934. Shortest Bridge

You are given an n x n binary matrix grid where 1 represents land and 0 represents water.

An **island** is a 4-directionally connected group of 1's not connected to any other 1's. There are **exactly two islands** in grid.

You may change 0's to 1's to connect the two islands to form **one island**.

Return *the smallest number of* 0*'s you must flip to connect the two islands*.

Example 1:

Input: grid = [[0,1],[1,0]]

Output: 1

Example 2:

Input: grid = [[0,1,0],[0,0,0],[0,0,1]]

Output: 2

Example 3:

Input: grid = [[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]]

Output: 1

Constraints:

- n == grid.length == grid[i].length
- 2 <= n <= 100
- grid[i][j] is either 0 or 1.
- There are exactly two islands in grid.

```
class Solution {
    private:
    void dfs(int i,int j,vector<vector<int>>&
grid,vector<vector<int>>& vis,int n,int dr[],int dc[],
    queue<vector<int>>& pq){
        vis[i][j]=1;
        grid[i][j]=2;
        pq.push({0,i,j});
        for(int k=0;k<4;k++){
            int nrow=i+dr[k];
            int ncol=j+dc[k];
            if(nrow>=0 && ncol>=0 && nrow<n && ncol<n &&
grid[nrow][ncol] && !vis[nrow][ncol]){
                dfs(nrow,ncol,grid,vis,n,dr,dc,pq);
            }
        }
    }
public:
    int shortestBridge(vector<vector<int>>& grid) {
        int n=grid.size();
        vector<vector<int>> vis(n,vector<int>(n,0));
        queue<vector<int>> q;
        int dr[]={-1,0,1,0};
        int dc[]={0,1,0,-1};
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(grid[i][j] && !vis[i][j]){
                    dfs(i,j,grid,vis,n,dr,dc,q);
                    goto x; //to break nested for loops;
                }
            }
        }
        x:
        while(!q.empty()){
            int row=q.front()[1];
            int col=q.front()[2];
            int lvl=q.front()[0];
            q.pop();
            for(int i=0;i<4;i++){
                int nrow=row+dr[i];
                int ncol=col+dc[i];
                if(nrow>=0 && ncol>=0 && nrow<n && ncol<n){
                    if(grid[nrow][ncol]==1) return lvl;
                    else if(!vis[nrow][ncol] &&
grid[nrow][ncol]==0){
                        vis[nrow][ncol]=1;
                        q.push({lvl+1,nrow,ncol});
                    }
                }
            }
        }
        return 0;
    }
};
```