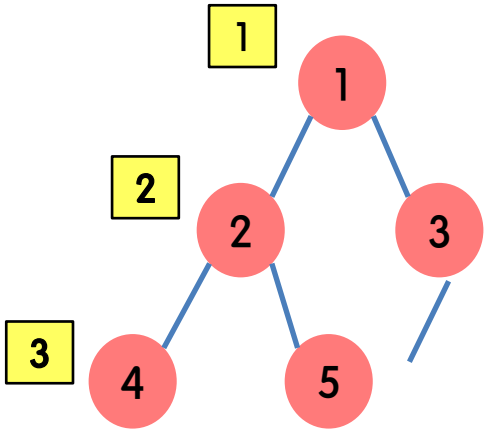
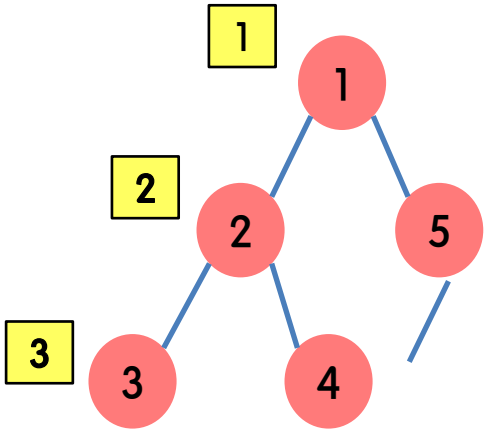
bfs dfs 개념 설명(시험에 나오는거 위주로)





Queue

Breadth First Search

근처에 있는것부터 찾는 알고리즘

(다익스트라, 페이스북)

Stack

Depth First Search

깊이 우선순위 탐색 ,알파고

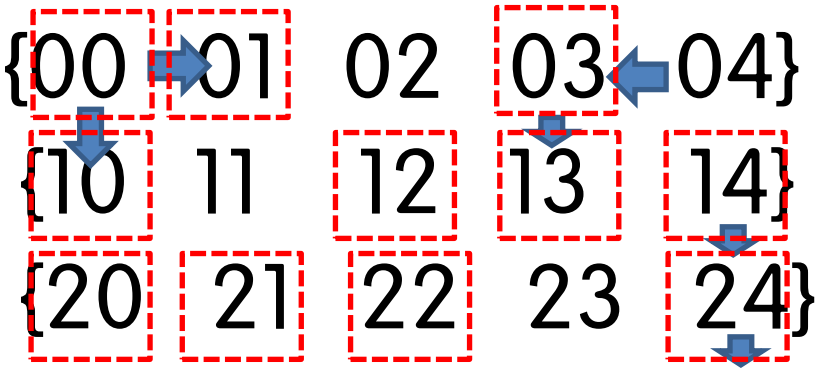
1. Basic_BFS

Problem

```
char[][] grid = {
{ '1', '1', '0', '0', '1' },
{ '1', '1', '0', '0', '0' },
{ '0', '0', '0', '0', '0' },
{ '0', '0', '0', '1', '1' } };
```

출력 :3

Example



1. Queue 방식

1. Basic_BFS

외워야할 부분

1 맞는 조건을 찾아내는 부분

```
for (int i = 0; i < grid.length; i++) {
    for (int j = 0; j < grid[i].length; j++) {
        if (grid[i][j] == '1') {
            count++;
            bfs(grid, i, j);
        }
    }
}
```

2 방향 설정 & 이차원배열 사이즈

```
int m, n;
m = grid.length;
n = grid[0].length;
int[][] dirs = { { -1, 0 }, { 1, 0 }, { 0, 1 }, { 0, -1 } };
```

1. Basic_BFS

외워야할 부분

3 큐에서 빼네는 부분

4 조건체크해서 큐에 넣는 부분(&&)
1. 마이너스 좌표체크 2. m*n 범위체크 3. grid[x][y] 값체크(문제제시값)

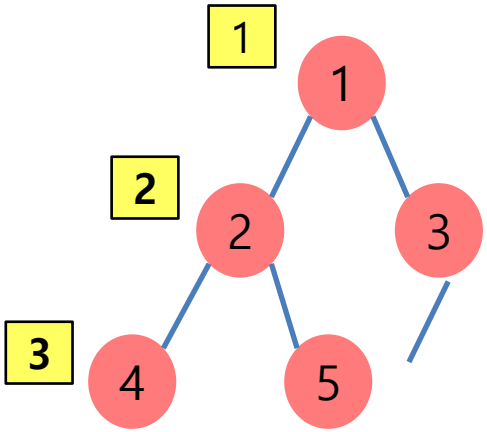
```
while (!queue.isEmpty()) {
        int[] point = queue.poll();
        for (int[] dir : dirs) {
                int x1 = point[0] + dir[0];
                int y1 = point[1] + dir[1]
                if (x1 >= 0 && y1 >= 0 && x1 < grid.length && y1 < grid[0].length &&
                   grid[x1][y1] == '1') {
                grid[x1][y1] = '0';

                }
      }
}
```

2. BFS

Problem

Input:



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

Queue bfs
Stack dfs

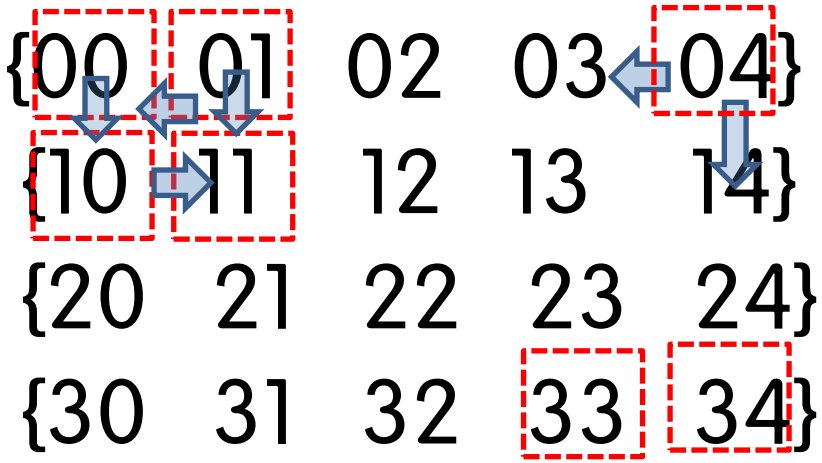
3. Basic_DFS

Problem

```
char[][] grid = {
{ '1', '1', '0', '0', '1' },
{ '1', '1', '0', '0', '0' },
{ '0', '0', '0', '0', '0' },
{ '0', '0', '0', '1', '1' } };
```

결과값 : 3

Example



1.파고들기 stack

3. Basic_DFS

외워야할 부분

1 맞는 조건을 찾아내는 부분

```
for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (grid[i][j] == '1') {
            count++;
            merge(grid, i, j);
            }
        }
}
```

2 방향 설정 & 이차원배열 사이즈

```
int m, n;
m = grid.length;
n = grid[0].length;
int[][] dirs = { { -1, 0 }, { 1, 0 }, { 0, 1 }, { 0, -1 } };
```

3. Basic_DFS

외워야할 부분

3 재귀를 이용한다(stack개념)

4 조건체크해서 큐에 넣는 부분(II)
1. 마이너스 좌표체크 2. m*n 범위체크 3. grid[x][y] 값체크(문제제시값)

```
public void merge(char[][] grid, int i, int j) {
	if (i < 0 || i >= m || j < 0 || j >= n || grid[i][j] != '1')
		return;
	grid[i][j] = 'X';
	for(int[] dir: dirs) {
		merge(grid, i+dir[0], j+dir[1]);
	}
}
```