# 宽度搜索1 ☆☆☆

## 题目描述

有一个 $n \times m$ 的棋盘，在某个点 $(x, y)$ 上有一个马，要求计算出马到达棋盘上任意一个点最少要走几步。

马每次走时，只能根据象棋的规则，走日字形状。

## 输入输出格式

**输入：**一行四个整数，分别为 $n, m, x, y$（$1 \le x \le n \le 400$，$1 \le y \le m \le 400$）。

**输出：**一个 $n \times m$ 的矩阵，代表马到达某个点最少要走几步（不能到达则输出 $-1$）。

| 输入示例 | 输出示例 |
| --- | --- |
| 3 3 1 1 | 0 3 2<br>3 -1 1<br>2 1 4 |

## 样例分析

以输入 `3 3 1 1` 为例：

| | | |
|---|---|---|
| 0 | 3 | 2 |
| 3 | -1 | 1 |
| 2 | 1 | 4 |

- **(1,1)** 是起点，距离为0
- **(1,2)** 需要3步：`(1,1)→(3,2)→(1,3)→(2,1)→(1,2)`
- **(1,3)** 需要2步：`(1,1)→(2,3)→(1,3)`
- **(2,1)** 需要3步：`(1,1)→(3,2)→(1,3)→(2,1)` 或 `(1,1)→(2,3)→(3,1)→(2,1)`
- **(2,2)** 无法到达
- **(2,3)** 需要1步：`(1,1)→(2,3)`
- **(3,1)** 需要2步：`(1,1)→(2,3)→(3,1)` 或 `(1,1)→(3,2)→(3,1)`
- **(3,2)** 需要1步：`(1,1)→(3,2)`
- **(3,3)** 需要4步：`(1,1)→(3,2)→(1,3)→(2,1)→(3,3)`

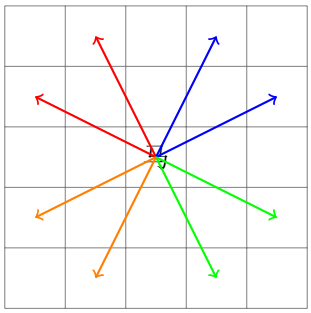
## 算法分析

1. 问题分析

   这是一个典型的最短路径问题，由于棋盘上每个位置的移动代价相同（每次移动一步），可以使用广度优先搜索（**BFS**）来解决。

2. **马走日的8个方向**

在国际象棋中，马走"日"字，有8个可能的移动方向：



用坐标表示这8个方向：

```
int dx[8] = {-2, -1, 1, 2, 2, 1, -1, -2};
int dy[8] = {1, 2, 2, 1, -1, -2, -2, -1};
```

3. **BFS算法步骤**

（1） 初始化距离数组，所有位置设为-1（表示未访问）

（2） 将起始位置加入队列，距离设为0

（3） BFS遍历：

- 从队列中取出当前位置
- 遍历8个方向，计算下一个位置
- 如果新位置在棋盘内且未被访问，更新距离并入队

## 代码实现

```
while (!q.empty()) {
    array<int, 2> TMP = q.front();
    int cx = TMP[0], cy = TMP[1];
    q.pop();
    for (int i = 0; i < 8; i++) {
        int nx = cx + dx[i], ny = cy + dy[i];
        if (nx < 0 || nx >= n || ny < 0 || ny >= m) continue;
        if (res[nx][ny] > res[cx][cy] + 1) {
            res[nx][ny] = res[cx][cy] + 1;
            q.push({nx, ny});
        }
    }
}
```

## 拓展思考

- 如果要求输出具体的行走路径而不仅仅是步数，该如何修改？
- 如果棋盘上有障碍物,且具有拌马腿的限制，该如何处理？