剑指 Offer 12. 矩阵中的路径

```
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;
    }
};
```

作者：jyd
链接：https://leetcode-cn.com/problems/ju-zhen-zhong-de-lu-jing-lcof/solution/mian-shi-ti-12-ju-zhen-zhong-de-lu-jing-shen-du-yo/
来源：力扣（LeetCode）

改bug改到趴下；我无语了；（一个上下左右的坐标）（一个忘了加函数）

起始if循环相关的代码是可以优化的，相当于代码优化，诶!!!

```
class Solution {
private:
    int m, n;//行和列
    bool res;
    //vector<vector<bool>>path;//path数组//设置为引用以节省空间
    //回溯函数
    void backtracing(vector<vector<char>>& board, vector<vector<bool>>&
path, int x, int y, int tag, string& word) {
        //终止条件
        if (tag == word.size() - 1 && word[tag] == board[x][y]) {
            res = true;
            return;
        }
        //向四周发散
        if (board[x][y] != word[tag]) {
            return;
        }
        if (y + 1 < n && !path[x][y + 1]) {
            path[x][y] = true;
            backtracing(board, path, x, y + 1, tag + 1, word);
            path[x][y] = false;
        }
        if (y - 1 >= 0 && !path[x][y - 1]) {
            path[x][y] = true;
            backtracing(board, path, x, y - 1, tag + 1, word);
            path[x][y] = false;
        }
        if (x - 1 >= 0 && !path[x - 1][y]) {
            path[x][y] = true;
            backtracing(board, path, x - 1, y, tag + 1, word);
            path[x][y] = false;
        }
        if (x + 1 < m && !path[x + 1][y]) {
            path[x][y] = true;
            backtracing(board, path, x + 1, y, tag + 1, word);
            path[x][y] = false;
        }
    }
public:
    bool exist(vector<vector<char>>& board, string word) {
        //回溯算法
        //开始的点不确定，但是开始点的字符必须与首字符相同
        //设置一个path数组以防止同一个位置的字符重复的访问
        m = board.size(), n = board[0].size();
        res = false;
```

```
        vector<vector<bool>>path(m, vector<bool>(n, false));
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                backtracing(board, path, i, j, 0, word);
            }
        }
        return res;
    }
};
```