- # 490. The Maze

Hint:

1. 经典bfs
2. 不能用set记录访问过的点，否则会超出内存限制

```
class Solution {
    public boolean hasPath(int[][] maze, int[] start, int[] destination) {
        Queue<int[]> queue = new LinkedList<>();
        //Set<int[]> set = new HashSet<>();
        boolean[][] visit = new boolean[maze.length][maze[0].length];
        queue.add(start);
        //set.add(start);
        visit[start[0]][start[1]] = true;
        while(!queue.isEmpty()){
            int size = queue.size();
            for(int i = 0; i < size; i++){
                int[] temp = queue.poll();
                if(temp[0] == destination[0] && temp[1] == destination[1]){
                    return true;
                }
                List<int[]> nextStepList = getNextStep(maze, temp[0], temp[1]);
                for(int[] step: nextStepList){
                    if(!visit[step[0]][step[1]]){
                        queue.add(step);
                        visit[step[0]][step[1]] = true;
                    }
                }
            }
        }
        return false;
    }

    private List<int[]> getNextStep(int[][] maze, int row, int col){
        int[][] dir = new int[][]{{0, 1}, {0, -1}, {1, 0}, {-1, 0}};
        List<int[]> nextStepList = new ArrayList<>();
        for(int i = 0; i < 4; i++){
            int newrow = row;
            int newcol = col;
            while(inBound(maze, newrow + dir[i][0], newcol + dir[i][1]) && maze[newrow + dir[i][0]][newcol + dir[i][1]] != 1){
                newrow = newrow + dir[i][0];
                newcol = newcol + dir[i][1];
            }
            nextStepList.add(new int[]{newrow, newcol});
        }
        return nextStepList;
    }

    private boolean inBound(int[][] maze, int row, int col){
        if(row < 0 || col < 0 || row >= maze.length || col >= maze[0].length){
            return false;
        }
        return true;
    }
}
```