链滴

Leetcode 每日抑题 (LCP 03. 机器人大冒险)

作者：ludengke95
原文链接：https://ld246.com/article/1601437305852
来源网站：链滴

LeetCode的每日抑题系列，LeetCode会每天随机一道题作为签到题.我也是菜鸡,如果今天没A掉,那只证明我离大神又进了一步.

PS: 最近几天的每日抑题都好水,我就不写了,最近的每日抑题是我自己找的中等难度的题,因为比赛的候中等难度的题都有好多做不出来,困难题就更别说了.

题目链接

题名	通过率	难度
LCP 03. 机器人大冒险	20.4%	中等

力扣团队买了一个可编程机器人，机器人初始位置在原点(0, 0)。小伙伴事先给机器人输入一串指令command，机器人就会无限循环这条指令的步骤进行移动。指令有两种：

U: 向y轴正方向移动一格

R: 向x轴正方向移动一格。

不幸的是，在 xy 平面上还有一些障碍物，他们的坐标用obstacles表示。机器人一旦碰到障碍物就会损毁。

给定终点坐标(x, y)，返回机器人能否完好地到达终点。如果能，返回true；否则返回false。

示例 1：

输入：command = "URR", obstacles = [], x = 3, y = 2

输出：true

解释：U(0, 1) -> R(1, 1) -> R(2, 1) -> U(2, 2) -> R(3, 2)。

示例 2：

输入：command = "URR", obstacles = [[2, 2]], x = 3, y = 2
输出：false
解释：机器人在到达终点前会碰到(2, 2)的障碍物。
示例 3：

输入：command = "URR", obstacles = [[4, 2]], x = 3, y = 2
输出：true
解释：到达终点后，再碰到障碍物也不影响返回结果。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/programmable-robot

解题思路

其实蛮简单的题,就是因为数量级的问题导致模拟的方式会出现超时,但是可以简化.

1. **计算出点可能在第几次循环**
2. **判断这几次的循环是否覆盖障碍点.**

java代码

```
/**
 * 描述:
 *
 * @author ludengke
 * @create 2020-09-29 11:42
 */
public class LeetCodeLCP03 {
    public boolean robot(String command, int[][] obstacles, int x, int y) {
        /**
         * 用于记录第一个循环的走过的x,y坐标, 以及走过的横向和纵向偏移量
         */
        int[] pointX = new int[command.length()];
        int[] pointY = new int[command.length()];
        int startX = 0, startY = 0, xMax = 0, yMax = 0;
        for (int i = 0; i < command.length(); i++) {
            if (command.charAt(i)=='U'){
                startY += 1;
            }else if(command.charAt(i)=='R'){
                startX += 1;
            }
            pointX[i] = startX;
            pointY[i] = startY;
            xMax = pointX[i] > xMax?pointX[i]:xMax;
            yMax = pointY[i] > yMax?pointY[i]:yMax;
        }
```

```
        for (int i = 0; i < obstacles.length; i++) {
            if(obstacles[i][0] <= x && obstacles[i][1] <= y){
                /**
                 * 计算出到达obstacles[i]可能需要几个循环,保险起见上下各加一个
                 * 双层循环判断某点是否在某个循坏上.
                 */
                int num = obstacles[i][0] / xMax;
                for(int k = num-1;k <= num+1;k++){
                    for (int j = 0; j < command.length(); j++) {
                        if(startX * k + pointX[j] == obstacles[i][0] && startY * k + pointY[j] == obstacle
[i][1]){
                            return false;
                        }
                    }
                }
            }
        }

        /**
         * 计算出到达终点可能需要几个循环,保险起见上下各加一个
         * 双层循环判断终点是否在某个循坏上, 判断机器人是否能够到达终点
         */
        int num = x / xMax;
        for(int k = num-1;k <= num+1;k++){
            for (int j = 0; j < command.length(); j++) {
                if(startX * k + pointX[j] == x && startY * k + pointY[j] == y){
                    return true;
                }
            }
        }
        return false;
    }

    public static void main(String[] args) {
        LeetCodeLCP03 l = new LeetCodeLCP03();
//        System.out.println(l.robot());
    }
}
```

算法or数据结构

1. 模拟