5931. Stamping the Grid

My Submissions (/contest/biweekly-contest-69/problems/stamping-the-grid/submissions/) Back to Contest (/contest/biweekly-contest-69/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Hard

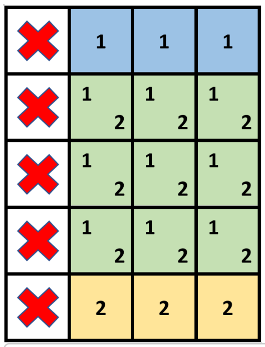
You are given an `m x n` binary matrix `grid` where each cell is either `0` (empty) or `1` (occupied).

You are then given stamps of size `stampHeight x stampWidth`. We want to fit the stamps such that they follow the given **restrictions** and **requirements**:

1. Cover all the **empty** cells.
2. Do not cover any of the **occupied** cells.
3. We can put as **many** stamps as we want.
4. Stamps can **overlap** with each other.
5. Stamps are not allowed to be **rotated**.
6. Stamps must stay completely **inside** the grid.

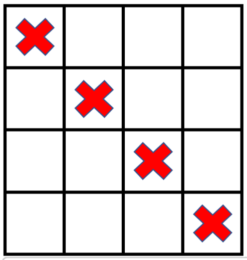
Return `true` *if it is possible to fit the stamps while following the given restrictions and requirements. Otherwise, return* `false`.

Example 1:



```
Input: grid = [[1,0,0,0],[1,0,0,0],[1,0,0,0],[1,0,0,0],[1,0,0,0]], stampHeight = 4, stampWidth = 3
Output: true
Explanation: We have two overlapping stamps (labeled 1 and 2 in the image) that are able to cover all the empty cells.
```

Example 2:



```
Input: grid = [[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]], stampHeight = 2, stampWidth = 2
Output: false
Explanation: There is no way to fit the stamps onto all the empty cells without the stamps going outside the grid.
```

Constraints:

- `m == grid.length`
- `n == grid[r].length`
- `1 <= m, n <=` 10^5
- `1 <= m * n <= 2 *` 10^5
- `grid[r][c]` is either `0` or `1`.
- `1 <= stampHeight, stampWidth <=` 10^5

JavaScript

```
const initialize2DArray = (n, m) => { let d = []; for (let i = 0; i < n; i++) { let t = Array(m).fill(0); d.push(t); } return d; };

const possibleToStamp = (g, h, w) => {
  let n = g.length, m = g[0].length, pre = preSum2D(g, n, m), visit = initialize2DArray(n + 1, m + 1);
  for (let i = 0; i + h <= n; i++) {
    for (let j = 0; j + w <= m; j++) {
      if (subMatrixSum(pre, i, i + h - 1, j, j + w - 1) == 0) {
        addMatrix(visit, i, i + h - 1, j, j + w - 1);
      }
    }
  }
  for (let i = 1; i < n; i++) {
    for (let j = 0; j < m; j++) {
      visit[i][j] += visit[i - 1][j];
    }
  }
  for (let i = 0; i < n; i++) {
    for (let j = 1; j < m; j++) {
      visit[i][j] += visit[i][j - 1];
    }
  }
  for (let i = 0; i < n; i++) {
    for (let j = 0; j < m; j++) {
      if (g[i][j] == 0 && visit[i][j] == 0) return false;
    }
  }
  return true;
};

const addMatrix = (visit, x1, x2, y1, y2) => {
  visit[x1][y1]++;
  visit[x2 + 1][y1]--;
  visit[x1][y2 + 1]--;
  visit[x2 + 1][y2 + 1]++;
};

const subMatrixSum = (pre, x1, x2, y1, y2) => {
  let res = pre[x2][y2];
  if (x1 > 0) res -= pre[x1 - 1][y2];
  if (y1 > 0) res -= pre[x2][y1 - 1];
  if (x1 > 0 && y1 > 0) res += pre[x1 - 1][y1 - 1];
  return res;
};

const preSum2D = (g, n, m) => {
  let pre = initialize2DArray(n, m);
  for (let i = 0; i < n; i++) {
    for (let j = 0; j < m; j++) {
      pre[i][j] = g[i][j];
    }
  }
  for (let i = 1; i < n; i++) {
    for (let j = 0; j < m; j++) {
      pre[i][j] += pre[i - 1][j];
    }
  }
  for (let i = 0; i < n; i++) {
    for (let j = 1; j < m; j++) {
      pre[i][j] += pre[i][j - 1];
    }
  }
  return pre;
};
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/615711519/) More Details (/submissions/detail/615711519/)

Share your acceptance!

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)