Given an m x n binary matrix filled with 0's and 1's, *find the largest square containing only* 1's *and return its area*.

**Example 1:**



**Input:** matrix = [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0","1","0"]]

**Output:** 4

**Example 2:**



**Input:** matrix = [["0","1"],["1","0"]]

**Output:** 1

**Example 3:**

**Input:** matrix = [["0"]]

**Output:** 0

**Solution:**

class Solution {

public int maximalSquare(char[][] matrix) {

int rows = matrix.length, cols = rows > 0 ? matrix[0].length : 0;

int[][] dp = new int[rows + 1][cols + 1];

int maxsqlen = 0;

for (int i = 1; i <= rows; i++) {

for (int j = 1; j <= cols; j++) {

if (matrix[i-1][j-1] == '1'){

dp[i][j] = Math.min(Math.min(dp[i][j - 1], dp[i - 1][j]), dp[i - 1][j - 1]) + 1;

maxsqlen = Math.max(maxsqlen, dp[i][j]);

}

}

}

return maxsqlen \* maxsqlen;

}

}

**Complexity Analysis**

* Time complexity : O(mn)*O*(*mn*). Single pass.
* Space complexity : O(mn)*O*(*mn*). Another matrix of same size is used for dp.