动态规划：44. 通配符匹配

```cpp
class Solution {
public:
    bool isMatch(string s, string p) {
        int m = s.size();
        int n = p.size();
        s = " " + s;
        p = " " + p;
        vector<vector<bool>> dp(m + 1, vector<bool>(n + 1));
        dp[0][0] = true;

        for (int j = 1; j <= n; j++) {
            if (p[j] == '*') {
                dp[0][j] = dp[0][j - 1]; // * 可以匹配空字符串
            }
        }

        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (p[j] == '?')
                    dp[i][j] = dp[i - 1][j - 1];
                else if (p[j] == '*') {
                     dp[i][j] = dp[i-1][j] || dp[i][j-1];
                } else {
                    if (s[i] == p[j] && dp[i - 1][j - 1]) {
                        dp[i][j] = true;
                    }
                }
            }
        }
        return dp[m][n];
    }
};
```

bfs： 130. 被围绕的区域

```cpp
class Solution {
    int dx[4] = {1, -1, 0, 0};
    int dy[4] = {0, 0, 1, -1};
    int m = 0;
    int n = 0;

public:
    void bfs(vector<vector<char>>& board, int i, int j) {
        queue<pair<int, int>> q;
        q.push({i, j});
        board[i][j] = '.';  // 标记已访问
        while (!q.empty()) {
            auto [a, b] = q.front();
            q.pop();

            for (int k = 0; k < 4; k++) {
                int x = a + dx[k];  // 更新应基于 a, b
                int y = b + dy[k];  // 更新应基于 a, b
                if (x >= 0 && x < m && y >= 0 && y < n && board[x][y] == 'O') {
                    q.push({x, y});
                    board[x][y] = '.';  // 标记已访问
                }
            }
        }
    }

    void solve(vector<vector<char>>& board) {
        if (board.empty()) return;  // 边界检查

        m = board.size();  // 行数
        n = board[0].size();  // 列数

        // 处理第一列和最后一列
        for (int i = 0; i < m; i++) {
            if (board[i][0] == 'O') {
                bfs(board, i, 0);
            }
            if (board[i][n - 1] == 'O') {
                bfs(board, i, n - 1);
```

```
            }
        }

        // 处理第一行和最后一行
        for (int j = 0; j < n; j++) {
            if (board[0][j] == 'O') {
                bfs(board, 0, j);
            }
            if (board[m - 1][j] == 'O') {
                bfs(board, m - 1, j);
            }
        }

        // 遍历整个 board，更新结果
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (board[i][j] == '.') {
                    board[i][j] = 'O';  // 从边界联通的 O
                } else if (board[i][j] == 'O') {
                    board[i][j] = 'X';  // 被包围的 O
                }
            }
        }
    }
};
```