Problem 1536: Minimum Swaps to Arrange a Binary Grid

Problem Information

Difficulty: Medium
Acceptance Rate: 48.96%
Paid Only: No
Tags: Array, Greedy, Matrix

Problem Description

Given an `n x n` binary `grid`, in one step you can choose two **adjacent rows** of the grid and swap them.

A grid is said to be **valid** if all the cells above the main diagonal are **zeros**.

Return _the minimum number of steps_ needed to make the grid valid, or **-1** if the grid cannot be valid.

The main diagonal of a grid is the diagonal that starts at cell `(1, 1)` and ends at cell `(n, n)`.

Example 1:

![](https://assets.leetcode.com/uploads/2020/07/28/fw.jpg)

Input: grid = [[0,0,1],[1,1,0],[1,0,0]] **Output:** 3

Example 2:

![](https://assets.leetcode.com/uploads/2020/07/16/e2.jpg)

Input: grid = [[0,1,1,0],[0,1,1,0],[0,1,1,0],[0,1,1,0]] **Output:** -1 **Explanation:** All rows are similar, swaps have no effect on the grid.

Example 3:

![](https://assets.leetcode.com/uploads/2020/07/16/e3.jpg)

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

Constraints:

* `n == grid.length` `== grid[i].length` * `1 <= n <= 200` * `grid[i][j]` is either `0` or `1`

Code Snippets

C++:

```
class Solution {
public:
int minSwaps(vector<vector<int>>& grid) {

}
};
```

Java:

```
class Solution {
public int minSwaps(int[][] grid) {

}
}
```

Python3:

```
class Solution:
def minSwaps(self, grid: List[List[int]]) -> int:
```