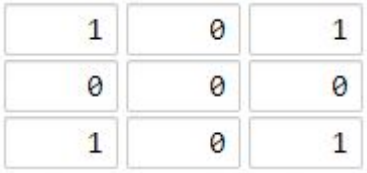
1162. As Far from Land as Possible

Given an `n x n` `grid` containing only values `0` and `1`, where `0` represents water and `1` represents land, find a water cell such that its distance to the nearest land cell is maximized, and return the distance. If no land or water exists in the grid, return `-1`.

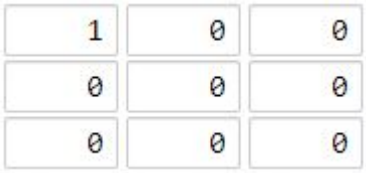
The distance used in this problem is the Manhattan distance: the distance between two cells `(x0, y0)` and `(x1, y1)` is `|x0 - x1| + |y0 - y1|`.

Example 1:




```
Input: grid = [[1,0,1],[0,0,0],[1,0,1]]
Output: 2
Explanation: The cell (1, 1) is as far as possible from all the land with
distance 2.
```

Example 2:




```
Input: grid = [[1,0,0],[0,0,0],[0,0,0]]
Output: 4
Explanation: The cell (2, 2) is as far as possible from all the land with
distance 4.
``````python
def maxDistance(self, mat: List[List[int]]) -> int:
        queue = []
        for i in range(len(mat)):
            for j in range(len(mat[0])):
                if mat[i][j]==1:
                    queue.append((i,j))

        directions = [(-1,0),(0,1),(1,0),(0,-1)]
```
```

```
        if len(queue)==0 or len(queue)==len(mat)*len(mat[0]):
            return -1
        dis = -1
        while len(queue):
            length = len(queue)
            dis = dis+1
            while length>0:
                x,y = queue.pop(0)
                for dx,dy in directions:
                    dxx = x+dx
                    dyy = y+dy
                    if dxx>=0 and dyy>=0 and dxx<len(mat) and
dyy<len(mat[0]) and mat[dxx][dyy]==0:
                        queue.append((dxx,dyy))
                        mat[dxx][dyy] = 1
                length = length-1
        return dis
```