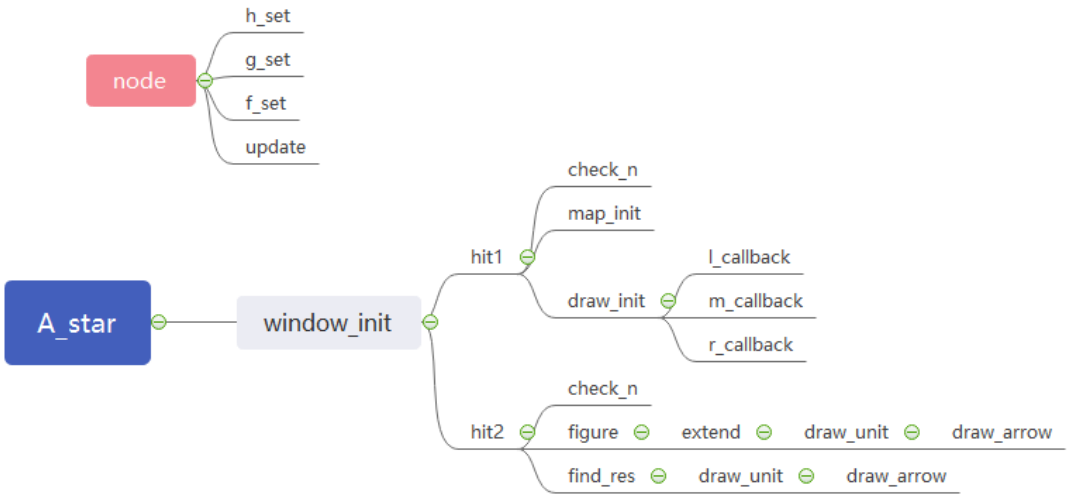
一．编译环境

在 ubuntu16.04, python3.7 下进行编程, 运用了 tkinter、numpy、time、tkinter.massagebox 模块

二．代码思路简述

由于不确定是否可以让物体斜角穿越障碍物, 所以根据在扩展节点处的代码做了微小的修改，提交了两个版本。

一共有 1 个类，14 个函数，关系如下:



（程序流程图）

1. Node
   1) 初始化 node 时，需要提供 father node(start node 和 end node 的父节点为 none)，row，col；
   2) Node 中的 h_set、g_set、f_set 函数，用来计算 A*的衡量标准；
   3) 当某个节点第二次被扩展到的时候，调用 update，通过比较 h 决定是否更新父亲节点。
2. Window_init 函数

   初始化窗口界面，创建 1100x1000 的窗口、一个 label、一个 text、两个 button
3. Hit1 函数:
   1) 包含 check_n 函数，用来检测在点击 hit1 前，是否已经输入了合法的 n；
   2) 调用 map_init 函数，初始化多个元组；
   3) 调用 draw_init 函数，建立棋盘。
4. Check_n 函数:

   通过检测 text 文本的长度和使用 len 时是否出错来决定输入的 n 是否是合法的。如果合法，返回 1；否则返回 0。
5. Map_init 函数

   初始化 openlist、closelist、障碍物群组 obstruction、用于防止画图覆盖的 searchlist。
6. Draw_init 函数
   1) 建立棋盘；
   2) 建立棋盘格子坐标和画布位置坐标对应的转换元组；
   3) 调用 3 个 callback 函数，绘制 start node、end node 和 obstruction node。
7. L_callback 函数

   当鼠标左键单击时，获取鼠标当前位置，把位置坐标转化为棋盘坐标，讲其设置为

start node 并添加到 openlist 并绘图。

8. M_callback 函数
   当鼠标中键单击时，获取鼠标当前位置，把位置坐标转化为棋盘坐标，讲其添加到 obstructlist 和 closelist 中，并绘图。
9. R_callback 函数
   当鼠标右键单击时，获取鼠标当前位置，把位置坐标转化为棋盘坐标，讲其设置为 end node 并绘图。
10. Hit2 函数
    1） 调用 check_n 函数，用来检测在点击 hit2 前，是否已经输入了合法的 n；
    2） 利用全局变量 done，检测在点击 hit2 前，是否已经点击过 hit1；
    3） 调用 figure 函数，不断迭代来扩展节点并绘图；
    4） 调用 find_res 函数，当成功找到路径时，用 end node 回溯至 start node，将路径保存并绘图。
11. Figure 函数
    通过迭代来扩展节点，反复调用 extend 函数，直到扩展到 end node。
12. Extend 函数
    1） 输入为作为父节点的 point，调用结果为将 point 从 openlist 移入 closelist，并将数个新的 node 添加到 openlist 中；
    2） 当父节点周围 8 个节点在 closelist（障碍物或超出地图边界）中时，不扩展（添加到 openlist）；
    3） 当父节点周围的节点 M 已在 openlist 中时，调用 update 函数对 M 进行更新，并覆盖输出新的 h，g，f；
    4） 当不满足 2 和 3 条件时，将被扩展的节点添加到 openlist，并绘图，表明其已经被扩展过了。
13. Draw_unit 函数
    1） 输入为节点 point，颜色 color 和 flag（默认为 0），输出为对 point 节点对应的位置进行绘图
    2） 将 point 的 f、h、g 值绘制在格子中，讲格子的边绘为 color 对应的颜色
    3） 用 flag 保证在绘制扩展节点时，被扩展的节点不会对已走过的节点进行覆盖。
    4） 调用 draw_arrow 函数，绘制由子节点指向父亲节点的箭头
14. Draw_arrow 函数
    1） 输入为节点 point，输出为由 point 格子点指向 point.father 格子点的箭头
    2） 获取 point 和 point.father 的中心位置，进行五等分，选择靠近 point 的 1/5 分位点 M
    3） 绘制从 point 到 M 的箭头，利用同样的原理绘制箭头的侧翼。
15. Find_res 函数
    1） 获取 end node，进行回溯寻找父亲节点，直到找到 start node，将走过的所有路径添加到 res2 元组中
    2） 将 res2 反转，得到 res
    3） 调用 draw_unit 绘制 res 中的所有节点
    4） 输出弹窗，提示搜索成功或无结果。

三．编写代码的收获和遇到的问题

1. 在扩展节点时，一直想调用 sleep 函数来让绘图的速度变慢，但是一直卡死，后来发现需要调用 canvas.update 函数对画布进行更新。

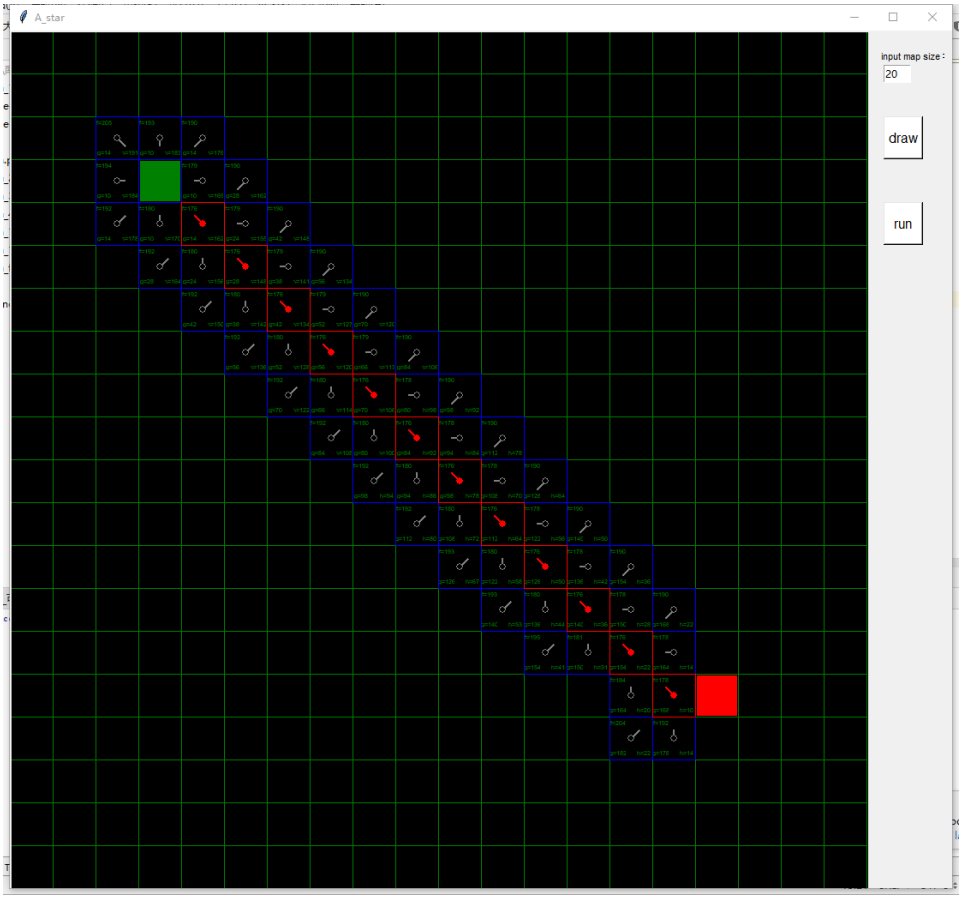
2. 调用 tkinter 的 Messagebox 时，需要额外 import 一下。
3. 在扩展节点时，绘制新的 node 的边界时，会将父节点的边界覆盖（颜色覆盖），新添加了一个元组来防止这种情况发生。
4. 在画棋盘时，会出现最右侧一类和最下面一行的大小和其他不一样的情况，通过动态设置 canvas 的大小来解决了这个问题。
5. 第一次在 python 接触到了鼠标和页面的交互，有所收获。最开始以为只有 frame 可以调用 bind 函数，后来发现 canvas 也可以调用，虽然在最初走了弯路，但在解决问题的过程中收获很多。

四．程序使用说明

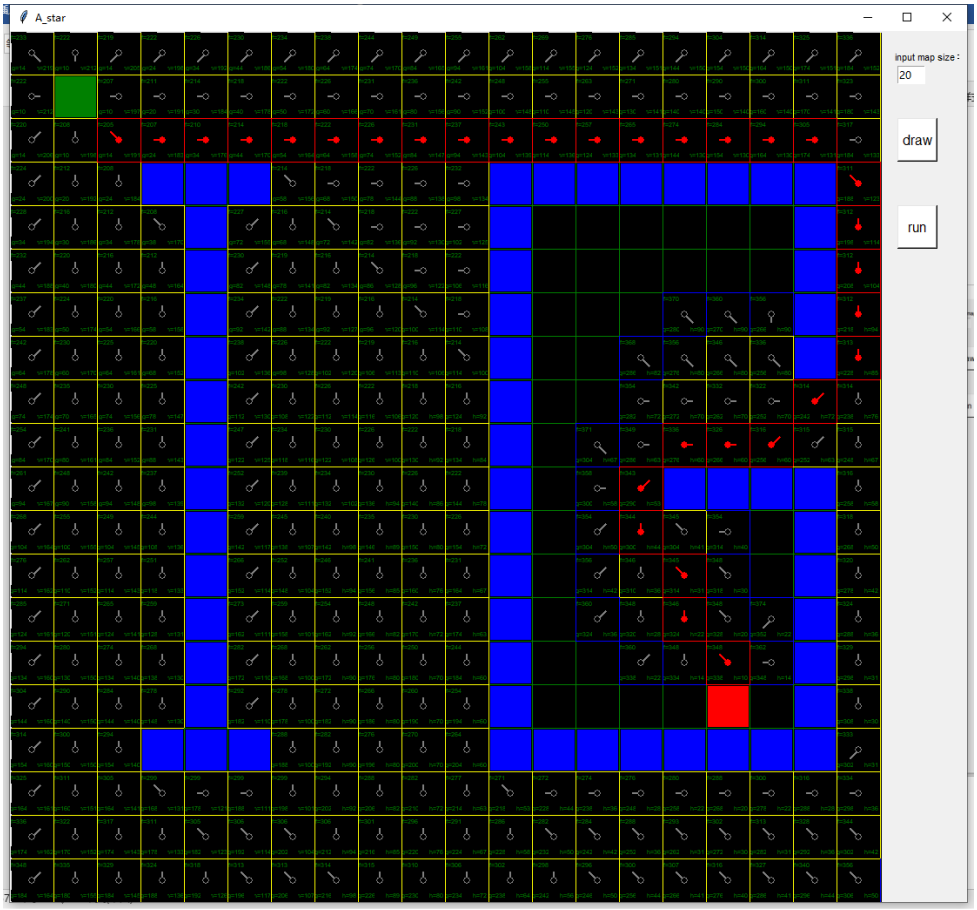
1. 首先输入该问题的 map size（n）；
2. 单击"draw"，绘制棋盘；
3. 鼠标左键选取开始节点，右键选取结束节点，中键选取障碍物节点；
4. 点击"run"，开始求解；
5. 求解完毕后，会弹窗提示求解成功或失败。

五．代码效果展示

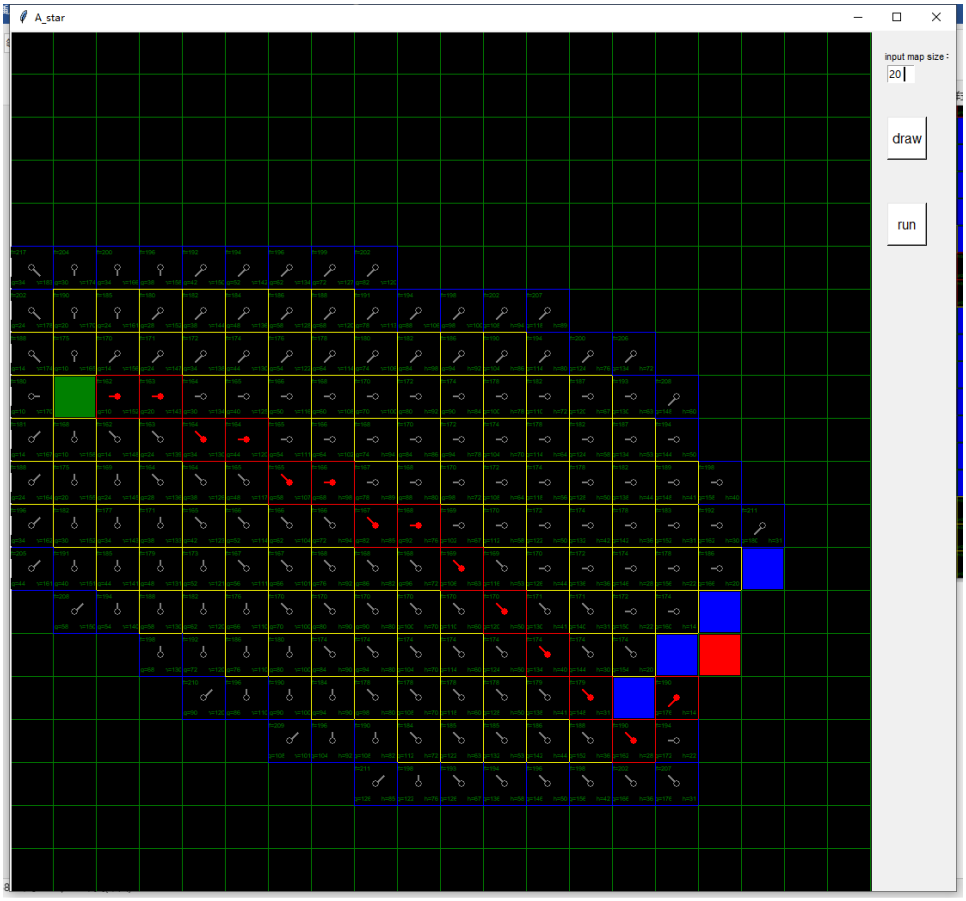
1．无障碍物运行



2．有障碍物运行


A_star
input map size :
20
draw
run

3．特殊障碍物（交叉）


A_star
input map size :
20
draw
run