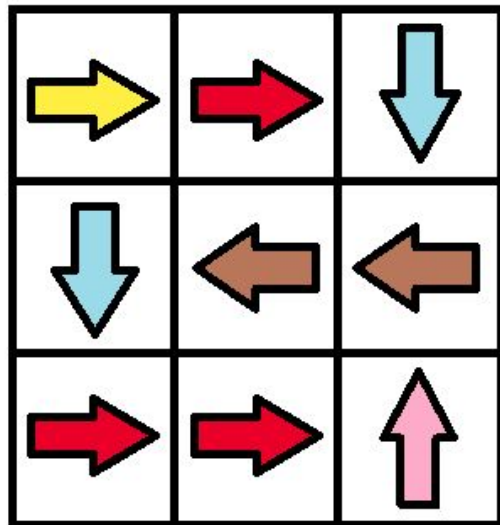
```

if (graph.containsKey(arr[current])) {
    graph.get(arr[current]).forEach(
        it -> {
            if (!visited[it]) {
                queue.offer(it);
                visited[it] = true;
            }
        }
    );
    // if not, time limit exceeded
    graph.remove(arr[current]);
}

```


1368. Minimum Cost to Make at Least One Valid Path in a Grid

<https://leetcode.com/problems/minimum-cost-to-make-at-least-one-valid-path-in-a-grid/>



idea is always check firstly the `stata(x,y,cost)` with the lowest cost in queue.

```
for (int index = 0; index < 4; index++) {  
    final int nextX = current[0] + deltaX[index];  
    final int nextY = current[1] + deltaY[index];  
    if (inBound(grid, nextX, nextY)) {  
        if (index + 1 == grid[current[0]][current[1]]) {  
            priorityQueue.offer(new int[]{nextX, nextY, current[2]});  
        } else {  
            priorityQueue.offer(new int[]{nextX, nextY, current[2] + 1});  
        }  
    }  
}
```

```
final PriorityQueue<int[]> priorityQueue = new PriorityQueue<>(  
    (Comparator.comparingInt(ints -> ints[2]))  
);
```

```
val priorityQueue = PriorityQueue { i1: IntArray, i2: IntArray -> compareValues(i1[2], i2[2]) }
```