<https://leetcode.com/problems/island-perimeter/description/>

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.

**Example 1:**



**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 2:**

**Input:** grid = [[1]]

**Output:** 4

**Example 3:**

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

**Output:** 4

**Constraints:**

row == grid.length

col == grid[i].length

1 <= row, col <= 100

grid[i][j] is 0 or 1.

There is exactly one island in grid.

**Attempt 1: 2024-12-21**

**Solution 1: Math (30 min)**

class Solution {

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

        int m = grid.length;

        int n = grid[0].length;

        int result = 0;

        for(int i = 0; i < m; i++) {

            for(int j = 0; j < n; j++) {

                // If the current cell is land

                if(grid[i][j] == 1) {

                    // Add 4 for the cell

                    result += 4;

                    // Subtract 2 for each adjacent land cell (to avoid double counting shared edges)

                    if(i > 0 && grid[i - 1][j] == 1) {

                        result -= 2;

                    }

                    if(j > 0 && grid[i][j - 1] == 1) {

                        result -= 2;

                    }

                }

            }

        }

        return result;

    }

}

Time Complexity: O(m \* n)

Space Complexity: O(1)

**Refer to chatGPT**

**Explanation:**

Iterate through the grid:

Traverse each cell in the grid using a nested loop.

For every land cell (1), initially add 4 to the perimeter since each land cell contributes 4 edges to the perimeter.

Check for shared edges:

If there is a land cell above (grid[i - 1][j] == 1), subtract 2 from the perimeter (as the edge between these two cells is shared and should not be double-counted).

If there is a land cell to the left (grid[i][j - 1] == 1), subtract 2 from the perimeter for the same reason.

Return the total perimeter:

After processing all the cells, the perimeter variable holds the final result.

**Why in 'check for shared edges' only consider 'a land cell above' and 'a land cell to the left' ? should we also consider 'a land cell below' or 'a land cell to the right' ?**

Good question! In the solution, we only check the **land cell above** and the **land cell to the left** because doing so is sufficient to handle all cases without double-counting. Let me explain why we do not need to explicitly check **land cells below** or **to the right**.

**Key Insight:**

The goal is to calculate the shared edges **exactly once**, without repeating work. By processing the grid row by row and column by column in a top-to-bottom, left-to-right order:

Any shared edge between two land cells will either:

Be handled when the **lower cell checks the cell above it**.

Be handled when the **right cell checks the cell to its left**.

Thus, **shared edges below and to the right** are automatically considered during the iteration, as they are covered when the corresponding neighboring cell processes the relationship.

**Example Walkthrough:**

**Grid:**

1 1

1 0

**Step-by-Step:**

Start iterating through the grid from top-left (row 0, col 0).

Cell (0,0):

It is land (1), so add 4 to the perimeter.

No cell above or to the left (boundary).

**Perimeter = 4**

Cell (0,1):

It is land (1), so add 4 to the perimeter.

Shared edge with the left cell (0,0): subtract 2.

**Perimeter = 4 + 4 - 2 = 6**

Cell (1,0):

It is land (1), so add 4 to the perimeter.

Shared edge with the cell above (0,0): subtract 2.

**Perimeter = 6 + 4 - 2 = 8**

Cell (1,1):

It is water (0), so skip it.

**Final Perimeter = 8**

**Observations:**

The **shared edge between (0,0) and (0,1)** was handled when processing **(0,1)** (the cell to the right checks its left).

The **shared edge between (0,0) and (1,0)** was handled when processing **(1,0)** (the cell below checks its top).

**Why Avoid "Below" and "Right"?**

If we also checked the cells **below** or **to the right**, we'd be counting the same shared edges multiple times. For instance:

The edge between (0,0) and (0,1) would be subtracted both when processing (0,0) and (0,1), leading to incorrect results.

**Conclusion:**

By considering **only the land cell above and to the left**, we ensure:

Each shared edge is counted exactly once.

The solution remains efficient, as we avoid redundant checks.

**Refer to**

[L1568.Minimum Number of Days to Disconnect Island (Ref.L200)](note://WEBe1f9dea2f2571376df84b64404041136)