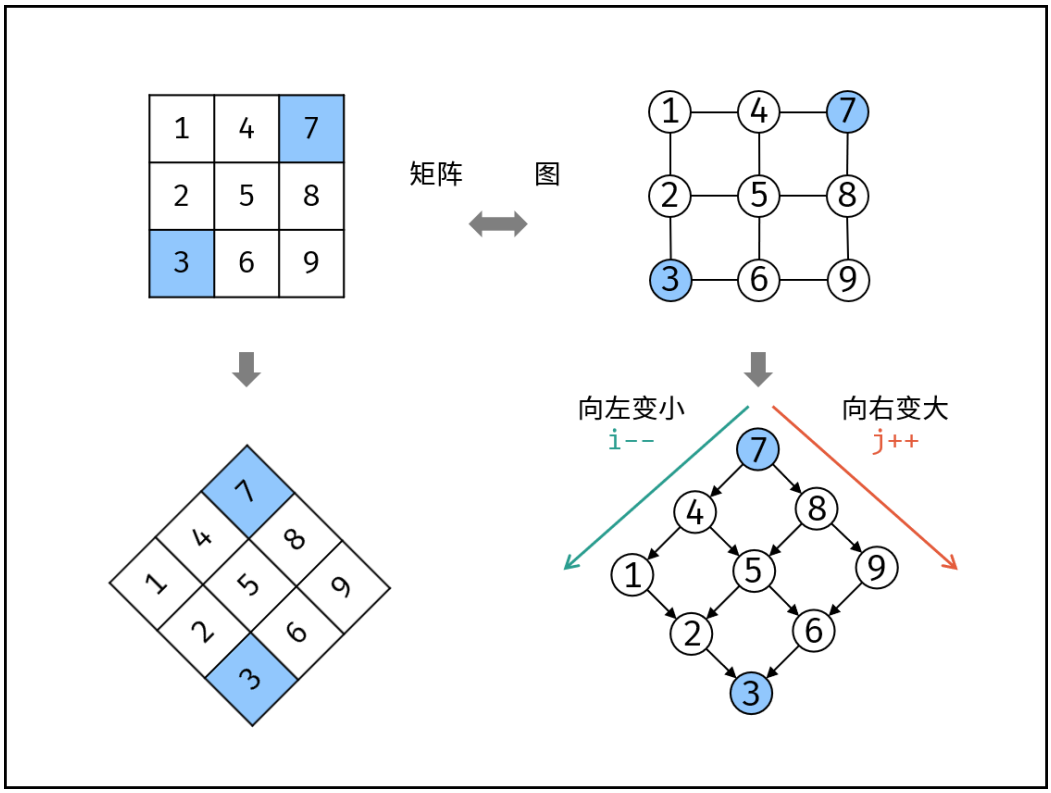
剑指04 二维数组中的查找

我的思路：广度优先。sb。和比直接遍历还垃圾。O(n*m)
其他思路：每一行2分搜索，是O(nlogn)也不错。
k神：时间O(n+m).右上角为起点，二叉搜索树的类似,只是子树之间有重叠的部分，但不影响逻辑正确性，如果小于当前节点，则寻找左子树，相当于排除最后一列。



感觉其实可以理解为左下角的数是该行的最小值，该列的最大值。所以如果target比左下角还小，则肯定比这一行都小，所以可以上移一行寻找；如果target比左下角还大，则肯定比这一列都大，所以可以右移一列寻找。

```cpp
// 垃圾
class Solution {
public:
    bool findNumberIn2DArray(vector<vector<int>>& matrix, int target) {
        if(matrix.size() == 0) return false;
        if(matrix[0].size() == 0) return false;
        queue<pair<int, int>> myq;
        myq.push(make_pair(0, 0));
        while(!myq.empty()) {
            pair<int, int> tmp = myq.front();
            myq.pop();
            if(matrix[tmp.first][tmp.second] == target) {
                return true;
            }
            else if(matrix[tmp.first][tmp.second] < target) {
                if(tmp.second + 1 < matrix[0].size())
                    myq.push(make_pair(tmp.first, tmp.second+1));
                if(tmp.second == 0 && tmp.first + 1 < matrix.size())
                    myq.push(make_pair(tmp.first + 1, tmp.second));
```

```
            }
        }
        return false;
    }
};
```

```
// 离谱的k神
class Solution {
public:
    bool findNumberIn2DArray(vector<vector<int>>& matrix, int target) {
        int i = matrix.size() - 1, j = 0;
        while(i >= 0 && j < matrix[0].size())
        {
            if(matrix[i][j] > target) i--;
            else if(matrix[i][j] < target) j++;
            else return true;
        }
        return false;
    }
};

```