LeetCode_16.22兰顿蚂蚁

原创 xyzso1z 2020-09-22 15:08:59 5 收藏 编辑 版权

分类专栏： # LeetCode 文章标签： leetcode 兰顿蚂蚁

16.22. 兰顿蚂蚁

一只蚂蚁坐在由白色和黑色方格构成的无限网格上。开始时，网格全白，蚂蚁面向右侧。每行走一步，蚂蚁执行以下操作。

1. 如果在白色方格上，则翻转方格的颜色，向右(顺时针)转 90 度，并向前移动一个单位。
2. 如果在黑色方格上，则翻转方格的颜色，向左(逆时针方向)转 90 度，并向前移动一个单位。

编写程序来模拟蚂蚁执行的前 K 个动作，并返回最终的网格。

网格由数组表示，每个元素是一个字符串，代表网格中的一行，黑色方格由 'X' 表示，白色方格由 '_' 表示，蚂蚁所在的位置由 'L', 'U', 'R', 'D' 表示，分别表示蚂蚁 左、上、右、下 的朝向。只需要返回能够包含蚂蚁走过的所有方格的最小矩形，如果当前位置是蚂蚁所在位置，则用蚂蚁方向代替方格颜色。

示例 1：

> **输入**：0
> **输出**：["R"]

示例 2：

> 输入：2
> 输出：["_X", "LX"]

示例 3：

> 输入：5
> 输出：[
> "_U",
> "X_",
> "XX"
>]

提示：

- K <= 100000

答案一

```
class Solution {

    // 白色方格
        private final int WHITE = 0;
        // 黑色方格
        private final int BLACK = 1;

        private static final int ANT_LEFT = 'L';
        private static final int ANT_TOP = 'U';
        private static final int ANT_RIGHT = 'R';
        private static final int ANT_BOTTOM = 'D';

        int size = 10000;


        // 当前蚂蚁处于表格中的下标
        int currentX = size / 2;
        int currentY = size / 2;

        // 当前蚂蚁头方向
        int currentDir = ANT_RIGHT;

        // 蚂蚁走过的表格下标最值
        int minX = currentX, maxX = currentX, minY = currentY, maxY = currentY;

    //保存黑色网格
        Set<String> blackSet = new HashSet<String>();

        private String getKey(int x, int y) {
                return String.valueOf(x) + y;
        }

        public List<String> printKMoves(int K) {
                for (int i = 0; i < K; i++) {
                        next();
                }
                List<String> list = new ArrayList<String>();
                for (int j = minY; j <= maxY; j++) {
                        StringBuilder sBuilder = new StringBuilder();
```

```
                for (int i = minX; i <= maxX; i++) {
                        if (i == currentX && j == currentY) {
                                sBuilder.append((char) currentDir);
                        } else {

                                sBuilder.append(blackSet.contains(getKey(
                        }
                }
                list.add(sBuilder.toString());
        }
        return list;
}

private void next() {
        int lastX = currentX;
        int lastY = currentY;
        if (!blackSet.contains(getKey(currentX, currentY))) {
                switch (currentDir) {
                case ANT_LEFT:
                        currentY = currentY - 1;
                        currentDir = ANT_TOP;
                        break;
                case ANT_TOP:
                        currentX = currentX + 1;
                        currentDir = ANT_RIGHT;
                        break;
                case ANT_RIGHT:
                        currentY = currentY + 1;
                        currentDir = ANT_BOTTOM;
                        break;
                case ANT_BOTTOM:
                        currentX = currentX - 1;
                        currentDir = ANT_LEFT;
                        break;
                default:
                }
        } else {
                switch (currentDir) {
                case ANT_LEFT:
                        currentY = currentY + 1;
                        currentDir = ANT_BOTTOM;
                        break;
                case ANT_TOP:
                        currentX = currentX - 1;
                        currentDir = ANT_LEFT;
                        break;
                case ANT_RIGHT:
```

```
                        currentY = currentY - 1;
                        currentDir = ANT_TOP;
                        break;
                case ANT_BOTTOM:
                        currentX = currentX + 1;
                        currentDir = ANT_RIGHT;
                        break;
                default:
                }
        }

        if (!blackSet.contains(getKey(lastX, lastY))) {
                blackSet.add(getKey(lastX, lastY));
        } else {
                blackSet.remove(getKey(lastX, lastY));
        }

        if (minX > currentX) {
                minX = currentX;
        }
        if (maxX < currentX) {
                maxX = currentX;
        }
        if (minY > currentY) {
                minY = currentY;
        }
        if (maxY < currentY) {
                maxY = currentY;
        }
    }

}
```

资料

1. LeetCode原题
2. Android客户端演示Demo
3. 演示效果