Question: https://leetcode.com/problems/shortest-bridge/

So first identify an island using DFS, and add the island nodes to the queue.  
Second use queue nodes level by level to search for an unvisited 1 using BFS.

Code:

public class Pair{

int i;

int j;

Pair(int i, int j){

this.i=i;

this.j=j;

}

}

class Solution {

int[][] direction = {{-1, 0}, {0,1}, {1,0}, {0,-1}};

public int shortestBridge(int[][] grid) {

Queue<Pair> queue = new LinkedList<Pair>();

boolean vis[][] = new boolean [grid.length][grid[0].length], flag=false;

for(int i=0; i<grid.length && !flag; i++){

for(int j=0; j<grid[0].length && !flag; j++){

//find for any node of any island

if(grid[i][j]==1){

//use DFS to store all node of the island in queue

dfs(i, j, queue, vis, grid);

flag = true;

}

}

}

//use BFS to find the next unvisited 1 node

int level = 0;

while(queue.size()>0){

int size = queue.size();

// this while loop itterate at each level i.e., measure distance at each level

while(size-->0){

Pair n = queue.poll();

int i=n.i, j=n.j;

for(int k=0; k<4; k++){

int row=i+direction[k][0];

int col=j+direction[k][1];

if(row<0 || col<0 || row>=vis[0].length || col>=vis.length || vis[row][col]==true)

continue;

if(grid[row][col]==1) return level;

queue.offer(new Pair(row, col));

vis[row][col] = true;

}

}

level++;

}

return -1;

}

public void dfs(int i, int j, Queue<Pair> queue, boolean vis[][], int[][] grid){

queue.offer(new Pair(i, j));

vis[i][j]=true;

for(int k=0; k<4; k++){

int row=i+direction[k][0];

int col=j+direction[k][1];

if(row<0 || col<0 || row>=vis[0].length || col>=vis.length || vis[row][col]==true || grid[row][col]==0) continue;

dfs(row, col, queue, vis, grid);

}

}

}

Github Link :<https://lnkd.in/ecwtJeaz>