2017. Grid Game

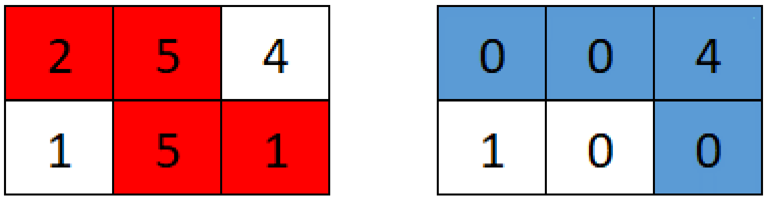
You are given a **0-indexed** 2D array `grid` of size `2 x n`, where `grid[r][c]` represents the number of points at position `(r, c)` on the matrix. Two robots are playing a game on this matrix.

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.

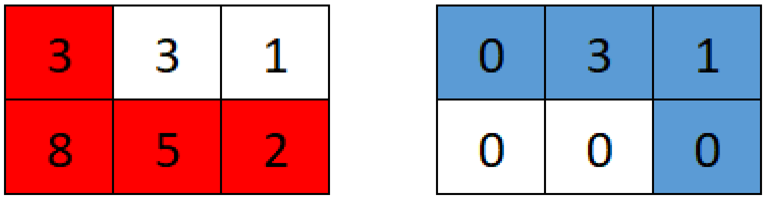
The **first** robot wants to **minimize** the number of points collected by the **second** robot. In contrast, the **second** robot wants to **maximize** the number of points it collects. If both robots play **optimally**, return *the **number of points** collected by the **second** robot.*

Example 1:



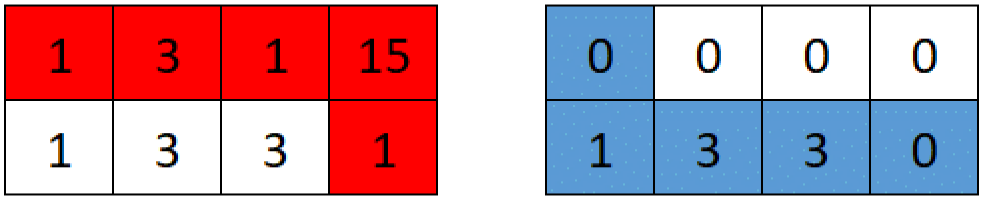
```
Input: grid = [[2,5,4],[1,5,1]]
Output: 4
Explanation: The optimal path taken by the first robot is shown in red, and the
optimal path taken by the second robot is shown in blue.
The cells visited by the first robot are set to 0.
The second robot will collect 0 + 0 + 4 + 0 = 4 points.
```

Example 2:



```
nput: grid = [[3,3,1],[8,5,2]]
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 + 3 + 1 + 0 = 4 points.
```

Example 3:



```
Input: grid = [[1,3,1,15],[1,3,3,1]]
Output: 7
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 + 1 + 3 + 3 + 0 = 7 points.
```

Constraints:

- `grid.length == 2`
- `n == grid[r].length`
- `1 <= n <= 5 * 104`
- `1 <= grid[r][c] <= 105`

Overview

We are given a matrix `grid` containing 2 rows and `n` columns. Each cell contains a value representing the number of points for that cell in the `grid`. Two robots are playing a game where they are initially positioned at `(0, 0)` and aim to reach `(1, n - 1)`.

Each robot can only move right or down in the grid. The task is to compute the points collected by the second robot, given the strategies of both robots.

The challenge is that the first robot moves first, and its goal is to reduce the points available for the second robot. The second robot then takes the best path to collect as many points as possible.

2017. Grid Game

```
/*
   Prefix sum+suffix sum+sliding path
   Time complexity: O(2n)
   Space compelxity: O(1)
*/

typedef long long ll;
class Solution{
   public:
      ll gridGame(std::vector<std::vector<int>>& grid) {
         int m=grid.size(); // m=2
         int n=grid[0].size();

         // Suffix sum for upper row
         ll upper_portion_sum=std::accumulate(grid[0].begin(),grid[0].end(),0LL);

         // Prefix sum for lower row
         ll lower_portion_sum=0;

         // Slide the path
         ll ans=LONG_LONG_MAX;
         for(int i=0;i<n;++i){
            // Substract the current value in the 1st row from the upper row's suffix sum
            upper_portion_sum-=grid[0][i];

            // Update the answer by minmizing the maximum of boths sums
            ans=std::min(ans,std::max(upper_portion_sum,lower_portion_sum));

            // Add the current value in the 2nd row to the lower row's prefix sum
            lower_portion_sum+=grid[1][i];
         }
         return ans;
      }
};
```