剑指12 矩阵中的路径

我的成绩：差点错对。。

解析：深度优先搜索（DFS）+ 剪枝
很适合用DFS，递归先从一个方向搜到底，然后回溯到上个节点，从另外方向搜索。
可行性剪枝，指接下去的路不可能和目标匹配的情况应该立刻返回，如矩阵元素和目标不同、此元素已经访问过等。

重点：已访问过元素的信息处理——在更深递归前，修改当前元素为特定标记，深层递归跳出后，再把原值还给它。这样在这个分支的内部递归中，能记得这个元素是已经访问过的；而跳出当前分支时，又能恢复未访问状态。

```
// 使用hash来记录已访问过的节点会超时，可能还有其他问题。所以一直无法通过。
// 借鉴参考答案后修改如下：

class Solution {
public:
    int m;
    int n;
    int wordlen;

    bool findword(vector<vector<char>>& board, int x, int y, string word, int index) {
        if (index == wordlen) return true;
        if (x < 0 || y < 0 || x == m || y == n || board[x][y] != word[index]) return false;

        board[x][y] = '\0';

        bool res = findword(board, x - 1, y, word, index + 1) ||
            findword(board, x + 1, y, word, index + 1) ||
            findword(board, x, y - 1, word, index + 1) ||
            findword(board, x, y + 1, word, index + 1);

        board[x][y] = word[index];
        return res;
    }
    bool exist(vector<vector<char>>& board, string word) {
        m = board.size();
        n = board[0].size();
        wordlen = word.size();
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (findword(board, i, j, word, 0)) return true;
            }
        }
        return false;
    }
};
```

```
// k神：
```

```cpp
class Solution {
public:
    bool exist(vector<vector<char>>& board, string word) {
        rows = board.size();
        cols = board[0].size();
        for(int i = 0; i < rows; i++) {
            for(int j = 0; j < cols; j++) {
                if(dfs(board, word, i, j, 0)) return true;
            }
        }
        return false;
    }
private:
    int rows, cols;
    bool dfs(vector<vector<char>>& board, string word, int i, int j, int k) {
        if(i >= rows || i < 0 || j >= cols || j < 0 || board[i][j] != word[k]) return false;
        if(k == word.size() - 1) return true;
        board[i][j] = '\0';
        bool res = dfs(board, word, i + 1, j, k + 1) || dfs(board, word, i - 1, j, k + 1) ||
                   dfs(board, word, i, j + 1, k + 1) || dfs(board, word, i , j - 1, k + 1);
        board[i][j] = word[k];
        return res;
    }
};

```