Problem 3742: Maximum Path Score in a Grid

Problem Information

Difficulty: Medium
Acceptance Rate: 36.05%
Paid Only: No
Tags: Array, Dynamic Programming, Matrix

Problem Description

You are given an `m x n` grid where each cell contains one of the values 0, 1, or 2. You are also given an integer `k`.

You start from the top-left corner `(0, 0)` and want to reach the bottom-right corner `(m - 1, n - 1)` by moving only **right** or **down**.

Each cell contributes a specific score and incurs an associated cost, according to their cell values:

* 0: adds 0 to your score and costs 0. * 1: adds 1 to your score and costs 1. * 2: adds 2 to your score and costs 1.

Return the **maximum** score achievable without exceeding a total cost of `k`, or -1 if no valid path exists.

Note: If you reach the last cell but the total cost exceeds `k`, the path is invalid.

Example 1:

Input: grid = [[0, 1],[2, 0]], k = 1

Output: 2

Explanation:

The optimal path is:

Cell | grid[i][j] | Score | Total Score | Cost | Total Cost ---|---|---|---|---|--- (0, 0) | 0 | 0 | 0 | 0 | 0 (1, 0) | 2 | 2 | 2 | 1 | 1 (1, 1) | 0 | 0 | 2 | 0 | 1 Thus, the maximum possible score is 2.

Example 2:

Input: grid = [[0, 1],[1, 2]], k = 1

Output: -1

Explanation:

There is no path that reaches cell `(1, 1)`■■■■■■■ without exceeding cost k. Thus, the answer is -1.

Constraints:

* `1 <= m, n <= 200` * `0 <= k <= 103■■■■■■■` * `■■■■■■■grid[0][0] == 0` * `0 <= grid[i][j] <= 2`

Code Snippets

C++:

```
class Solution {
public:
int maxPathScore(vector<vector<int>>& grid, int k) {

}
};
```

Java:

```
class Solution {
public int maxPathScore(int[][] grid, int k) {

}
}
```

Python3:

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