

6.00: 计算机科学和编程导论

问题6: 模拟机器人

开始时间: 2007.10.23 星期二

时间:

- 部分 I 开始时间: 2007.10.26 星期五 11:00am
- 部分 II 开始时间: 2007.10.30 星期二 11:00am

安装 Matplotlib / pylab

你将需要安装 `pylab` 为部分问题创建数据图 (plots)

在 Windows and Mac OS X 上安装 pylab

1. 下载并安装 `Numpy`. 检查你机器上的 `Python` 版本 (in `IDLE`, 选择 `Help->About IDLE`) 并确定版本为 `Python 2.5`. 然后选择相对应的平台: [Windows](#), [Mac](#).
2. 然后下载并安装 `Matplotlib`. 然后选择相对应的平台: [Windows](#), [Mac](#).

在 Athena Linux 上使用 pylab

如果在 `Athena clusters` 上运行, 找一台运行 `Athena Linux` 的 `Intel` 机器 (也就是说, 不是 `Sun Solaris`). 然后打开命令提示符窗口, 打入如下指令来加载 `IDLE`:

```
athena% add 6.00
athena% source /mit/6.00/arch/share/bin/setup_matplotlib
athena% idle&
```

你必须遵循这些步骤只要每次在 `Athena` 机器上开始工作

简介

在这个问题中你将会给模拟机器人编写程序, 告诉它怎样在封闭的房间里运动

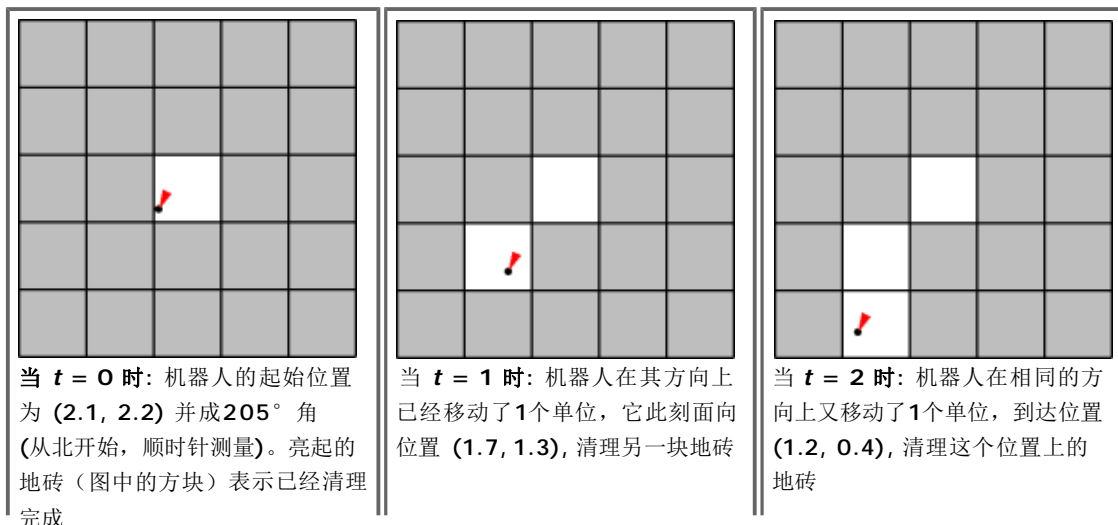
`iRobot` 是一家公司 (由 MIT 的校友和教职员创立) 销售 [Roomba vacuuming robot](#) (观看其中一个产品视频来了解这些机器是怎么行动的). `Roomba robots` 将会清理和打扫他们走过的地方

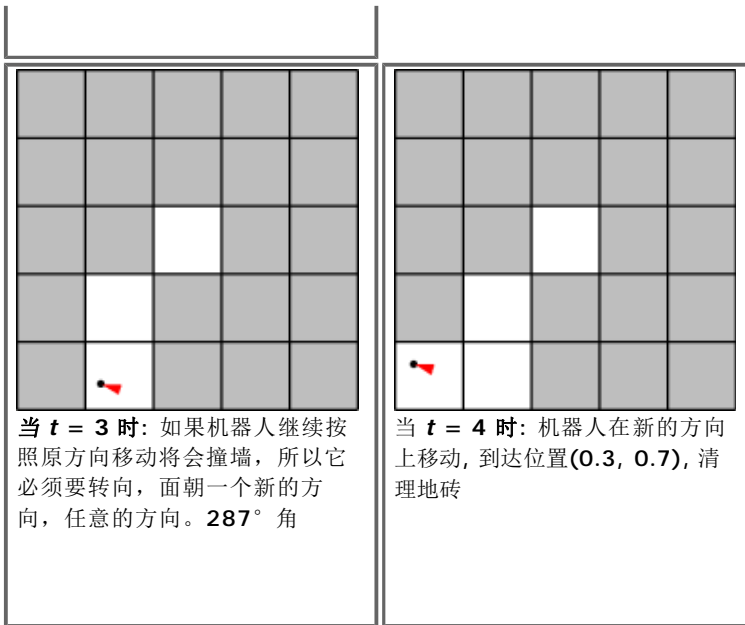
你将会设计一个模拟程序来估算一组类似于 `Roomba` 的机器人打扫一个矩形房间所用的时间

模拟概述

下列是单个机器人在一个 `5x5` 房间里移动的简化模型, 可以给你些感觉关于我们模拟的系统

机器人在房间里任意一个位置, 并且任意移动. 下面的图将阐述机器人的位置 (黑点表示) 和它的行进方向 (红色箭头表示).





模拟细节

这些是附加的模拟细节:

- **多个机器人.** 总的来说, 有 $N > 0$ 个机器人在房间里时, 这里的 N 是你程序的参数。为了简化, 假设机器人都是点并且可以没有影响的互相穿过。
- **房间.** 房间是宽 w 和高 h (都是整数) 的矩形, 宽和高都是你程序的参数。初始时所有的地板都是脏的。机器人不允许穿过墙到达房间。机器人不可以到达房间外的一点。
- **地砖.** 你将需要记录哪块地砖已经清理过了。我们将把房间分成一些 1×1 的格子, 标记为整数对: $(0, 0), (0, 1), \dots, (w-1, h-1)$ 当机器人的位置为任意一块地砖是, 我们要考虑到所有的地砖将被清理。
- **机器人移动规则:**
 - 每个机器人在房间里有个位置。我们将用坐标 (x, y) 表示位置, x, y 为实数, $0 \leq x < w, 0 \leq y < h$.
 - 机器人向着一个方向运动。我们将用角度 d 表示方向, 角度为整数, $0 \leq d < 360$.
 - 所有机器人以相同的速度 s 移动, 速度在初始时设定并且在以后的模拟中不改变。每个单位时间, 机器人在它的方向上移动 s 个单位。
 - 当机器人碰到墙壁, 它将随机选择新的方向, 沿着新的方向继续运动, 直到碰到另一块墙。(你的程序可以用不止一种的方法来让模拟机器人在墙周围移动, 自己选择一个合理的方法.)
- **结束.** 当特定的一块地砖被清理后, 模拟结束。

部分 I: 数据结构的设计

你将需要设计一个数据结构来记录哪块地砖已经被清理, 另一个数据结构来记录每个机器人的位置和方向。

下载 `ps6.py`, 其中包含了一些你需要用到得核心代码。

问题 #1

在这个部分中, 你将需要决定怎样表现模拟中的数据, 并且展现怎样演示你数据结构中的重要操作。

- 用数据结构来记录已经被清理过的地砖, 考虑怎样演示这些操作:
 - 初始化数据结构
 - 标记已清理完毕的地砖
 - 判断地砖是否被清理
 - 判断房间里有多少块地砖

- 判断房间中已经清理了多少块地砖
- 在房间中确定随机的位置
- 判断给定的位置是否在房间中
- 用数据结构来记录机器人的位置和方向，考虑怎样演示这些操作：
 - 初始化数据结构
 - 接入给定机器人的位置
 - 接入给定机器人的方向
 - 设置给定机器人的位置
 - 设置给定机器人的方向

完成下列方程来实施上述的操作。

(尽管这个部分看起来很复杂，但是它不会花费很长时间，一旦你决定则怎么表现你的数据。为了合理的表现，只需要如下主方程中的一行代码。)

```
def new_room(width, height):
    """
    返回一个新的，用WIDTH和HEIGHT的乘积，表示房间大小的数据结构，并储存房间中的每块地砖，无论
    是否被清理。初始情况下，所有地砖都没有被清理。
    """
    # 返回值将会作为 ROOM的参数，传给如下的方程。
    return # ... 此处为你的代码

def set_tile_cleaned(room, m, n):
    """
    房间中已经清理的地砖，标记为(m, n)。

    假设(m, n)表示房间中一块有效的地砖。
    """
    # 此处为你的代码

def is_tile_cleaned(room, m, n):
    """
    返回 True，如果房间中的地砖(m, n)已经被清理。

    假设(m, n)表示房间中一块有效的地砖。
    """
    # 此处为你的代码

def get_num_tiles(room):
    """
    返回房间中地砖的总数。
    """
    # 此处为你的代码

def get_num_cleaned_tiles(room):
    """
    返回房间中已经清理的地砖数。
    """
    # 此处为你的代码

def get_random_position(room):
    """
    返回一个房间中的随机位置(元组(x, y))。
    """
    # 此处为你的代码

def is_position_in_room(room, position):
    """
    返回 True，如果位置(元组(x, y))在房间中。
    """
    # 此处为你的代码

def new_robot_structure(n, room):
    """
    返回一个新的数据结构(我们叫它 '机器人数据结构')来存储房间中 n个不同机器人的位置和方向。
    位置和方向需要被随机的初始化。
    """
    # 返回值将会作为 robot_structure的参数，传给如下的方程
    return # ... 此处为你的代码
```

```

def get_robot_position(robot_structure, i):
    """
    机器人数据结构中返回第 i 个机器人的位置(元组(x, y)).

    假设 0 <= i < n, n 是机器人的总数.
    """
    # 此处为你的代码

def get_robot_direction(robot_structure, i):
    """
    在机器人数据结构中返回第 i 个机器人的方向(角度, 0 <= dir < 360).

    假设 0 <= i < n, n 是机器人的总数.
    """
    # 此处为你的代码

def set_robot_position(robot_structure, i, position):
    """
    在机器人数据结构中设置第 i 个机器人的位置到(元组(x, y)).

    假设 0 <= i < n, n 是机器人的总数, 位置表示房间中有效的位置.
    """
    # 此处为你的代码

def set_robot_direction(robot_structure, i, dir):
    """
    在机器人数据结构中设置第 i 个机器人的方向到 DIR(一个角度).

    假设 0 <= i < n, n 是机器人的总数, DIR 是有效的角度(0 <= dir < 360).
    """
    # 此处为你的代码

```

你可以用自己的方式和任何一种数据结构来表示数据。无论你怎么表示，重要的事是，当一起使用时，你的方程可以一如既往的良好运行。例如，如果我们运行如下的代码，我们可以得到确定的答案：

```

r = new_room(10, 10) # 创建一个 10x10 的房间并叫它 r.
num_tiles = get_num_tiles(r) # 这个值应为100.
x = is_tile_cleaned(r, 5, 7) # 这个值应为False, 因为房间 r 中还没有地砖被清理.
set_tile_cleaned(r, 8, 1) # 这个值应为True, 因为我们刚刚标记过它.
y = is_tile_cleaned(r, 8, 1) # 房间 r 中被清理的地砖.

```

部分 II: 创建模拟器

问题 #2

写下模拟机器人的代码，如上所述。目的是为了确定在房间中特定的地砖被清理之前，需要多少时间步长，完成如下的方程：

```

def run_simulation(num_robots, speed, width, height, min_coverage, num_trials):
    """
    运行 NUM_TRIALS 模拟实验并且返回清理房间中 MIN_COVERAGE 部分所需要的平均时间.

    用 NUM_ROBOTS 机器人运行模拟, 每个的速度为 SPEED, 在面积为 WIDTH x HEIGHT 的房间中.
    """
    # 此处为你的代码

```

写出任何你感觉有用的方程。

你会用到如下有用的方程 (你将会用到 `import math`):

```
def get_new_position(coords, angle, speed):
    """
    在一个时钟周期之后, 计算并返回新位置 (x,y), 给出现在的位置, 角度, 和速度.

    不要测试返回位置是否在房间内.

    坐标: 元组(x, y)代表现在的位置
    角度: 浮点数代表角度, 0 <= angle < 360
    速度: 正的浮点数代表速度

    返回值: 元组(x,y)代表新的 x, y坐标
    """
    old_x, old_y = coords

    # 计算位置的改变
    delta_y = speed * math.cos(math.radians(angle))
    delta_x = speed * math.sin(math.radians(angle))

    # 把它加到现有的位置里
    new_x = old_x + delta_x
    new_y = old_y + delta_y

    return (new_x, new_y)
```

如下是一些打扫房间的大概所需时间. 这些时间都是在机器人速度为 **1.0**的情况下得出.

- 一个机器人用大概 **150**个时钟周期完全打扫完一个 **5x5**的房间.
- 一个机器人用大概 **190**个时钟周期打扫一个 **10x10**房间的**75%**.
- 一个机器人用大概 **310**个时钟周期打扫一个 **10x10**房间的**90%**.
- 一个机器人用大概 **3250**个时钟周期完全打扫完一个 **20x20**的房间.

(这些仅供参考. 取决于你实施的准确细节, 你可能会得到与我们不同的时间.)

你应该测试你模拟的结果在速度不为 **1.0**的情况. 其中的一个方法就是, 在上述的测试中, 改变速度, 并得到合理的结果.

可视化机器人

我们已经提供一些代码来为你的机器人产生动画, 当它们去打扫房间的时候. 这些动画可以帮助你调试你的模拟, 在直观上判断哪里出了问题.

下载 `ps6_visualize.py` 并且与 `ps6.py`保存在相同的目录下. 添加如下的代码到 `ps6.py`的最前端:

```
import ps6_visualize
```

怎样运行可视化:

1. 在你的模拟中, 在实验开始的时候, 按照如下的做法来开始动画:

```
anim = ps6_visualize.RobotVisualization(num_robots, width, height)
```

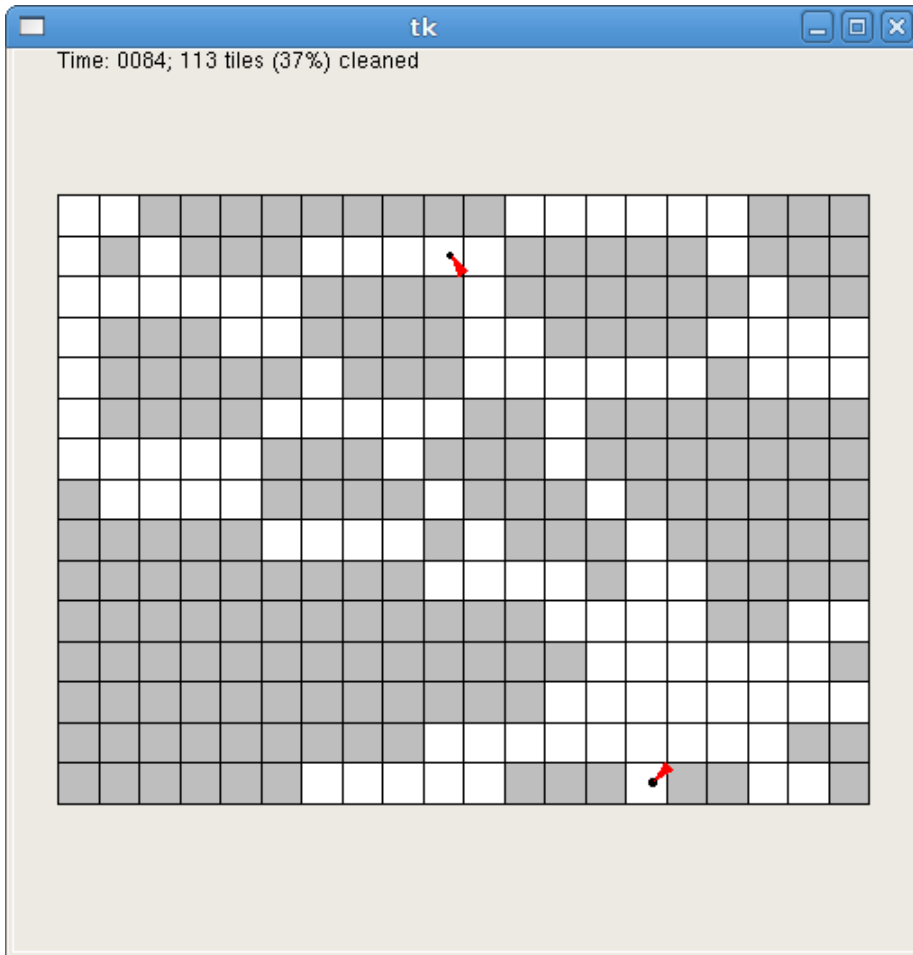
(当然要传输恰当的参数给实验.) 这将会打开一个新的窗口来显示动画并画出房间的图像.

2. 然后, 每个时间周期, 按照如下的做法来画出新的动画:

```
anim.update(room, robots)
```

传递进数据结构, 表示现阶段房间和机器人的状态. 你必须使用相同的数据结构, 在问题**1**中定义的, 因为我们将会使用你在问题**1**中写的方程来得到数据.

最终的动画将与如下所示的图像很相像:



可视化代码使你的模拟变得缓慢，以至于动画不能很快的传送（默认的情况下，每秒5个时间步长）。很自然的，当你一次运行许多实验的时候，你倾向于不使用动画代码（例如，当你运行全部的模拟）。

为了调试你的模拟，你可以进一步减慢动画。你可以通过运行 `RobotVisualization`来达到目的，如下：

```
anim = ps6_visualize.RobotVisualization(num_robots, width, height, delay)
```

参数的延迟说明了，两幅画面间的停顿有多少秒。默认的是0.2（每秒5幅画面）。你可以提高这个数值使得动画更慢。

问题 #3

现在，用你的模拟来回答一些关于机器人表现相关的问题。

对如下3个问题中的每一个，用 `pylab`写下可以产生数据图（plot）的代码。把你的代码放进 `ps6.py`中相对应的 `skeleton`方程中。

每个数据图（plot）在x轴和y轴上都应该有标题和描述。

1. 每个1-10 机器人打扫大小为 `20x20`房间的80%用多长时间？输出的图像，绘制出时间（y轴）和机器人数量之间的关系。
2. 用3个机器人分别打扫大小为`10x10`，`20x20`，`30x30`，和`40x40`房间的70%各用多少时间？输出的图像，绘制出时间（y轴）和房间的大小之间的关系（单位：每平方英尺）。
3. 用2个机器人分别打扫大小为 `20x20`，`25x16`，`40x10`，`50x8`，`80x5`，和`100x4`房间的80%用多长时间？（注意这些房间的面积都是一样的）。输出的图像，绘制出时间（y轴）和宽高之间比率的关系。

多次试验。对于你的数据图，进行大量的实验来确保可靠的结果。

上交程序

1. 保存

你所有的代码应该在单个的名称为 `ps6.py` 的文件里。

2. 时间和合作者的信息

在文件的开始，在注释里，写下所用的小时数（大概）在这个问题上，并写下合作者的姓名。例如：

```
# 问题 6
# 姓名: Jane Lee
# 合作者: John Doe
# 时间: 1:30
#
```

3. 上交

上交文件，上传至你的工作区。如果上传出现问题，用email发送文件到6.00 职员处。