221. Maximal Square

Description | Hints | Submissions | Discuss | Solution

Pick One

Given a 2D binary matrix filled with 0's and 1's, find the largest square containing only 1's and return its area.

Example:

```
Input:

1 0 1 0 0
1 0 1 1 1
1 1 1 1 1
1 0 0 1 0

Output: 4
```

我们知道当 maxtrix[i][j] = '1' 时，以 matrix[i][j] 为正方形右下角的边长，最多总是比以 matrix[i - 1][j]、matrix[i][j - 1]、matrix[i - 1][j - 1] 为右下角的正方形边长中最小的边长大1。这是因为，如果以 matrix[i - 1][j]、matrix[i][j - 1]、matrix[i - 1][j - 1] 为右下角的正方形边长相等，那么加上该点后就可以构成一个更大的正方形。如果它们不相等，那么因为缺失某部分，而无法构成更大正方形，那么只能取3个正方形中最小的一个加1，为此我们可以得到动态规划递推式。

可以进行检测：

```
For example, given the following matrix:

1 0 1 0 0
1 0 1 1 1
1 1 1 1 1
1 0 0 1 0

Return 4.
```

```java
public class L221 {
    public int maximalSquare(char[][] matrix) {
        if(matrix.length == 0)
            return 0;
        int [][] dp = new int [matrix.length][matrix[0].length];
        int max = 0;

        for(int i = 0; i < matrix.length; i ++) {
            dp[i][0] = matrix[i][0] - '0';
            max = Math.max(max, dp[i][0]);
        }

        for(int j = 0; j < matrix[0].length; j ++) {
            dp[0][j] = matrix[0][j] - '0';
            max = Math.max(max, dp[0][j]);
        }

        for(int i = 1; i < matrix.length; i ++) {
            for(int j = 1; j < matrix[0].length; j ++) {
                if(matrix[i][j] == 1) {
                    dp[i][j] = Math.min(dp[i-1][j-1], Math.min(dp[i][j-1], dp[i-1][j])) + 1;
                    max = Math.max(dp[i][j], max);
                }
            }
        }
        return max * max;
    }
}
```