Largest square formed in a matrix

Given a binary matrix **mat** of size **n** * **m**, find out the maximum size square sub-matrix with all 1s.

Example 1:

```
Input: n = 2, m = 2
mat = {{1, 1}, 
       {1, 1}}
Output: 2
Explaination: The maximum size of the square
sub-matrix is 2. The matrix itself is the
maximum sized sub-matrix in this case.
```

Example 2:

```
Input: n = 2, m = 2
mat = {{0, 0}, 
       {0, 0}}
Output: 0
Explaination: There is no 1 in the matrix.
```

Your Task:
You do not need to read input or print anything. Your task is to complete the function **maxSquare()** which takes n, m and mat as input parameters and returns the size of the maximum square sub-matrix of given matrix.

Expected Time Complexity: O(nm)
Expected Auxiliary Space: O(nm)

Constraints:
1 ≤ n, m ≤ 50
0 ≤ mat[i][j] ≤ 1

```python
class Solution:
    def maxSquare(self, n, m, mat):
        # code here
        if n==1 or m==1:
            if n==1:
                return max(mat[0])
            if m==1:
                return max([mat[i][0] for i in range(len(mat))])
        dp = [[0]*m for _ in range(n)]
        ans = 0
        for i in range(n-1,-1,-1):
            for j in range(m-1,-1,-1):
                if i==n-1 and j==m-1:
                    dp[i][j] = mat[i][j]
                elif i==n-1:
                    dp[i][j] = mat[i][j]
                elif j==m-1:
                    dp[i][j] = mat[i][j]
                else:
                    if mat[i][j]==0:
                        dp[i][j] = 0
                    else:
                        dp[i][j] = min(dp[i+1][j],min(dp[i][j+1],dp[i+1]
[j+1]))+1
                        if dp[i][j]>ans:
                            ans = dp[i][j]
        # print(dp)
        return ans
```