2017. Grid Game

Medium | Topics | Companies | Hint

You are given a **0-indexed** 2D array `grid` of size `2 x n`, where `grid[r][c]` represents the number of points at position `(r, c)` on the matrix. Two robots are playing a game on this matrix.

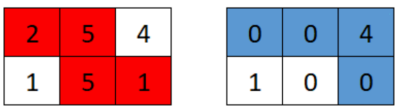
Both robots initially start at `(0, 0)` and want to reach `(1, n-1)`. Each robot may only move to the **right** (`(r, c)` to `(r, c + 1)`) or **down** (`(r, c)` to `(r + 1, c)`).

At the start of the game, the **first** robot moves from `(0, 0)` to `(1, n-1)`, collecting all the points from the cells on its path. For all cells `(r, c)` traversed on the path, `grid[r][c]` is set to `0`. Then, the **second** robot moves from `(0, 0)` to `(1, n-1)`, collecting the points on its path. Note that their paths may intersect with one another.

The **first** robot wants to **minimize** the number of points collected by the **second** robot. In contrast, the **second** robot wants to **maximize** the number of points it collects. If both robots play **optimally**, return *the **number of points** collected by the **second** robot.*

All robots being greedy to maximize points collected.

Example 1:




```
Input: grid = [[2,5,4],[1,5,1]]
Output: 4
Explanation: The optimal path taken by the first robot is shown in red, and the optimal path taken by
the second robot is shown in blue.
The cells visited by the first robot are set to 0.
The second robot will collect 0 + 0 + 4 + 0 = 4 points.
```

```python
class Solution:
    def gridGame(self, grid: List[List[int]]) -> int:
        n = len(grid[0])

        def findBreakpoint():
            res = 0
            ssum = sum(grid[0][:res+1]) + sum(grid[1][res:])

            for i in range(n):
                temp = sum(grid[0][:i+1]) + sum(grid[1][i:])
                if temp > ssum:
                    res = i
                    ssum = temp
            print("break at: ", res)
            return res

        c = findBreakpoint()

        for i in range(n):
            if i == c:
                grid[0][i] = 0
                grid[1][i] = 0
            elif i < c:
                grid[0][i] = 0
            else:
                grid[1][i] = 0
        print(grid)

        nc = findBreakpoint()
        res = sum(grid[0][:nc+1]) + sum(grid[1][nc:])

        return res
```

Not working.

Accepted 109 / 109 testcases passed

Andr... submitted at Jan 21, 2025 10:03

Editorial | Solution

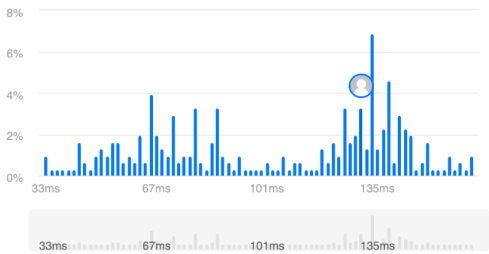
Runtime

130 ms | Beats 40.00%

Analyze Complexity

Memory

30.18 MB | Beats 32.79%



Code | Python3

```python
class Solution:
    def gridGame(self, grid: List[List[int]]) -> int:
        n = len(grid[0])

        # Prefix sums for top and bottom rows
        prefix_top = [0] * n
        prefix_bottom = [0] * n
```

```python
class Solution:
    def gridGame(self, grid: List[List[int]]) -> int:
        n = len(grid[0])

        # Prefix sums for top and bottom rows
        prefix_top = [0] * n
        prefix_bottom = [0] * n

        # Calculate prefix sums for top and bottom rows
        prefix_top[0] = grid[0][0]
        prefix_bottom[0] = grid[1][0]

        for i in range(1, n):
            prefix_top[i] = prefix_top[i - 1] + grid[0][i]
            prefix_bottom[i] = prefix_bottom[i - 1] + grid[1][i]

        # Total sum of each row
        total_top = prefix_top[-1]
        total_bottom = prefix_bottom[-1]

        # Find the minimum possible score for the second robot
        result = float('inf')
        for i in range(n):
            # Robot 2's options:
            # 1. Top row after the breakpoint
            score_top = total_top - prefix_top[i]
            # 2. Bottom row before the breakpoint
            score_bottom = prefix_bottom[i - 1] if i > 0 else 0

            # Robot 2's maximum score for this breakpoint
            robot2_score = max(score_top, score_bottom)

            # Minimize Robot 2's score
            result = min(result, robot2_score)

        return result
```

GPT 4o

prefix x 2