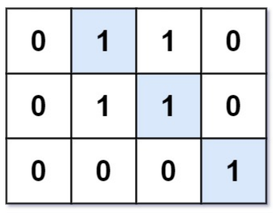
562. Longest Line of Consecutive One in Matrix

Medium 620 93 Add to List Share

Given an `m x n` binary matrix `mat`, return *the length of the longest line of consecutive one in the matrix*.

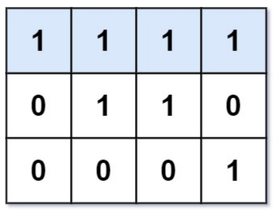
The line could be horizontal, vertical, diagonal, or anti-diagonal.

Example 1:




```
Input: mat = [[0,1,1,0],[0,1,1,0],[0,0,0,1]]
Output: 3
```

Example 2:




```
Input: mat = [[1,1,1,1],[0,1,1,0],[0,0,0,1]]
Output: 4
```

Constraints:

- `m == mat.length`
- `n == mat[i].length`
- `1 <= m, n <= 10^4`
- `1 <= m * n <= 10^4`
- `mat[i][j]` is either `0` or `1`.

Accepted 51,059 | Submissions 105,539

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months | 6 months ~ 1 year | 1 year ~ 2 years

Google | 7

Related Topics

Array Dynamic Programming Matrix

Hide Hint 1

One solution is to count ones in each direction separately and find the longest line. Don't you think it will take too much lines of code?

Hide Hint 2

Is it possible to use some extra space to make the solution simple?

Hide Hint 3

Can we use dynamic programming to make use of intermediate results?

Hide Hint 4

Think of a 3D array which can be used to store the longest line obtained so far for each direction.

Java | Autocomplete

```java
class Solution {
  public int longestLine(int[][] M) {
    if (M.length == 0) return 0;
    int ones = 0;
    int[][] dp = new int[M[0].length][4];
    for (int i = 0; i < M.length; i++) {
      int old = 0;
      for (int j = 0; j < M[0].length; j++) {
        if (M[i][j] == 1) {
          dp[j][0] = j > 0 ? dp[j - 1][0] + 1 : 1;
          dp[j][1] = i > 0 ? dp[j][1] + 1 : 1;
          int prev = dp[j][2];
          dp[j][2] = (i > 0 && j > 0) ? old + 1 : 1;
          old = prev;
          dp[j][3] = (i > 0 && j < M[0].length - 1) ? dp[j + 1][3] + 1 : 1;
          ones =
              Math.max(ones, Math.max(Math.max(dp[j][0], dp[j][1]), Math.max(dp[j][2], dp[j][3])));
        } else {
          old = dp[j][2];
          dp[j][0] = dp[j][1] = dp[j][2] = dp[j][3] = 0;
        }
      }
    }
    return ones;
  }
}
```