221. Maximal Square

题目描述：https://leetcode.com/problems/maximal-square/

> 给定一个二维数组，该数组中的元素都是由0和1组成的，求这个数组中最大的都是由1组成的正方形。
> 例如：
>
> ```
> 1 0 1 0 0
> 1 0 1 1 1
> 1 1 1 1 1
> 1 0 0 1 0
>
> Return 4
> ```

解题思路：

动态规划。
使用f[i][j]代表以i,j结尾的最大的sqaure的边长。
则初始条件为

```
f[0][j] = matrix[0][j]
f[i][0] = matrix[i][0]
```

由于是以(i,j)结尾的正方形，所以这个正方形除了本身如果要包括其他节点，必须包括(i-1,j),(i,j-1),(i-1,j-1)三个节点。
又因为是正方形所以只能取这三个节点结尾的正方形中最小的那个。
则递推方程为

```
f[i][j] = min(f[i][j-1], f[i-1][j], f[i-1][j-1]) + 1
```

代码：

```
class Solution {
public:
    int maximalSquare(vector<vector<char>>& matrix) {
        int row = matrix.size();
        int res = 0;
        if(row == 0)
            return 0;
        int col = matrix[0].size();
        vector<vector<int> > f(row, vector<int> (col, 0));
        for(int i = 0; i < row; i++) {
            f[i][0] = matrix[i][0] - '0';
            res = max(res, f[i][0]);
        }
        for(int j = 0; j < col; j++) {
            f[0][j] = matrix[0][j] - '0';
            res = max(res, f[0][j]);
        }
        for(int i = 1; i < row; i++) {
            for(int j = 1; j < col; j++) {
                if(matrix[i][j] == '1') {
                    f[i][j] = min(f[i-1][j], min(f[i][j-1], f[i-1][j-1])) + 1;
                    res = max(res, f[i][j]);
                }
                else
                    f[i][j] = 0;
            }
        }
        return res*res;
    }
};
```