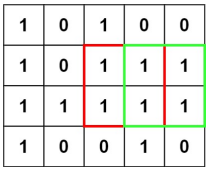
221 最大正方形

Label：动态规划
在一个由 '0' 和 '1' 组成的二维矩阵内，找到只包含 '1' 的最大正方形，并返回其面积。



- 暴力

```
class Solution {
    public int maximalSquare(char[][] matrix) {
        int maxSide = 0;
        if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {
            return maxSide;
        }
        int rows = matrix.length, columns = matrix[0].length;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                if (matrix[i][j] == '1') {
                    // 遇到一个 1 作为正方形的左上角
                    maxSide = Math.max(maxSide, 1);

                    // 计算可能的最大正方形边长
                    int currentMaxSide = Math.min(rows - i, columns - j);

                    for (int k = 1; k < currentMaxSide; k++) { // 递增拓展
                        // 判断新增的一行一列是否均为 1
                        boolean flag = true;
                        if (matrix[i + k][j + k] == '0') {
                            break;
                        }
                        for (int m = 0; m < k; m++) {  // 下侧和右侧
                            if (matrix[i + k][j + m] == '0' || matrix[i + m][j +
k] == '0') {
                                flag = false;
                                break;
                            }
                        }
                        if (flag)
                            maxSide = Math.max(maxSide, k + 1);
                        else
                            break;
                    }
                }
            }
        }
        return  maxSide * maxSide;
    }
}
```

- 动态规划 注意 拓展边

```
class Solution {
    public int maximalSquare(char[][] matrix) {
        if (matrix.length < 1) return 0;

        int[][] dict = new int[matrix.length + 1][matrix[0].length + 1];
        int max = 0;
        for (int i = 1; i <= matrix.length; i++) {
            for (int j = 1; j <= matrix[0].length; j++) {

                if(matrix[i-1][j-1] == '1') {
                    dict[i][j] = 1 + Math.min(dict[i - 1][j - 1],
Math.min(dict[i - 1][j], dict[i][j - 1]));
                    max = Math.max(max, dict[i][j]);
                }

            }
        }
        return max*max;
    }
}
```