A星算法实验报告

姓名：杨帆　学号：1711503　专业：智能科学与技术

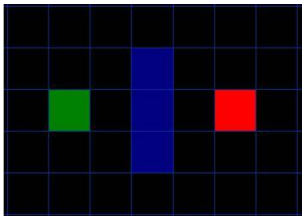
一、编译环境

- 语言：python
- 相关包调用：numpy、tkinter、math

二、算法解析

- 概述

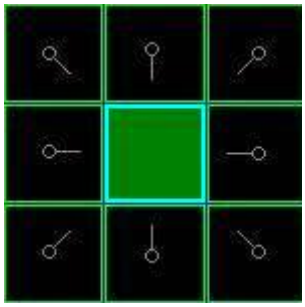
  假设某人要从 A 点移动到 B 点，但是这两点之间被一堵墙隔开。如图，绿色是 A，红色是 B，中间蓝色是墙。



把搜寻的区域划分成了正方形的格子。这个方法把我们的搜索区域简化为成 2 维数组。方格的中心点称为节　点（node）。

- 开始搜索

  1. 将A放入open list(开放列表) 中。现在 open list 里只有一项，它就是起点 A，后面会慢慢加入更多的项。 Open list 里的格子是路径可能会是沿途经过的，也有可能不经过。基本上 open list 是一个待检查的方格列表。
  2. 查看与起点 A 相邻的方格，把其中可走的 (walkable) 或可到达的 (reachable) 方格也加入到 open list 中。把起点 A 设置为这些方格的父亲 (parent node) 。
  3. 把 A 从 open list 中移除，加入到 close list(封闭列表) 中， close list 中的每个方格都是现在不需要再关注的。如下图所示，深绿色的方格为起点，它的外框是亮蓝色，表示该方格被加入到了 close list 。与它相邻的黑色方格是需要被检查的，他们的外框是亮绿色。每个黑方格都有一个灰色的指针指向他们的父节点，这里是起点 A 。



- 路径排序

  F = G + H

  - G = 从起点 A 移动到指定方格的移动代价，沿着到达该方格而生成的路径。
  - H = 从指定的方格移动到终点 B 的估算成本。（本次实验中我采用曼哈顿距离）

  我们的路径是这么产生的：反复遍历 open list ，选择 F 值最小的方格。

  为了继续搜索，我们从 open list 中选择 F 值最小的 (方格) 节点，然后对所选择的方格作如下操作：

1. 把它从 open list 里取出，放到 close list 中。
2. 检查所有与它相邻的方格，忽略其中在 close list 中或是不可走 (unwalkable) 的方格（障碍），如果方格不在open list 中，则把它们加入到 open list 中。
3. 如果某个相邻的方格已经在 open list 中，则检查这条路径是否更优，也就是说经由当前方格 (我们选中的方格) 到达那个方格是否具有更小的 G 值。如果没有，不做任何操作。

- 停止搜索
  - 当把终点加入open list时，搜索结束，找到路径。从终止节点沿父指针方向回溯，即为路径。
  - 当open list表为空时，即无法找到路径，查找失败。

三、核心代码分析

```
    def search(self):
        self.put_into_openlist(self.Start_node_tuple)#将初始节点放入open表中
        end_node_flag = 1
        #一直循环直至结束
        while end_node_flag:
            current_node = self.find_min_F_openlist()#将open表中F值最小的节点作为当前节点
            if -1 < current_node[0] < 30 and -1 < current_node[1] < 20:
                self.take_out_of_openlist(current_node)#将当前节点从open表中取出
                self.put_into_closelist(current_node)#将当前节点放入close表中
                self.checking_adjacent_area(current_node)#检查周围相邻节点看是否更优
                #终点出现在open list中，搜索结束，找到路径
                if self.End_node_tuple in self.openlist_dict:
                    self.paint_path()
                    tkinter.messagebox.showinfo(title="提示", message="找到路径!")
                    end_node_flag = 0
                #open list为空，搜索结束，无路径
                if not self.openlist_dict:
                    tkinter.messagebox.showinfo(title="提示", message="没有路径!")
                    end_node_flag = 0
```

即对应如下伪代码：

1. 把起点加入 open list 。

2. 重复如下过程:

a. 遍历 open list ，查找 F 值最小的节点，把它作为当前要处理的节点。

b. 把这个节点移到 close list 。

c. 对当前方格的 8 个相邻方格的每一个方格?

- 如果它是不可抵达的或者它在 close list 中，忽略它。否则，做如下操作。
- 如果它不在 open list 中，把它加入 open list ，并且把当前方格设置为它的父亲，记录该方格的 F ， G 和 H 值。
- 如果它已经在 open list 中，检查这条路径 (即经由当前方格到达它那里) 是否更好，用 G 值作参考。更小的 G 值表示这是更好的路径。如果是这样，把它的父亲设置为当前方格，并重新计算它的 G 和 F 值。如果你的 open list 是按 F 值排序的话，改变后你可能需要重新排序。

d. 停止，当你

- 把终点加入到了 open list 中，此时路径已经找到了，或者
- 查找终点失败，并且 open list 是空的，此时没有路径。

3. 保存路径。从终点开始，每个方格沿着父节点移动直至起点，这就是你的路径。

四、实验总结

- 了解了基本的非启发式寻路算法（迪杰斯特拉）和启发式寻路算法（A星）；
- 了解了A星算法的基本原理并将其做可视化实现；
- 熟悉tkinter绘图的基本用法，相比于上一次使用八皇后的QT而言，虽然界面没有QT美观，但使用方便，代码的可写性强，但是可读性差。