

# 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:

- $K \leq 100000$

## 答案一

```
1  class Solution {
2
3      // 白色方格
4      private final int WHITE = 0;
5      // 黑色方格
6      private final int BLACK = 1;
7
8      private static final int ANT_LEFT = 'L';
9      private static final int ANT_TOP = 'U';
10     private static final int ANT_RIGHT = 'R';
11     private static final int ANT_BOTTOM = 'D';
12
13     int size = 10000;
14
15
16     // 当前蚂蚁处于表格中的下标
17     int currentX = size / 2;
18     int currentY = size / 2;
19
20     // 当前蚂蚁头方向
21     int currentDir = ANT_RIGHT;
22
23     // 蚂蚁走过的表格下标最值
24     int minX = currentX, maxX = currentX, minY = currentY, maxY = currentY;
25
26     //保存黑色网格
27     Set<String> blackSet = new HashSet<String>();
28
29     private String getKey(int x, int y) {
30         return String.valueOf(x) + y;
31     }
32
33     public List<String> printKMoves(int K) {
34         for (int i = 0; i < K; i++) {
35             next();
36         }
37         List<String> list = new ArrayList<String>();
38         for (int j = minY; j <= maxY; j++) {
39             StringBuilder sBuilder = new StringBuilder();
40
```

```
41     for (int i = minX; i <= maxX; i++) {
42         if (i == currentX && j == currentY) {
43             sBuilder.append((char) currentDir);
44         } else {
45
46             sBuilder.append(blackSet.contains(getKey(
47                 i, j, currentDir)
48             ));
49             list.add(sBuilder.toString());
50         }
51     }
52     return list;
53 }
54 private void next() {
55     int lastX = currentX;
56     int lastY = currentY;
57     if (!blackSet.contains(getKey(currentX, currentY))) {
58         switch (currentDir) {
59             case ANT_LEFT:
60                 currentY = currentY - 1;
61                 currentDir = ANT_TOP;
62                 break;
63             case ANT_TOP:
64                 currentX = currentX + 1;
65                 currentDir = ANT_RIGHT;
66                 break;
67             case ANT_RIGHT:
68                 currentY = currentY + 1;
69                 currentDir = ANT_BOTTOM;
70                 break;
71             case ANT_BOTTOM:
72                 currentX = currentX - 1;
73                 currentDir = ANT_LEFT;
74                 break;
75             default:
76                 break;
77         }
78     } else {
79         switch (currentDir) {
80             case ANT_LEFT:
81                 currentY = currentY + 1;
82                 currentDir = ANT_BOTTOM;
83                 break;
84             case ANT_TOP:
85                 currentX = currentX - 1;
86                 currentDir = ANT_LEFT;
87                 break;
88             case ANT_RIGHT:
```

```
88         currentY = currentY - 1;
89         currentDir = ANT_TOP;
90         break;
91     case ANT_BOTTOM:
92         currentX = currentX + 1;
93         currentDir = ANT_RIGHT;
94         break;
95     default:
96     }
97 }
98
99 if (!blackSet.contains(getKey(lastX, lastY))) {
100     blackSet.add(getKey(lastX, lastY));
101 } else {
102     blackSet.remove(getKey(lastX, lastY));
103 }
104
105 if (minX > currentX) {
106     minX = currentX;
107 }
108 if (maxX < currentX) {
109     maxX = currentX;
110 }
111 if (minY > currentY) {
112     minY = currentY;
113 }
114 if (maxY < currentY) {
115     maxY = currentY;
116 }
117 }
118 }
```

## 资料

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