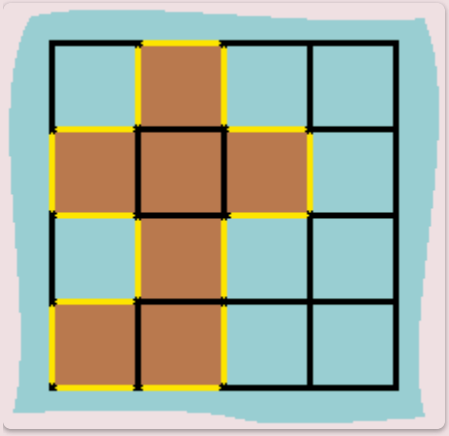
Yunyeong-kim

Question

You are given `row x col` `grid` representing a map where `grid[i][j] = 1` represents land and `grid[i][j] = 0` represents water.
Grid cells are connected horizontally/vertically (not diagonally).
The `grid` is completely surrounded by water, and there is exactly one island (i.e., one or more connected land cells).
The island doesn't have "lakes", meaning the water inside isn't connected to the water around the island.
One cell is a square with side length 1.
The grid is rectangular, width and height don't exceed 100. Determine the perimeter of the island.

Base Image



Example

Input: grid = [[0,1,0,0],[1,1,1,0],[0,1,0,0],[1,1,0,0]]
Output: 16
Explanation: The perimeter is the 16 yellow stripes in the image above.

Example

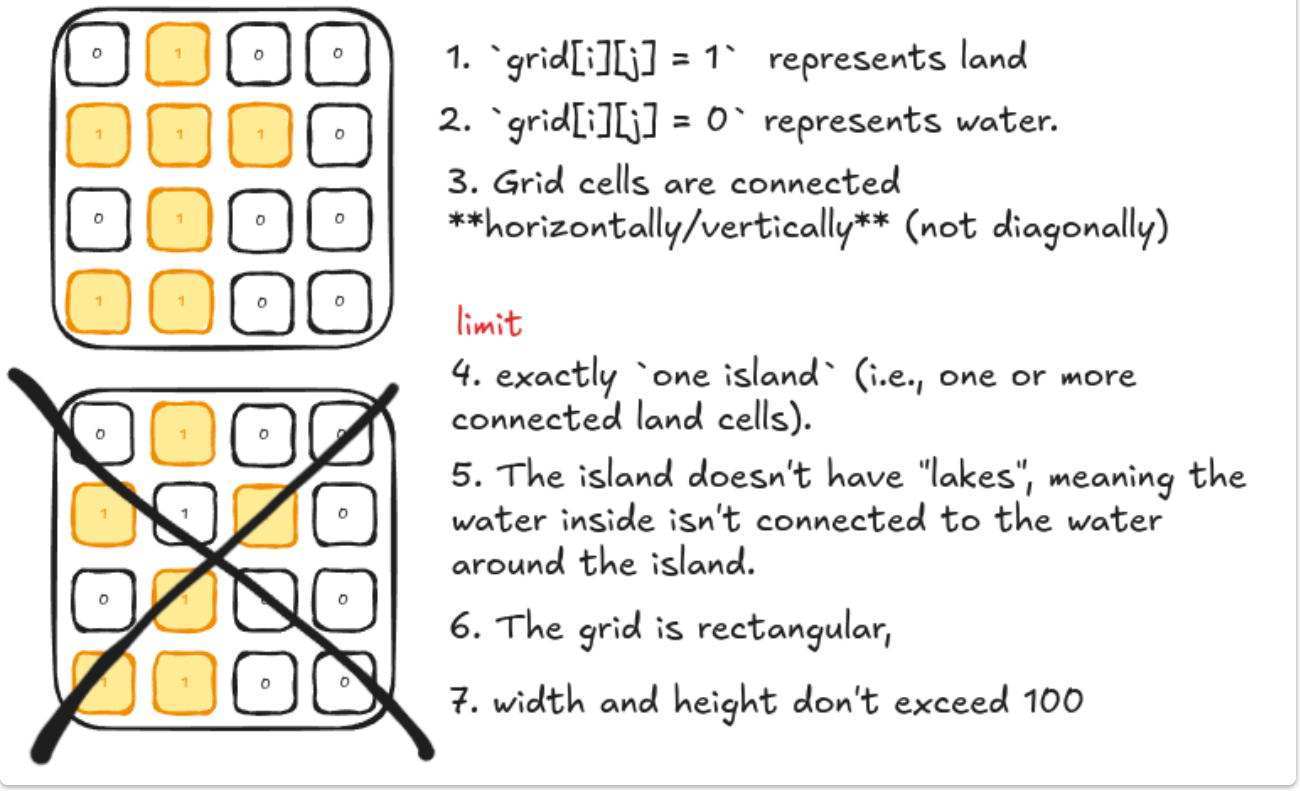
Input: grid = 1
Output: 4

Example

Input: grid = 1,0
Output: 4

Definition

1. `grid[i][j] = 1` represents land
2. `grid[i][j] = 0` represents water.
3. Grid cells are connected **horizontally/vertically** (not diagonally)
4. exactly `one island` (i.e., one or more connected land cells).
5. The island doesn't have "lakes", meaning the water inside isn't connected to the water around the island.
6. The grid is rectangular,
7. width and height don't exceed 100

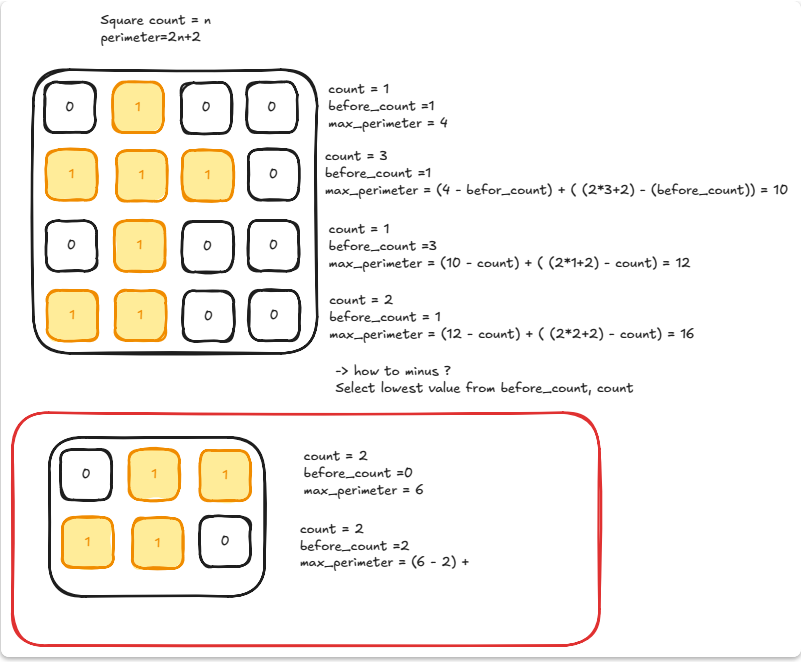


First Solution

```
class Solution(object):
    def islandPerimeter(self, grid):
        max_perimeter = 0
        perimeter = 0
        before_count = 0
        for i in range(len(grid)):
            count = 0
            for j in range(len(grid[i])):
                if grid[i][j] == 1:
                    count += 1
            if count != 0:
                perimeter = 2*count + 2
                if i == 0 :
                    max_perimeter = perimeter
                elif before_count < count:
                    max_perimeter = (max_perimeter - before_count) +
(perimeter - before_count)
                else:
                    max_perimeter = (max_perimeter - count) + (perimeter -
count)
                print(f"max = {max_perimeter}")
                before_count = count
        return max_perimeter

# Failed.
```

- It didn't expect cols align.
- `[[0,1,1],[1,1,0]]`

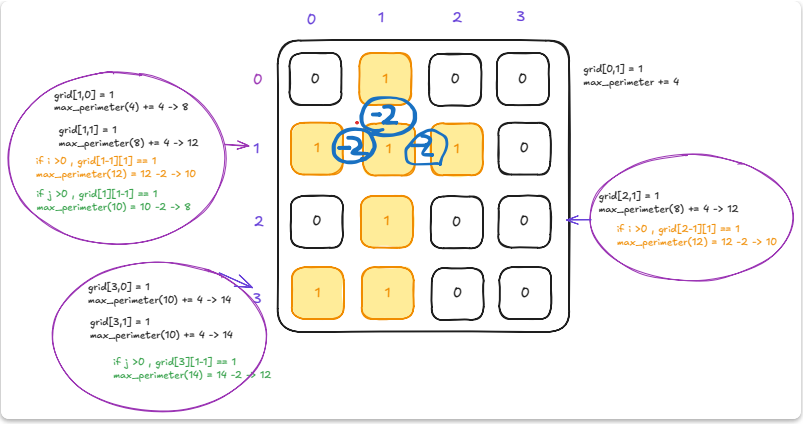
Firstcode explain



Second try

```python
    def islandPerimeterSecond(self, grid):
        max_perimeter = 0
        for i in range(len(grid)):
            for j in range(len(grid[i])):
                # base case is 4
                if grid[i][j] == 1:
                    max_perimeter += 4
                    # if square exist above , remove 2
                    if i > 0 and grid[i-1][j] == 1:
                        max_perimeter -= 2

                    # if square exist before remove -2
                    if j > 0 and grid[i][j-1] == 1:
                        max_perimeter -= 2

        return max_perimeter

#O(M*N) 99.86% / O(1)
```

Second code explain

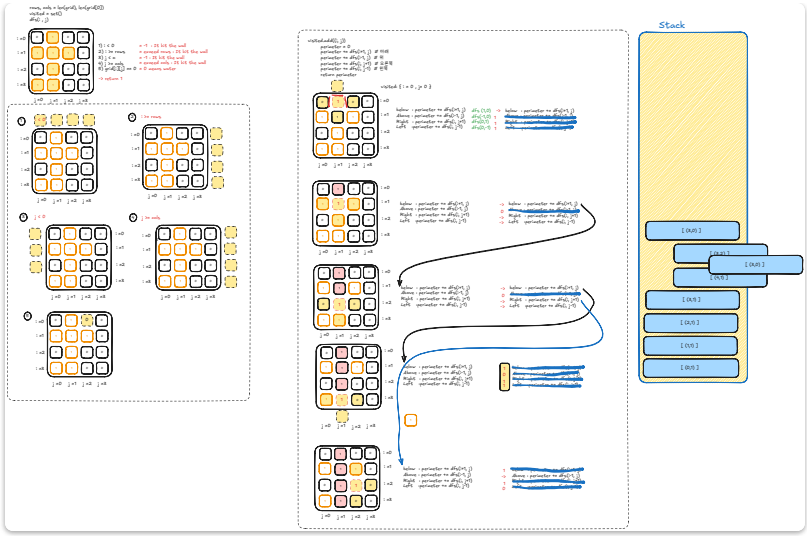


Solution (with other way)

- counting 0 , empty spaces.
- DFS Search.

```
class Solution:
    def islandPerimeter(self, grid):
        rows, cols = len(grid), len(grid[0])
        visited = set()

        def dfs(i, j):
            if (i < 0 or i >= rows or j < 0 or j >= cols or grid[i][j] ==
0):
                return 1  # 물에 닿았으면 perimeter +1
            if (i, j) in visited:
                return 0  # 이미 방문한 경우

            visited.add((i, j))
            perimeter = 0
            perimeter += dfs(i+1, j)  # 아래
            perimeter += dfs(i-1, j)  # 위
            perimeter += dfs(i, j+1)  # 오른쪽
            perimeter += dfs(i, j-1)  # 왼쪽
            return perimeter

        # 섬이 하나뿐이므로 DFS를 시작할 위치를 찾음
        for i in range(rows):
            for j in range(cols):
                if grid[i][j] == 1:
                    return dfs(i, j)  # 섬 찾으면 DFS 실행

        return 0  # 만약 섬이 없다면 0 반환
```

Solution explain


Stack
[0,0]
[0,1]
[0,2]
[0,3]
[0,4]
[0,5]
[0,6]
[0,7]
[0,8]