

矩阵中的最长递增路径

给定一个 $m \times n$ 整数矩阵 `matrix`，找出其中 **最长递增路径** 的长度。

对于每个单元格，你可以往上，下，左，右四个方向移动。你 **不能** 在 **对角线** 方向上移动或移动到 **边界外**（即不允许环绕）。

示例 1：

9	9	4
6	6	8
2	1	1

输入：matrix = [[9,9,4],[6,6,8],[2,1,1]]

输出：4

解释：最长递增路径为 [1, 2, 6, 9]。

示例 2：

3	4	5
3	2	6
2	2	1

输入：matrix = [[3,4,5],[3,2,6],[2,2,1]]

输出：4

解释：最长递增路径是 [3, 4, 5, 6]。注意不允许在对角线方向上移动。

示例 3:

输入: matrix = [[1]]

输出: 1

```
class Solution {
public:
    int dfs(vector<vector<int>>& memo, vector<vector<int>>& matrix, int i,
int j) {
        if (memo[i][j] != 0)
            return memo[i][j]; // 当记忆数组中值不为 0 时 作为递归结束返回值
        memo[i][j]++; // 初始化为 1, memo[i][j] 最小为 1, 意味着此时这个点上下左右
        // 都没法走, 它是上下左右最大的, 那么最长也就是 1 了
        // memo 表示从 (i, j) 这个点出发最长的递增路径长度。

        // 需要选四个方向中最长的路径长度将其记录进 memo[i][j]
        if (i - 1 >= 0 && matrix[i - 1][j] > matrix[i][j]) // 左 同时注意是否
        越界
            memo[i][j] = max(memo[i][j], dfs(memo, matrix, i - 1, j) + 1);

        if (i + 1 < matrix.size() && matrix[i + 1][j] > matrix[i][j]) //
        右 同时注意是否越界
            memo[i][j] = max(memo[i][j], dfs(memo, matrix, i + 1, j) + 1);
        if (j - 1 >= 0 && matrix[i][j - 1] > matrix[i][j]) // 上 同时注意是否
        越界
            memo[i][j] = max(memo[i][j], dfs(memo, matrix, i, j - 1) + 1);
        if (j + 1 < matrix[0].size() && matrix[i][j + 1] > matrix[i][j]) //
        下 同时注意是否越界
            memo[i][j] = max(memo[i][j], dfs(memo, matrix, i, j + 1) + 1);
        return memo[i][j];
    }
    int longestIncreasingPath(vector<vector<int>>& matrix) {
        if (matrix.size() == 0)
            return 0;
        int ans = 0;
        vector<vector<int>> memo(matrix.size(), vector<int>(matrix[0].size(
), 0)); // 将二维容器初始化全部为 0
        for (int i = 0; i < matrix.size(); i++)
            for (int j = 0; j < matrix[i].size(); j++)
                ans = max(ans, dfs(memo, matrix, i, j));
        return ans;
    }
}
```

```
};
```