【华为OD机考 统一考试机试C卷】反射计数（C++ Java JavaScript Python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

2023年11月份，华为官方已经将 华为OD机考：OD统一考试（A卷 / B卷）切换到 OD统一考试（C卷）和 OD统一考试（D卷） 。根据考友反馈：目前抽到的试卷为B卷或C卷/D卷，其中C卷居多 ，按照之前的经验C卷D卷部分考题会复用A卷/B卷题，博主正积极从考过的同学收集C卷和D卷真题，可以查看下面的真题目录。

真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境 华为OD机考B卷C卷华为OD机考华为OD机考B卷华为OD机试B卷华为OD机试C卷华为OD机考C卷华为OD机考D卷题目华为OD机考C卷/D卷答案华为OD机考C卷/D卷解析华为OD机考C卷和D卷真题华为OD机考C卷和D卷题解

题目描述

给定一个包含 0 和 1 的二维矩阵，给定一个初始位置和速度，一个物体从给定的初始位置触发, 在给定的速度下进行移动, 遇到矩阵的边缘则发生镜面反射。

无论物体经过 0 还是 1, 都不影响其速度

请计算并给出经过 t 时间单位后, 物体经过 1 点的次数

矩阵以左上角位置为[0, 0](列(x), 行(行)), 例如下面A点坐标为[2, 1] (第二列, 第一行)

```
+-------------------------- 递增(x)
| 0 0 1 0 0 0 0 1 0 0 0 0
| 0 0 1 0 0 0 0 1 0 0 0 0
| 0 0 1 0 0 0 0 1 0 0 0 0
| 0 0 1 0 0 0 0 1 0 0 0 0
| 0 0 1 0 0 0 0 1 0 0 0 0
| 0 0 1 0 0 0 0 1 0 0 0 0
| 0 0 1 0 0 0 0 1 0 0 0 0
|
递增(y)
```

注意:

- 如果初始位置的点是 1, 也计算在内
- 时间的最小单位为1, 不考虑小于 1 个时间单位内经过的点

输入描述

第一行为初始信息

```
```

第二行开始一共 h 行，为二维矩阵信息

其中：

- w，h 为矩阵的宽和高
- x，y 为起始位置
- sx，sy 为初始速度
- t 为经过的时间

所有输入都是有效的，数据范围如下：

- 0 < w < 100
- 0 < h < 100
- 0 ≤ x < w
- 0 ≤ y < h
- -1 ≤ sx ≤ 1
- -1 ≤ sy ≤ 1
- 0 ≤ t < 100

输出描述

经过 1 的个数

注意初始位置也要计算在内

用例

输入：

```
12 7 2 1 1 -1 13
001000010000
001000010000
001000010000
001000010000
001000010000
001000010000
001000010000
```

输出：

```
3
```

说明：

> 初始位置为(2, 1), 速度为(1, -1), 那么13个时间单位后, 经过点1的个数为3

解题思路

这段Java代码的主要目标是模拟一个物体在二维矩阵中的运动，并计算在给定的时间单位内，物体经过1的次数。

解题思路如下：

1. **模拟运动**：首先初始化一个计数器 `count` ，并将物体的初始位置的值（即 `matrix[y][x]` ）加到计数器上。然后，进入一个循环，每次循环模拟一个时间单位的运动。
    - **移动物体**：在每个时间单位内，根据物体的速度 `(sx, sy)` 移动物体，即更新物体的位置 `(x, y)` 。
    - **处理反射**：如果物体移动后到达了二维矩阵的边界（即 `x` 等于0或 `w-1` ，或者 `y` 等于0或 `h-1` ），则改变物体的方向，即反转速度的相应分量。
    - **更新计数器**：如果物体移动后的位置是1，即 `matrix[y][x]` 等于1，则将该值加到计数器上。

C++

```cpp
#include <iostream>
#include <vector>

using namespace std;

// 计算物体在移动过程中经过的1的数量的函数
int countOnes(int w, int h, int x, int y, int sx, int sy, int t, vector<vector<int>>& matrix) {
    int count = matrix[y][x];
    while (t-- > 0) {
        x += sx;
        y += sy;

        // 如果物体撞到左右边界，反转x轴方向速度分量
        if (x == 0 || x == w - 1) sx = -sx;
        // 如果物体撞到上下边界，反转y轴方向速度分量
        if (y == 0 || y == h - 1) sy = -sy;

        // 每次移动后，如果新位置是1，则累加到计数器
        count += matrix[y][x];
    }
    return count;
}

int main() {
    int w, h, x, y, sx, sy, t;
    cin >> w >> h >> x >> y >> sx >> sy >> t;
    vector<vector<int>> matrix(h, vector<int>(w));

    for (int i = 0; i < h; ++i) {
        for (int j = 0; j < w; ++j) {
            char ch;
            cin >> ch;
            matrix[i][j] = ch - '0';
        }
    }

    cout << countOnes(w, h, x, y, sx, sy, t, matrix) << endl;


```

```
    return 0;
}
```

Java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // 创建Scanner对象用于接收控制台输入
        Scanner scanner = new Scanner(System.in);
        // 读取矩阵的宽度w、高度h、物体初始位置x、y、物体移动速度分量sx、sy以及总时间t
        int w = scanner.nextInt(); // 矩阵宽度
        int h = scanner.nextInt(); // 矩阵高度
        int x = scanner.nextInt(); // 物体初始x坐标
        int y = scanner.nextInt(); // 物体初始y坐标
        int sx = scanner.nextInt(); // 物体在x轴方向的速度分量
        int sy = scanner.nextInt(); // 物体在y轴方向的速度分量
        int t = scanner.nextInt(); // 总移动时间
        scanner.nextLine(); // 读取并丢弃当前行剩余的所有数据，包括换行符

        // 根据输入的高度和宽度初始化矩阵
        int[][] matrix = new int[h][w];
        for (int i = 0; i < h; i++) {
            // 读取矩阵的每一行，每行是一个由'0'和'1'组成的字符串
            String line = scanner.nextLine();
            for (int j = 0; j < w; j++) {
                // 将每个字符转换为整数0或1，并存入矩阵对应位置
                matrix[i][j] = line.charAt(j) - '0';
            }
        }
        // 关闭scanner对象
        scanner.close();

        // 调用countOnes方法计算并输出物体在给定时间内经过的1的数量
        System.out.println(countOnes(w, h, x, y, sx, sy, t, matrix));
    }

    // countOnes方法用于计算物体在移动过程中经过的1的数量
    public static int countOnes(int w, int h, int x, int y, int sx, int sy, int t, int[][] matrix) {
```

```
        // 从物体的初始位置开始计数，如果初始位置是1，则计数器初始值为1
        int count = matrix[y][x];
        // 循环t次，每次代表物体移动一个时间单位
        while (t-- > 0) {
            // 根据速度分量更新物体位置
            x += sx;
            y += sy;
            // 如果物体撞到左右边界，反转x轴方向速度分量
            if (x == 0 || x == w - 1) sx = -sx;
            // 如果物体撞到上下边界，反转y轴方向速度分量
            if (y == 0 || y == h - 1) sy = -sy;
            // 每次移动后，如果新位置是1，则累加到计数器
            count += matrix[y][x];
        }
        // 返回计数器的值，即物体经过的1的总数
        return count;
    }
}
```

javaScript

```
const readline = require('readline');
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

const lines = [];
rl.on('line', (line) => {
    lines.push(line);
}).on('close', () => {
    // 读取输入的矩阵宽度w、高度h、物体初始位置x、y、物体移动速度分量sx、sy以及总时间t
    const [w, h, x, y, sx, sy, t] = lines[0].split(' ').map(Number);

    // 根据输入的高度和宽度初始化矩阵
    const matrix = lines.slice(1, h + 1).map(line => line.split('').map(Number));

    // 计算并输出物体在给定时间内经过的1的数量
    console.log(countOnes(w, h, x, y, sx, sy, t, matrix));
});
```

```
20
21
22  // countOnes函数用于计算物体在移动过程中经过的1的数量
23  function countOnes(w, h, x, y, sx, sy, t, matrix) {
24      let count = matrix[y][x];
25      while (t-- > 0) {
26          x += sx;
27          y += sy;
28
29          // 如果物体撞到左右边界，反转x轴方向速度分量
30          if (x == 0 || x == w - 1) sx = -sx;
31          // 如果物体撞到上下边界，反转y轴方向速度分量
32          if (y == 0 || y == h - 1) sy = -sy;
33
34          // 每次移动后，如果新位置是1，则累加到计数器
35          count += matrix[y][x];
36      }
37      return count;
}
```

Python

```
1   # 导入sys模块用于从控制台读取输入
2   import sys
3
4   # countOnes函数用于计算物体在移动过程中经过的1的数量
5   def count_ones(w, h, x, y, sx, sy, t, matrix):
6       count = matrix[y][x]
7       while t > 0:
8           t -= 1
9           x += sx
10          y += sy
11
12          # 如果物体撞到左右边界，反转x轴方向速度分量
13          if x == 0 or x == w - 1:
14              sx = -sx
15          # 如果物体撞到上下边界，反转y轴方向速度分量
16          if y == 0 or y == h - 1:
17              sy = -sy
18
19          # 每次移动后，如果新位置是1，则累加到计数器
```

```
        count += matrix[y][x]

    return count

def main():
    # 从控制台读取矩阵的宽度w、高度h、物体初始位置x、y、物体移动速度分量sx、sy以及总时间t
    w, h, x, y, sx, sy, t = map(int, input().split())

    # 根据输入的高度和宽度初始化矩阵
    matrix = []
    for _ in range(h):
        line = input()
        # 将每个字符转换为整数0或1，并存入矩阵对应位置
        matrix.append([int(ch) for ch in line])

    # 调用count_ones函数计算并输出物体在给定时间内经过的1的数量
    print(count_ones(w, h, x, y, sx, sy, t, matrix))

if __name__ == "__main__":
    main()
```

C语言

```
#include <stdio.h>
#include <stdlib.h>

// 计算物体在移动过程中经过的1的数量的函数
int countOnes(int w, int h, int x, int y, int sx, int sy, int t, int **matrix) {
    int count = matrix[y][x];
    while (t-- > 0) {
        x += sx;
        y += sy;

        // 如果物体撞到左右边界，反转x轴方向速度分量
        if (x == 0 || x == w - 1) sx = -sx;
        // 如果物体撞到上下边界，反转y轴方向速度分量
        if (y == 0 || y == h - 1) sy = -sy;

        // 每次移动后，如果新位置是1，则累加到计数器
        count += matrix[y][x];
```

```
    }
    return count;
}

int main() {
    int w, h, x, y, sx, sy, t;
    scanf("%d %d %d %d %d %d %d", &w, &h, &x, &y, &sx, &sy, &t);

    // 使用动态内存分配创建二维数组
    int **matrix = (int **)malloc(h * sizeof(int *));
    for (int i = 0; i < h; i++) {
        matrix[i] = (int *)malloc(w * sizeof(int));
    }

    // 读取矩阵数据
    for (int i = 0; i < h; i++) {
        for (int j = 0; j < w; j++) {
            char ch;
            scanf(" %c", &ch); // 注意%c前的空格，用于跳过任何空白字符
            matrix[i][j] = ch - '0';
        }
    }

    // 计算并输出物体在给定时间内经过的1的数量
    printf("%d\n", countOnes(w, h, x, y, sx, sy, t, matrix));

    // 释放动态分配的内存
    for (int i = 0; i < h; i++) {
        free(matrix[i]);
    }
    free(matrix);

    return 0;
}
```

文章目录