

Problem set3 report

17340027 姚洁倩

一、 题目描述

在此次作业中，题目要求我们设计和实现一个程序模拟机器人的运动，使用类来实现。

需要实现 Rectangular Room、Robot 及其子类。并且运行模拟的过程。通过运行模拟，需要对不同情况下的机器人清理房间的时间进行分析，并回答给出的问题。最后，将模拟的过程可视化。

二、 解题思路

1. 实现 RectangularRoom 和 Robot 类

➤ RectangularRoom

- 成员变量：width, height 表示宽度和高度，dirt_amount 表示每个地板块初始时的尘土的数目，tiles 是一个二维数组，用于记录这个房间里面的地板块的尘土的情况，如 tiles[4][2]表示的是在坐标轴上左下角坐标为（4，2）的地板块。初始的时候将每个地板块都设置为 dirt_amount
- 成员函数：
 - ✧ Clean_tile_at_position: 给出清理容量和位置，首先需要将位置进行处理，使用 math.floor 函数，向下取整，定位地板块，接着检查 tiles 这个地板块的尘土数量是否比清理容量大，若大于，则减去此数量的尘土，若小于等于，就置其为 0，这样就避免了负数出现的情况。
 - ✧ Is_tile_cleaned: 给出坐标的位置，判断地板块是否被清理干净了。检查 tiles 这个位置的地板块的尘土数目，若为 0 返回真，否则返回假。
 - ✧ Get_num_cleaned_tiles: 统计已经被清理好的地板块数量。遍历存储地板块尘土情况的二维数组，看看有多少被清理干净了。返回清理干净的地板块数目。
 - ✧ Is_position_in_room: 判断某位置是否在房间里。首先 x, y 必须大于 0，其次，x 大小不能超过房间的宽度，y 大小不能超过房间的高度。若不符合，返回假，都符合则返回真。
 - ✧ Get_dirt_amount: 获取某块地板上尘土的数目，直接返回 tiles[m][n]即可
 - ✧ Get_num_tiles、Is_position_valid、Get_random_position 这三个需要在子类中实现。

➤ Robot

- 成员变量:room 是 RectangularRoom 的一个实例,表示 robot 所在的房间，speed 是 robot 运动的速度，capacity 是它的单次清理容量，direction 是它的方向，pos 是一个 position 实例，标示其位置。
- 成员函数
 - ✧ Get_robot_position: 获取机器人的位置，直接返回 pos 即可
 - ✧ Get_robot_direction: 获取机器人的方向，直接返回 direction 即可
 - ✧ Set_robot_position: 设置机器人的位置，直接赋值即可

- ◇ Set_robot_direction: 设置机器人的方向, 直接赋值即可。
- ◇ Update_position_and_clean: 这个函数在子类中实现。

2. 实现 EmptyRoom 和 FurnishedRoom 类

➤ EmptyRoom

- 继承自 RectangularRoom, 没有额外的成员变量
- 成员函数:
 - ◇ Get_num_tiles: 对于一个空的房间, 地板块的数目等于房间的宽度乘以房间的高度
 - ◇ Is_position_valid: 对于一个空房间, 只要位置在这个房间内就是合法的
 - ◇ Get_random_position: 获取一个在空房间中的随机位置, 使用 random 模块产生随机数 x, y 表示横纵坐标, 其中, x 的范围是[0,width-1], y 的范围是[0,height-1]

➤ FurnishedRoom

- 成员变量: 多一个 furniture_tile 的列表, 存储放置了家具的地板块的坐标, 其中每个坐标以二元组表示。
- 成员函数
 - ◇ Add_furniture_to_room: 这是帮助函数, 添加家具进入这个房间
 - ◇ Is_tile_furnished: 判断此块地板是否有家具, 只要判断二元组(m,n)是否在 furniture_tiles 中即可。
 - ◇ Get_num_tiles: 对于带有家具装修的房间, 能够被访问的地板块是总的宽乘以高减去有家具的地板块的数量, 而有家具的地板块的数量通过 furniture_tile 的长度可以获知
 - ◇ Is_position_valid: 对于有家具的房间, 首先判断此位置是否超出了房间的范围, 若超出, 返回假。其次, 判断这个地方有没有放置家具, 若放置了家具, 则返回假。只有当位置在这个房间内且没有家具放置在上面的时候才返回真。
 - ◇ Get_random_position: 获取一个随机的位置。首先要明确这个位置上不能有家具, 使用随机数获取 x, y 然后看(x,y)上是否有家具, 若有, 重新计算 x, y, 直到(x,y)上没有家具为止。

3. 实现 StandardRobot 和模拟单步时间

StandardRobot 继承自 Robot 类, 需要实现的函数为 update_and_clean, 在这个函数中, 需要模拟机器人在一个单位时间内所做的事情: 首先计算出它要去的位置, 接下来, 判断位置是否合法, 若合法则去那个地方并清理那个地方; 否则, 调整自己的方向, 调整方向的时候不进行移动, 也不对现在所处的块进行清理。

4. 实现 FaultyRobot

FaultyRobot 继承自 Robot 类, 多了一个成员变量 p, 表示单个时间单位内出错的概率。Get_faulty 函数判断 robot 是否出了问题。在 update_position_and_clean 函数中, 首先需要判断此机器人是否出错, 若出错, 则仅改变自己的方向, 除此之外什么也不做; 否则, 像正常机器人一样计算下一步, 看看下一步是否合法位置, 若合法, 则移动并清理, 若不合法则调整位置。

5. 创建模拟器

在此处，需要模拟机器人清理房间的到一个给定的百分比的过程，并且输出这种情形下平均清理所需的时间。

实现 `run_simulation` 函数：函数的参数有机器人的总数、机器人的速度、机器人的清理容量，房间的宽和高、每块地板尘土初始数目，最小覆盖率（需要清干净的比例）、试验次数、机器人的类型。首先，设置一个变量 `total_time`，用于记录几次试验所用的总时间，接着使用 `for` 循环进行试验。在每次试验开始的时候，重新创建一个空房间，使用给定的宽度和高度，并且创建一个列表用于存储机器人的实例，使用 `for` 循环创建机器人并且将它们加进列表中。在它们开始清理之前将已经清理好的比率设置为 0，将时间设置为 0。使用 `while` 循环开始清理的过程，循环终止的条件是已经清理好的比率大于等于所要求清理的比率。循环体内，首先将时间加一，接着遍历机器人列表，每个机器人都执行清理的操作，使用 `update_and_clean` 函数。当循环结束时，在 `total_time` 中加入这次执行的时间。最后，当所有试验完成，返回 `total_time/试验次数`，即平均时间。

6. 运行模拟器

运行 `show plot compare strategies` 和 `show plot room shape` 函数，可以得到两张图表。对这两张图表进行分析，可以得到这两个问题的答案

ANSWER THE FOLLOWING QUESTIONS:

1)How does the performance of the two robot types compare when cleaning 80% of a 20x20 room?

从图中曲线可见，蓝色曲线总在黄色曲线的下方，可见，对于同样大小、同样清理比率的房间，有着同样清理容量和运行速度的机器人，Standard Robot 所需要的时间比 Faulty Robot 要小，因此 Standard Robot 的性能比 Faulty Robot 要好。对于两种机器人来说，机器人总数的增加，能够显著地减少清理房间的时间。

2) How does the performance of the two robot types compare when two of each robot cleans 80% of rooms with dimensions 10x30, 20x15, 25x12, and 50x6?

在第二张图中，不论是什么形状的房间，Standard Robot 清理的时间都比 Faulty Robot 使用的要少得多。对于两种机器人来说，相同面积下，纵横比越高的房间，清理所需要的时间也越长。而对于 Faulty Robot 来说，这个清理所需时间的增长要比 Standard Robot 要稍快一些。

7. 可视化机器人模拟

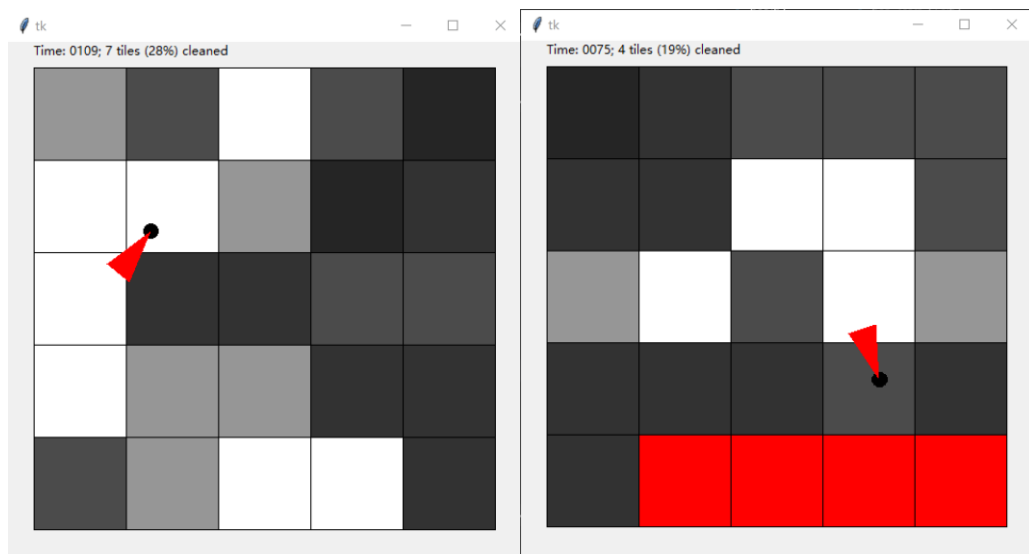
按照所给的提示，在 `run_simulation` 函数中加入相对应的三条语句，即可成功地进行可视化的模拟

三、 运行结果

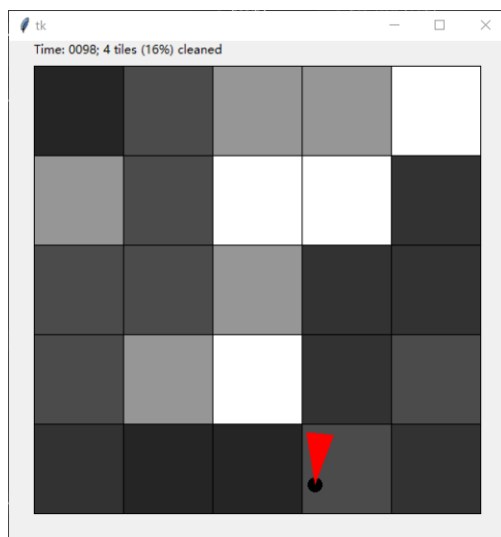
a) Standard Robot 在 Empty Room 和 Furnished Room 的测试

在测试中，机器人表现正常，并没有越出房间或者撞上家具等行为，在遇到墙壁或者家具的时候会进行方向的随机改变，并且清理功能表现正常，每到一个地板

块，地板块上的灰尘量都会减少。



b) Faulty Robot 在 Empty Room 的测试

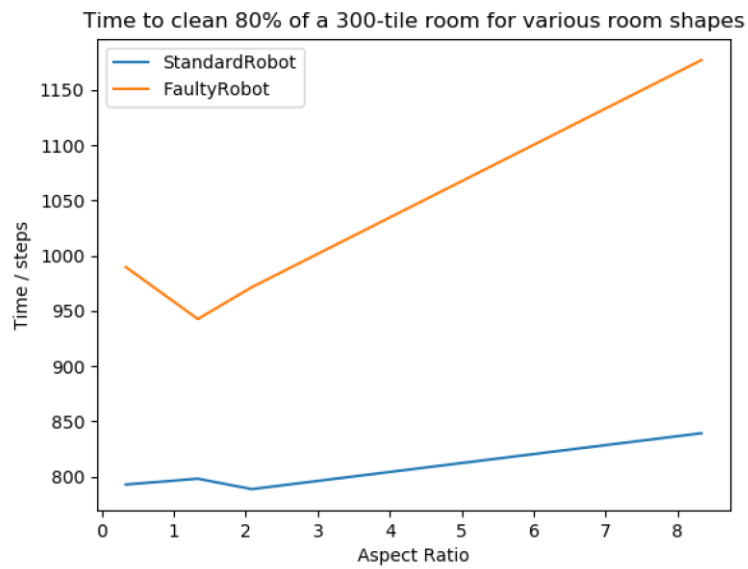
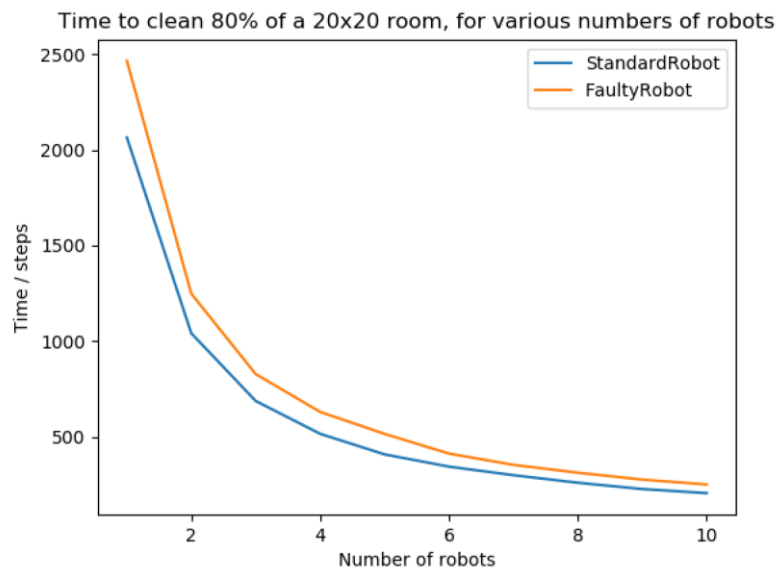


c) 模拟不同情况下机器人清理房间的情形并计算平均时间

```
PS C:\coding\Python\17340027-姚洁倩-p3> python .\ps3.py
avg time steps: 308.98
avg time steps: 566.84
avg time steps: 698.42
avg time steps: 1241.8
avg time steps: 420.34
```

d) 绘制图表

分析的结果在解题思路的第六点中



e) 可视化模拟过程

下图是使用三个机器人清理比率为 50%的清醒的其中一个模拟截图。



The grid world is a 20x20 grid. The start cell is at (1, 1) and the goal cell is at (18, 18). Obstacles are located at various cells, including (3, 3), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (3, 10), (3, 11), (3, 12), (3, 13), (3, 14), (3, 15), (3, 16), (3, 17), (3, 18), (3, 19), (3, 20), (4, 3), (4, 4), (4, 5), (4, 6), (4, 7), (4, 8), (4, 9), (4, 10), (4, 11), (4, 12), (4, 13), (4, 14), (4, 15), (4, 16), (4, 17), (4, 18), (4, 19), (4, 20), (5, 3), (5, 4), (5, 5), (5, 6), (5, 7), (5, 8), (5, 9), (5, 10), (5, 11), (5, 12), (5, 13), (5, 14), (5, 15), (5, 16), (5, 17), (5, 18), (5, 19), (5, 20), (6, 3), (6, 4), (6, 5), (6, 6), (6, 7), (6, 8), (6, 9), (6, 10), (6, 11), (6, 12), (6, 13), (6, 14), (6, 15), (6, 16), (6, 17), (6, 18), (6, 19), (6, 20), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8), (7, 9), (7, 10), (7, 11), (7, 12), (7, 13), (7, 14), (7, 15), (7, 16), (7, 17), (7, 18), (7, 19), (7, 20), (8, 3), (8, 4), (8, 5), (8, 6), (8, 7), (8, 8), (8, 9), (8, 10), (8, 11), (8, 12), (8, 13), (8, 14), (8, 15), (8, 16), (8, 17), (8, 18), (8, 19), (8, 20), (9, 3), (9, 4), (9, 5), (9, 6), (9, 7), (9, 8), (9, 9), (9, 10), (9, 11), (9, 12), (9, 13), (9, 14), (9, 15), (9, 16), (9, 17), (9, 18), (9, 19), (9, 20), (10, 3), (10, 4), (10, 5), (10, 6), (10, 7), (10, 8), (10, 9), (10, 10), (10, 11), (10, 12), (10, 13), (10, 14), (10, 15), (10, 16), (10, 17), (10, 18), (10, 19), (10, 20), (11, 3), (11, 4), (11, 5), (11, 6), (11, 7), (11, 8), (11, 9), (11, 10), (11, 11), (11, 12), (11, 13), (11, 14), (11, 15), (11, 16), (11, 17), (11, 18), (11, 19), (11, 20), (12, 3), (12, 4), (12, 5), (12, 6), (12, 7), (12, 8), (12, 9), (12, 10), (12, 11), (12, 12), (12, 13), (12, 14), (12, 15), (12, 16), (12, 17), (12, 18), (12, 19), (12, 20), (13, 3), (13, 4), (13, 5), (13, 6), (13, 7), (13, 8), (13, 9), (13, 10), (13, 11), (13, 12), (13, 13), (13, 14), (13, 15), (13, 16), (13, 17), (13, 18), (13, 19), (13, 20), (14, 3), (14, 4), (14, 5), (14, 6), (14, 7), (14, 8), (14, 9), (14, 10), (14, 11), (14, 12), (14, 13), (14, 14), (14, 15), (14, 16), (14, 17), (14, 18), (14, 19), (14, 20), (15, 3), (15, 4), (15, 5), (15, 6), (15, 7), (15, 8), (15, 9), (15, 10), (15, 11), (15, 12), (15, 13), (15, 14), (15, 15), (15, 16), (15, 17), (15, 18), (15, 19), (15, 20), (16, 3), (16, 4), (16, 5), (16, 6), (16, 7), (16, 8), (16, 9), (16, 10), (16, 11), (16, 12), (16, 13), (16, 14), (16, 15), (16, 16), (16, 17), (16, 18), (16, 19), (16, 20), (17, 3), (17, 4), (17, 5), (17, 6), (17, 7), (17, 8), (17, 9), (17, 10), (17, 11), (17, 12), (17, 13), (17, 14), (17, 15), (17, 16), (17, 17), (17, 18), (17, 19), (17, 20), (18, 3), (18, 4), (18, 5), (18, 6), (18, 7), (18, 8), (18, 9), (18, 10), (18, 11), (18, 12), (18, 13), (18, 14), (18, 15), (18, 16), (18, 17), (18, 18), (18, 19), (18, 20), (19, 3), (19, 4), (19, 5), (19, 6), (19, 7), (19, 8), (19, 9), (19, 10), (19, 11), (19, 12), (19, 13), (19, 14), (19, 15), (19, 16), (19, 17), (19, 18), (19, 19), (19, 20), (20, 3), (20, 4), (20, 5), (20, 6), (20, 7), (20, 8), (20, 9), (20, 10), (20, 11), (20, 12), (20, 13), (20, 14), (20, 15), (20, 16), (20, 17), (20, 18), (20, 19), (20, 20).