

（深圳）

实验报告

开课学期： 大三上学期

课程名称： 人工智能

实验名称： 搜索策略

实验性质： 设计

实验时间： 地点： 实验台号

学生专业： 计算机科学与技术

学生学号： SZ170110202

学生姓名： 陈抒语

评阅教师：

报告成绩：

实验与创新实践教育中心印制

2019年10月

**小组贡献：25%**

**吃豆人**

**一．简介/问题描述**

**1.1待解决问题的解释**

## 问题1：应用深度优先算法找到一个特定的位置的豆

## 问题2：应用宽度优先算法找到一个特定位置的豆

## 问题3：使用代价一致算法，根据各种路线的代价规划路线，吃到豆子

## 问题4：使用A\* 算法，根据代价预估，规划路线，吃到特定位置的豆子

## 问题5：找到一条最佳的路线，可以访问四个角落的豆子

## 问题6：使用启发式算法，找到一条最佳路线，根据预估代价，规划路线，吃到四个角落的豆子

## 问题7：使用启发式函数，在尽可能少的步数里吃完所有的豆子

## 问题8：进行算法优化，定义一个优先吃最近的豆子的函数，从而提高搜索速度

**1.2问题的形式化描述**

如果四周没有墙壁，吃豆人可以向上下左右四个方向移动，用up/down/left/right来表达四个方向，在没有加入代价估值函数前，会按照上下左右的顺序，检查是否有豆子，是否可达，当加入了代价估值函数的时候，会得到对于四个方向的代价估值，根据比较代价的大小， 选择适当的路径进行前进。

**1.3解决方案介绍（原理）**

使用广度优先搜索、深度优先搜索等基础算法，并在这两种基础算法上进行优化，配合使用各种估计代价的函数以及启发式函数，对搜索的顺序、搜索的路径进行优化，选择较小代价的路线优先进行扩展，避免基础的深度优先搜索方法与广度优先搜索方法存在的一些极端情况下过慢的状况，优化了搜索，并且在搜索的过程中，各种情况下，例如代价相等的情况下，也可以给出搜索算法一定的决断策略。

**二．算法介绍**

**2.1 所用方法的一般介绍**

**1.深度优先搜素**：

深度优先遍历图的方法是，从图中某顶点v出发：

（1）访问顶点v；

（2）依次从v的未被访问的邻接点出发，对图进行深度优先遍历；直至图中和v有路径相通的顶点都被访问；

（3）若此时图中尚有顶点未被访问，则从一个未被访问的顶点出发，重新进行深度优先遍历，直到图中所有顶点均被访问过为止。

**2.广度优先搜索**：

广度优先搜索使用队列（queue）来实现

1、把根节点放到队列的末尾。

2、每次从队列的头部取出一个元素，查看这个元素所有的下一级元素，把它们放到队列的末尾。并把这个元素记为它下一级元素的前驱。

3、找到所要找的元素时结束程序。

4、如果遍历整个树还没有找到，结束程序

1. **代价一致算法：**

使用代价一致算法的搜索总是扩展路径消耗最小的结点n，n点的路径消耗等于前1结点n-1的路径到n的路径消耗。每次都选择到达该点代价最小的点进行扩展。

1. **A\*算法**

A\* 算法是一种静态路网中求解最短路径最有效的直接搜索方法，也是许多其他问题的常用启发式算法。

公式表示为： f(n)=g(n)+h(n),

其中， f(n) 是从初始状态经由状态n到目标状态的代价估计，

g(n) 是在状态空间中从初始状态到状态n的实际代价，

h(n) 是从状态n到目标状态的最佳路径的估计代价。

每次选择f（n）最小的点进行扩展，扩展后重新选择包括新产生的的结点中f(n)最小的进行扩展，直到完成任务。

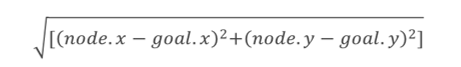
1. **启发式搜索：**

启发式搜索，它是利用问题拥有的启发信息来引导搜索，达到减少搜索范围、降低问题复杂度的目的，这种利用启发信息的搜索过程称为启发式搜索。选择适合的启发函数就能简化搜索，在本次实验中利用曼哈顿距离作为启发式寻路算法，曼哈顿距离：两点在南北方向上的距离加上东西方向上的距离，就曼哈顿距离的概念来说，只能上、下、左、右四个方向进行移动，而且两点之间的曼哈顿距离是两点之间的最短距离（在只能向上、下、左、右四个方向进行移动的前提下）。

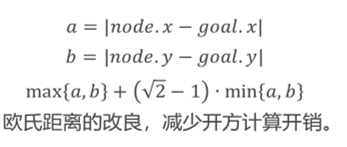
各种启发式函数：

曼哈顿：|X1-X2|+|Y1-Y2|(只能用于四方向网格地图）

切比雪夫：max{|X1-X2|-|Y1-Y2|}(可以用于八方向移动网络地图）

Euclidean：（可以用于任何方向的地图）

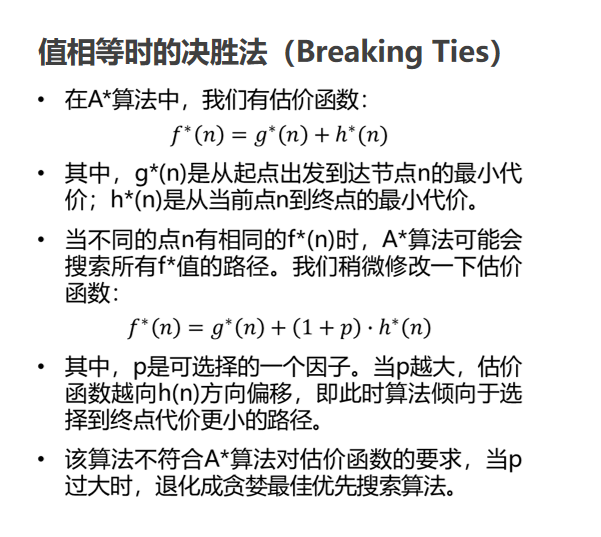
Octile：



1. **次优代价搜索：**

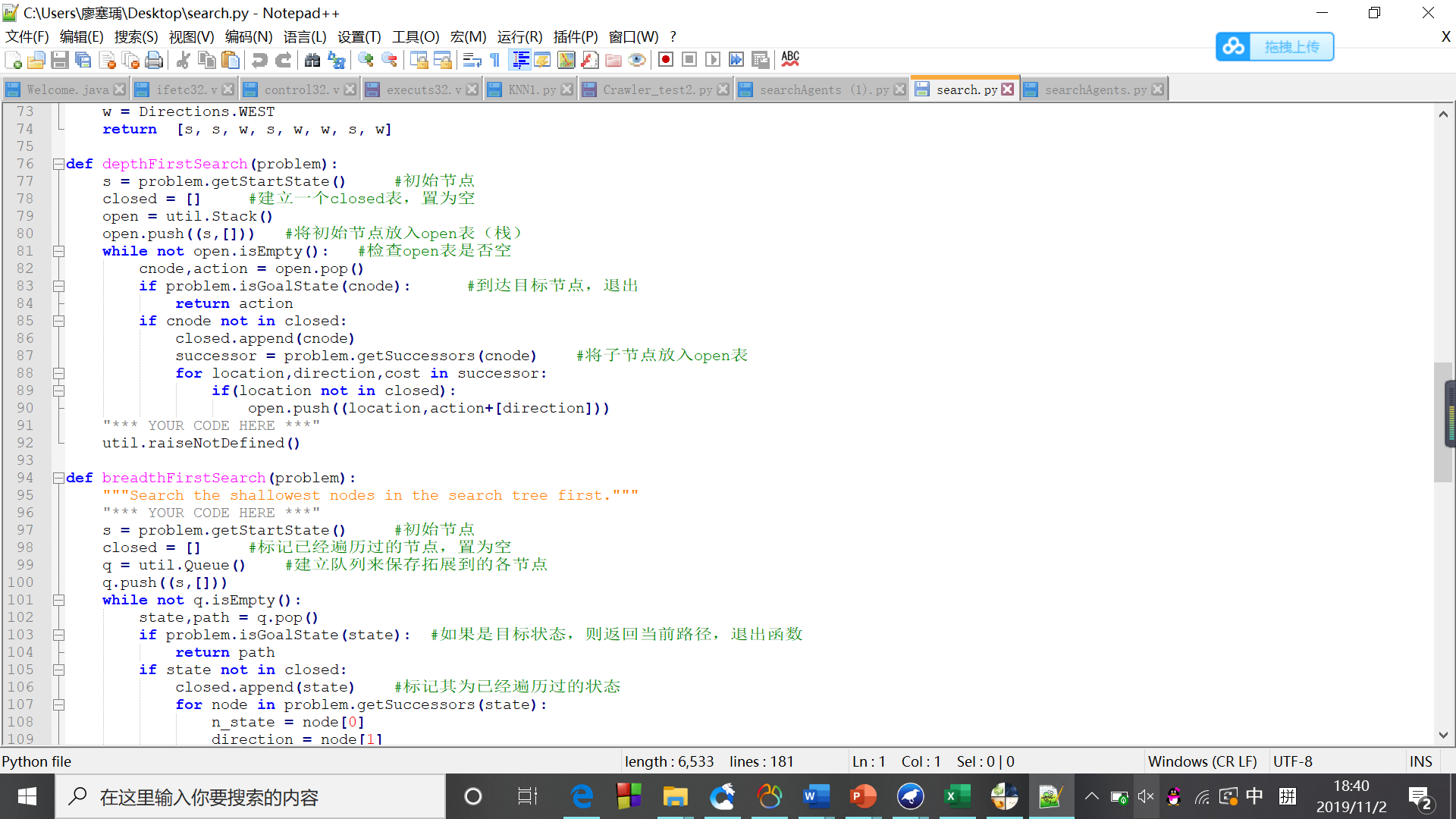
由于构造最优查找树代价较大，所以采用比较折中的方法，构造次优查找树，不仅能优化查找，而且可以减小开销。先选择最近的豆子吃掉，完成最容易完成的部分。

1. **值相等时的决胜法：**

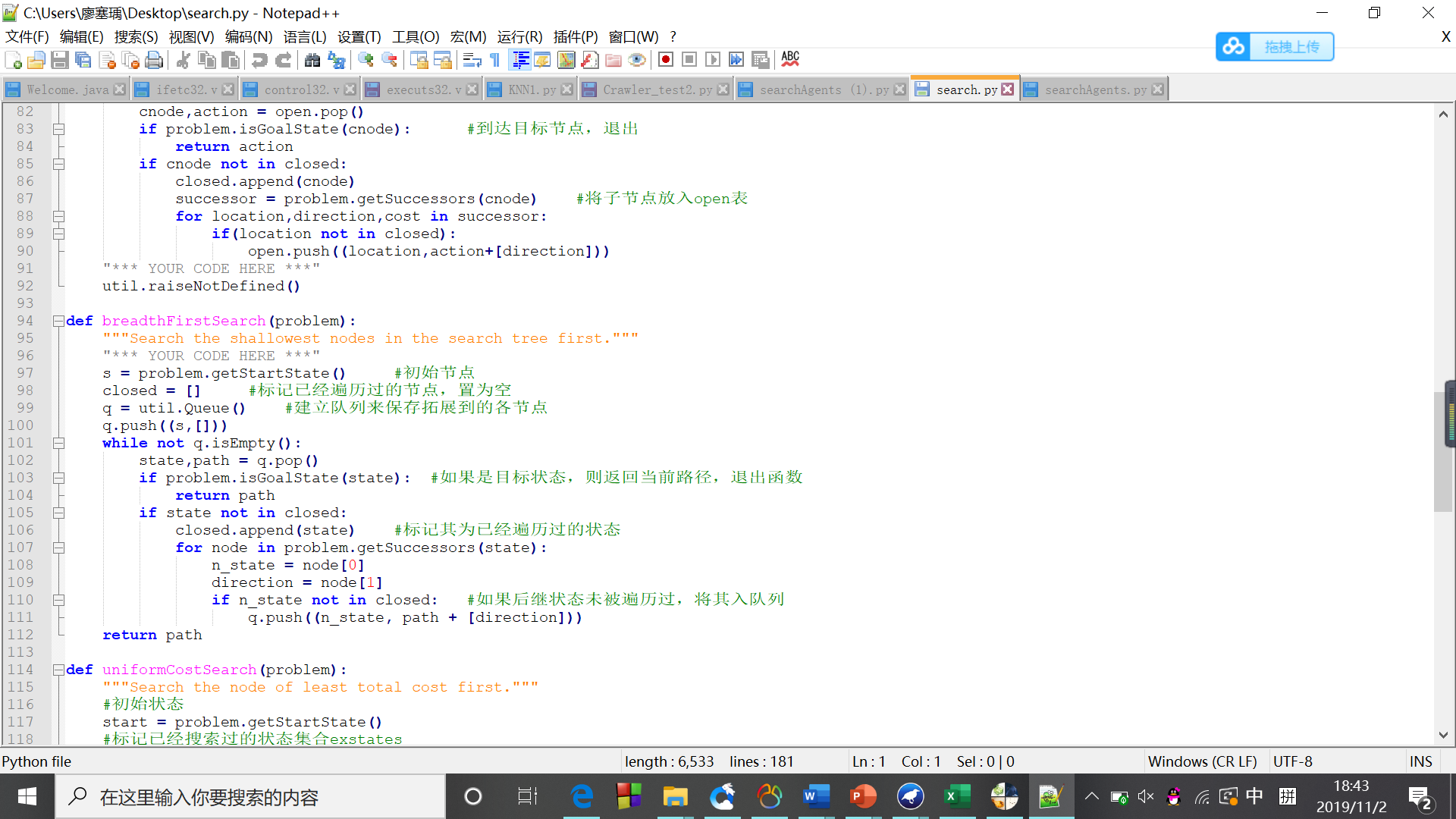
****

**2.2 算法伪代码**

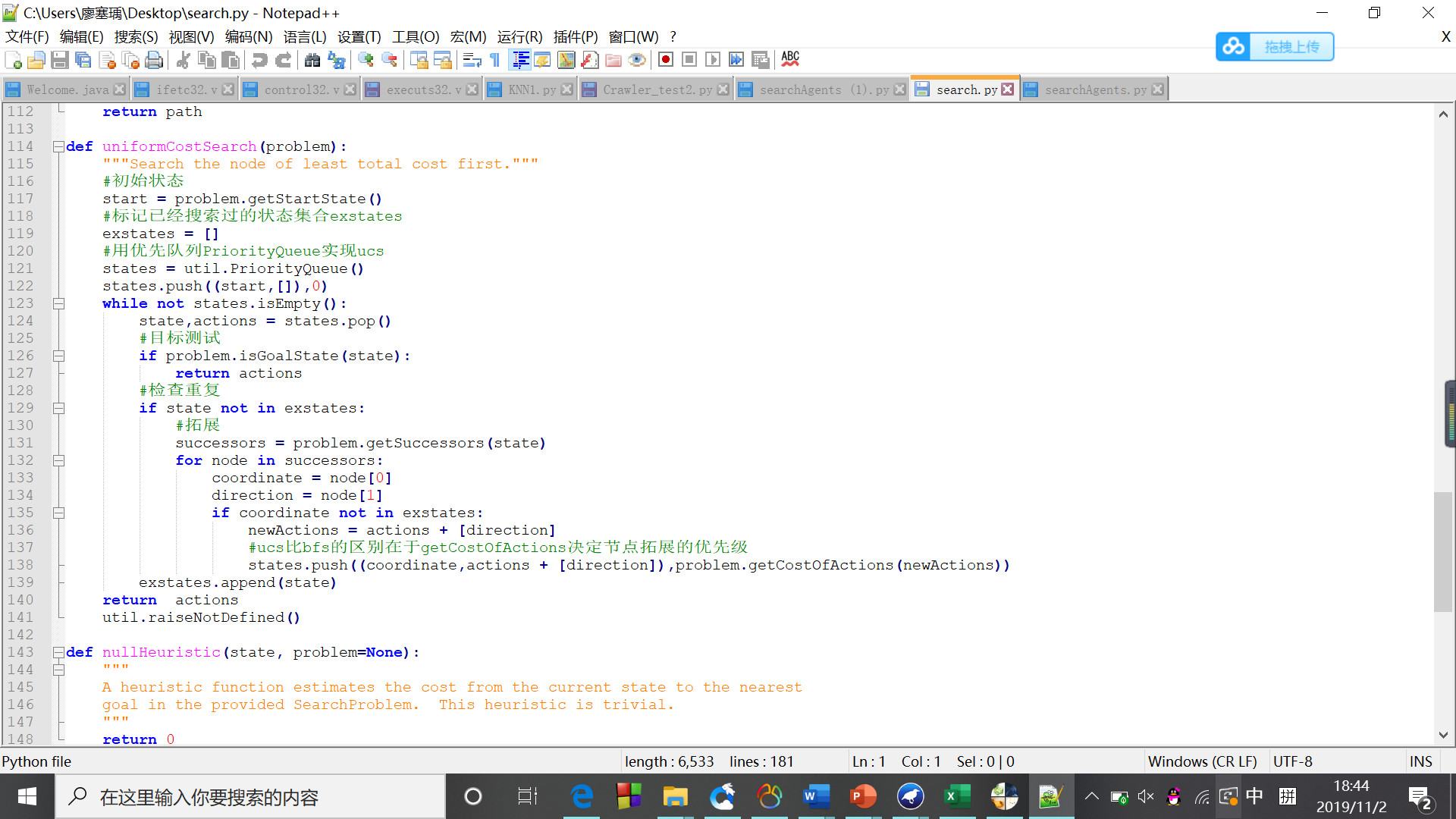
问题一：深度优先搜索



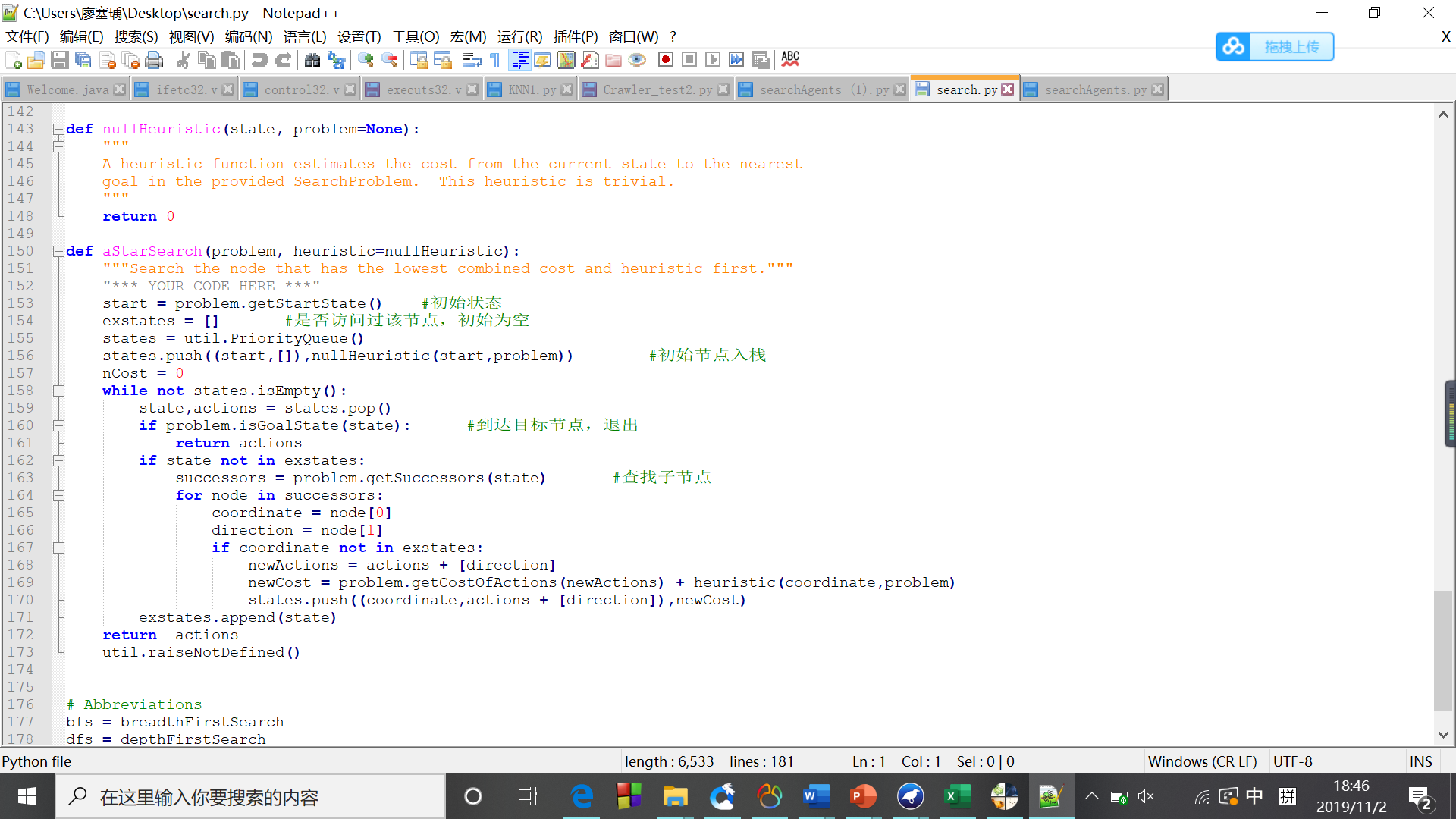
问题二：广度优先搜索



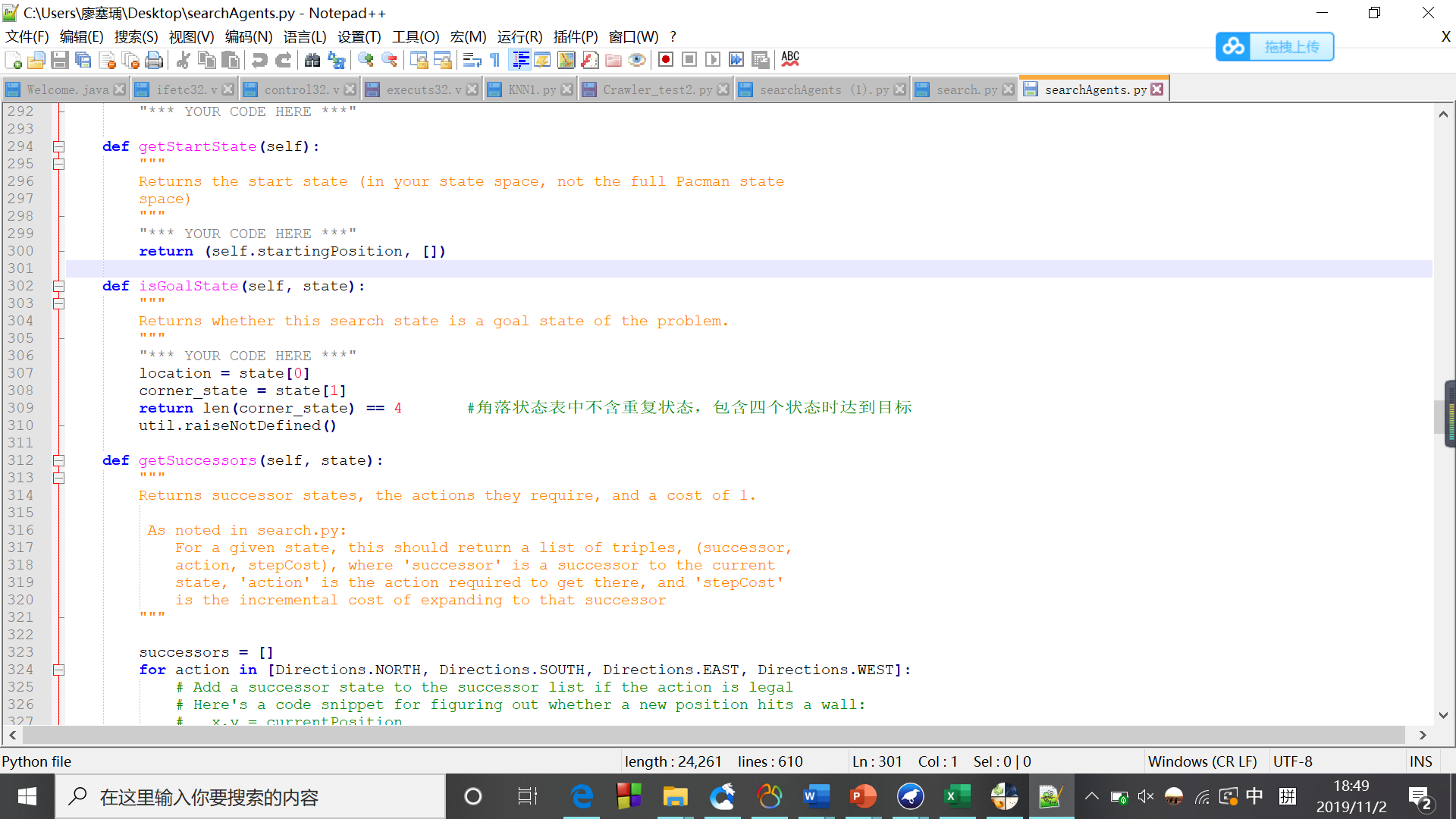
问题三：代价一致算法

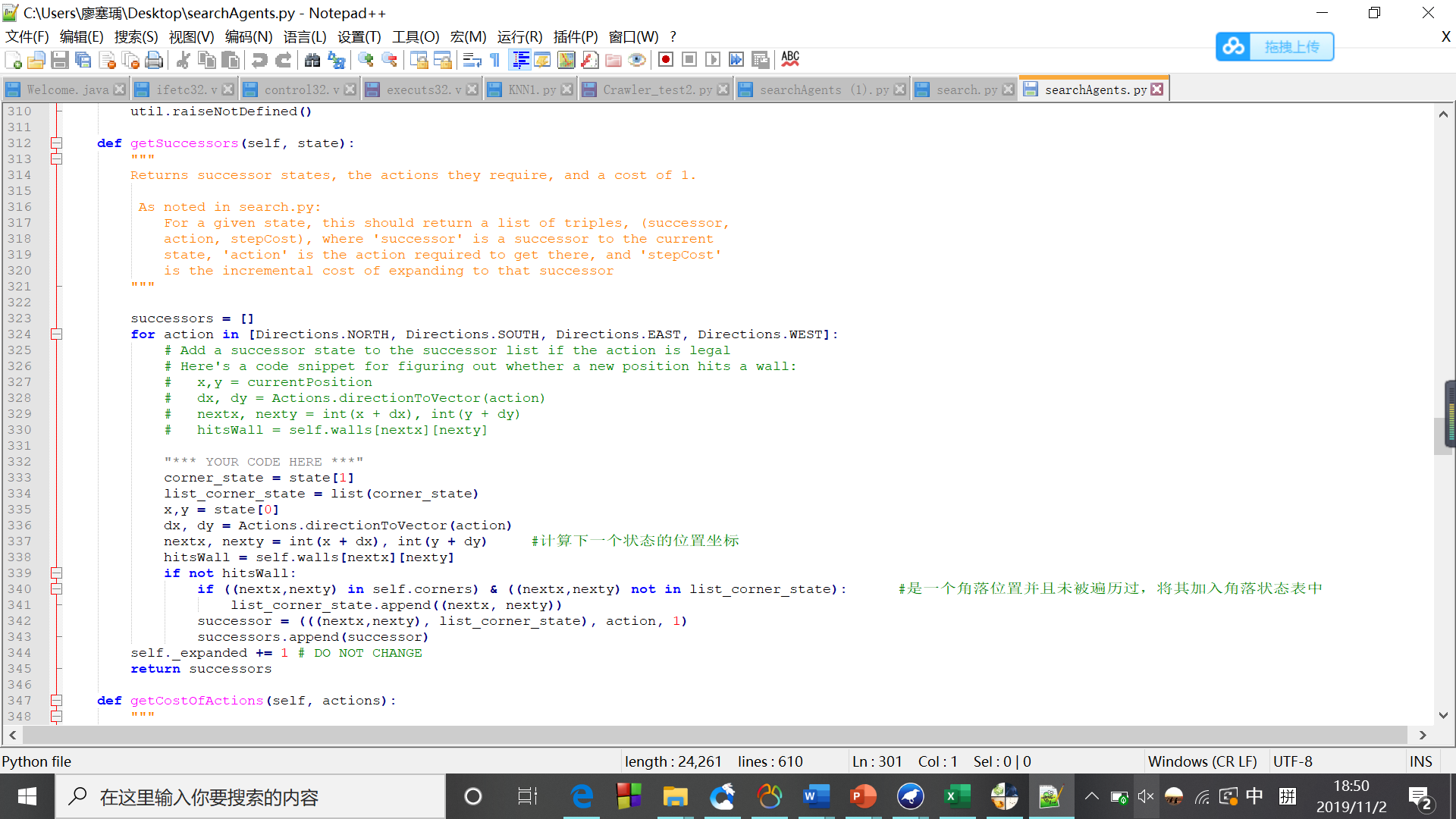


问题四：A\*算法

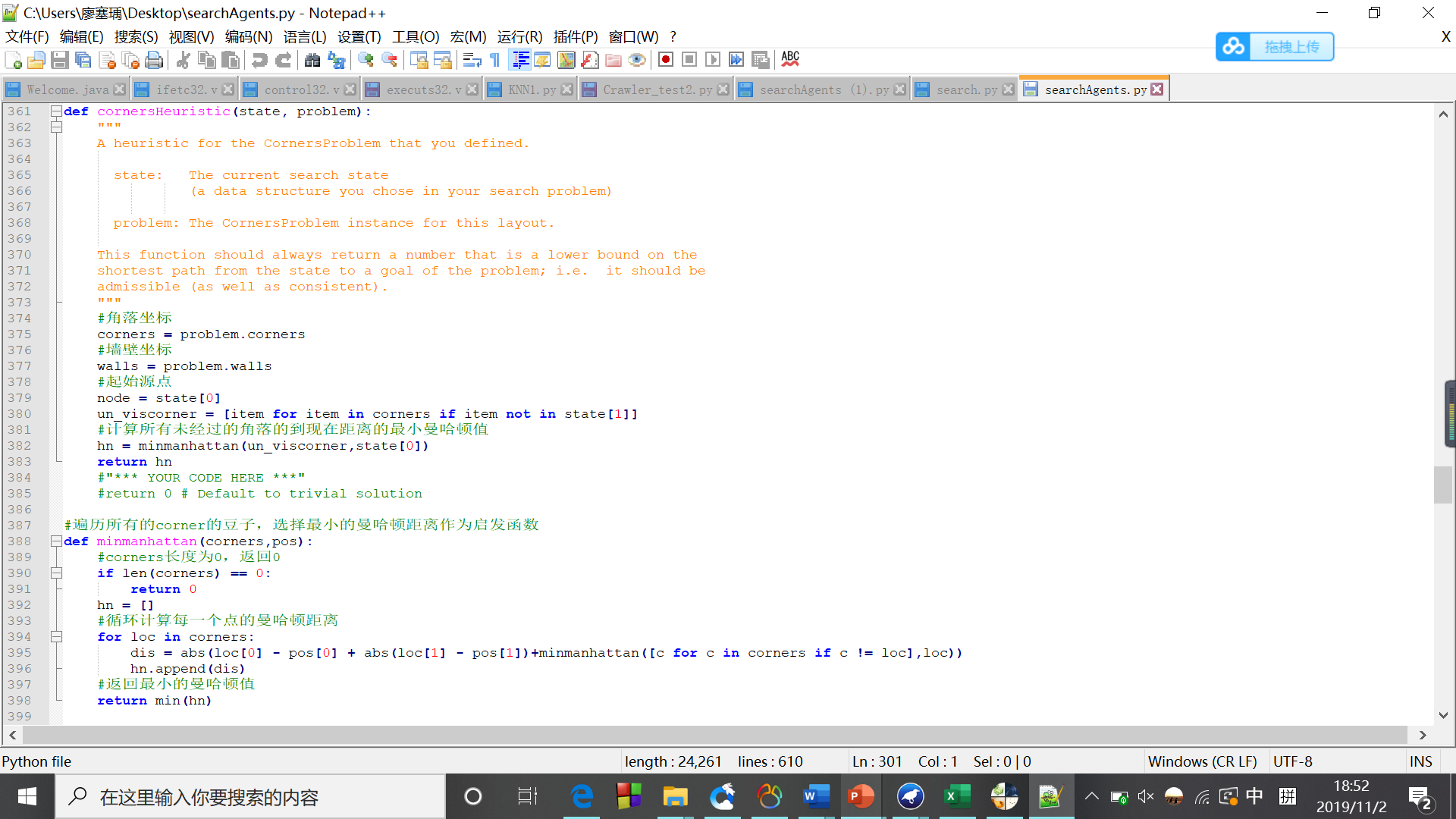


问题五：查找所有角落

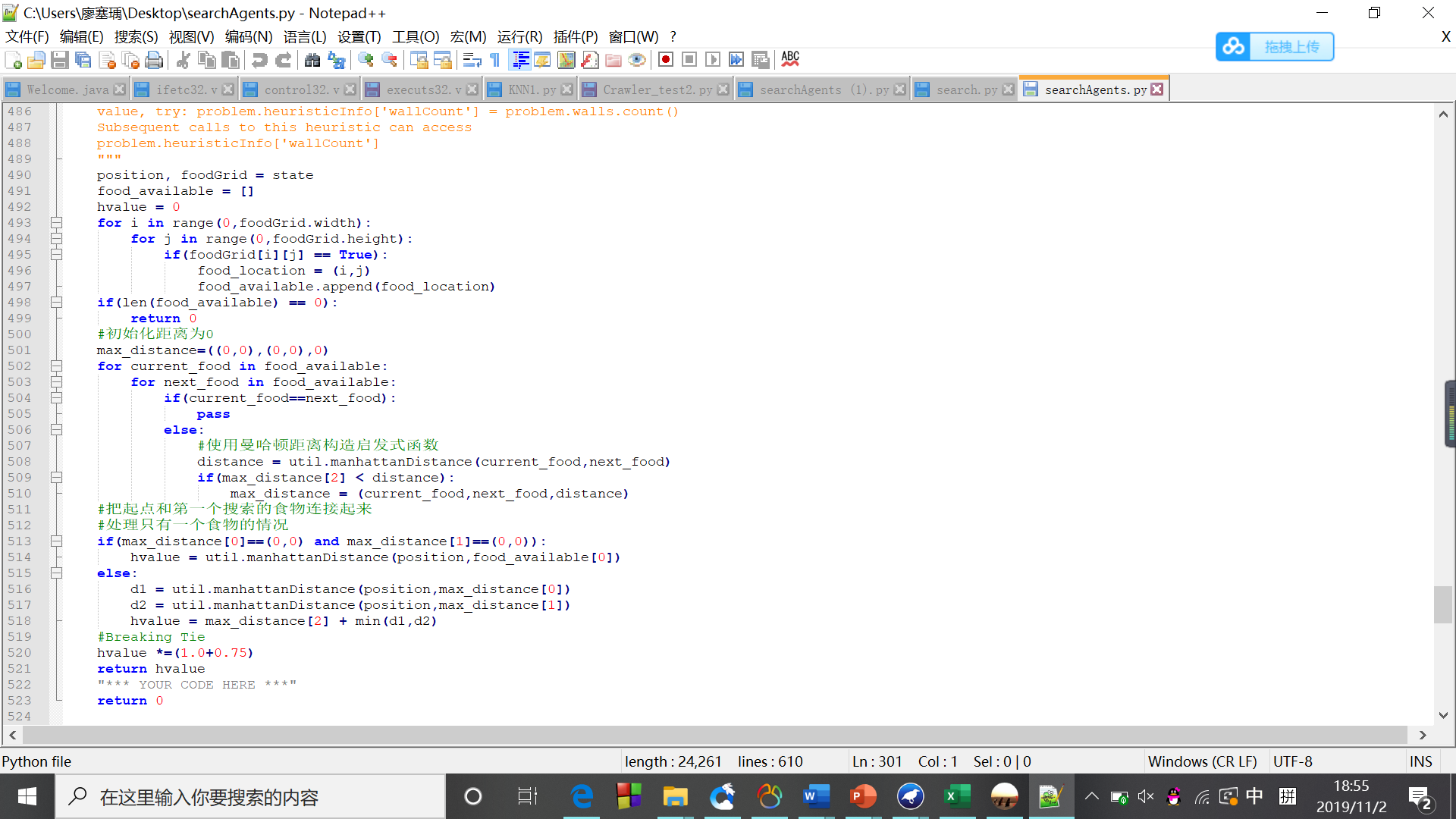




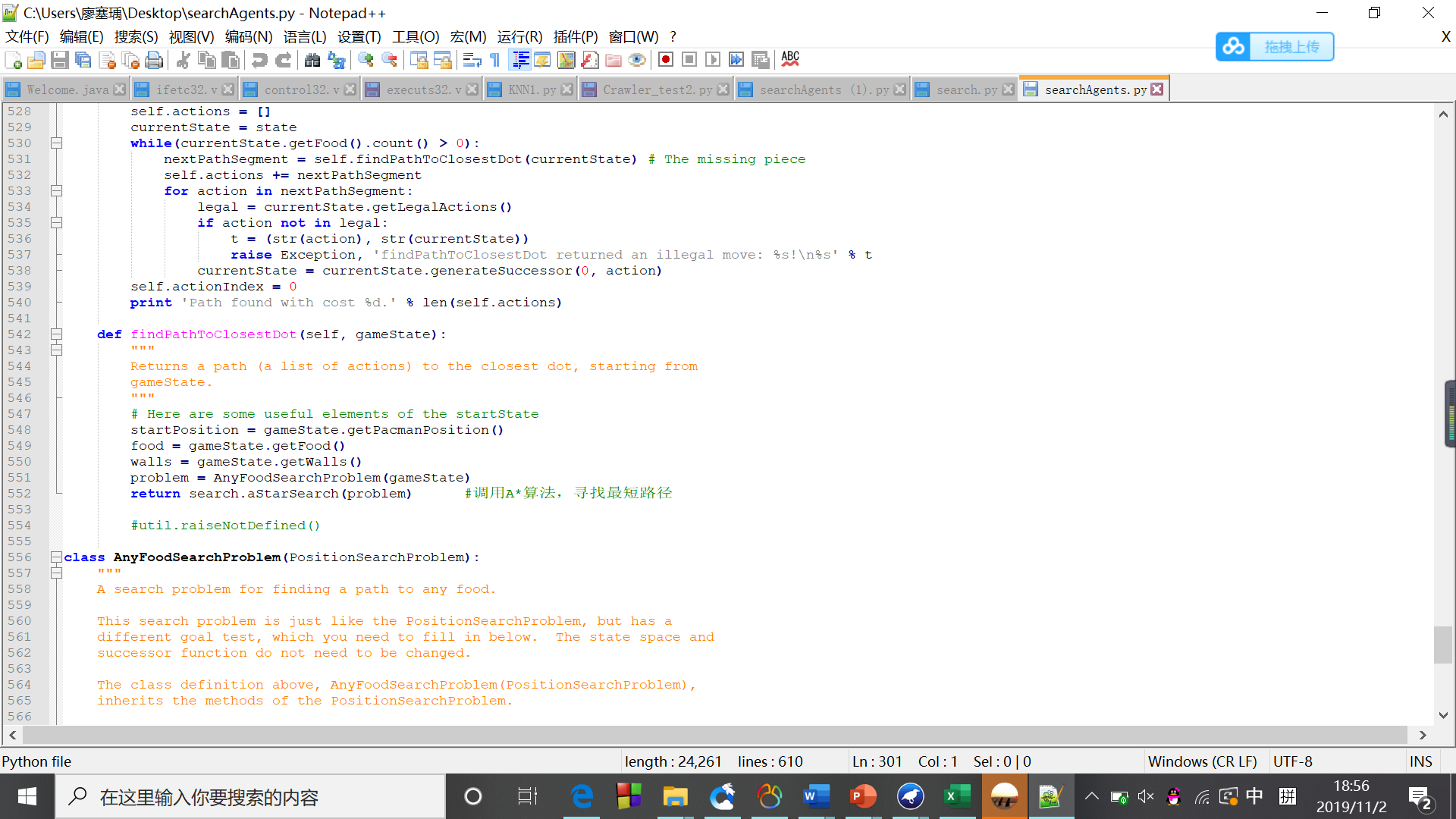
问题六：角落问题——启发式

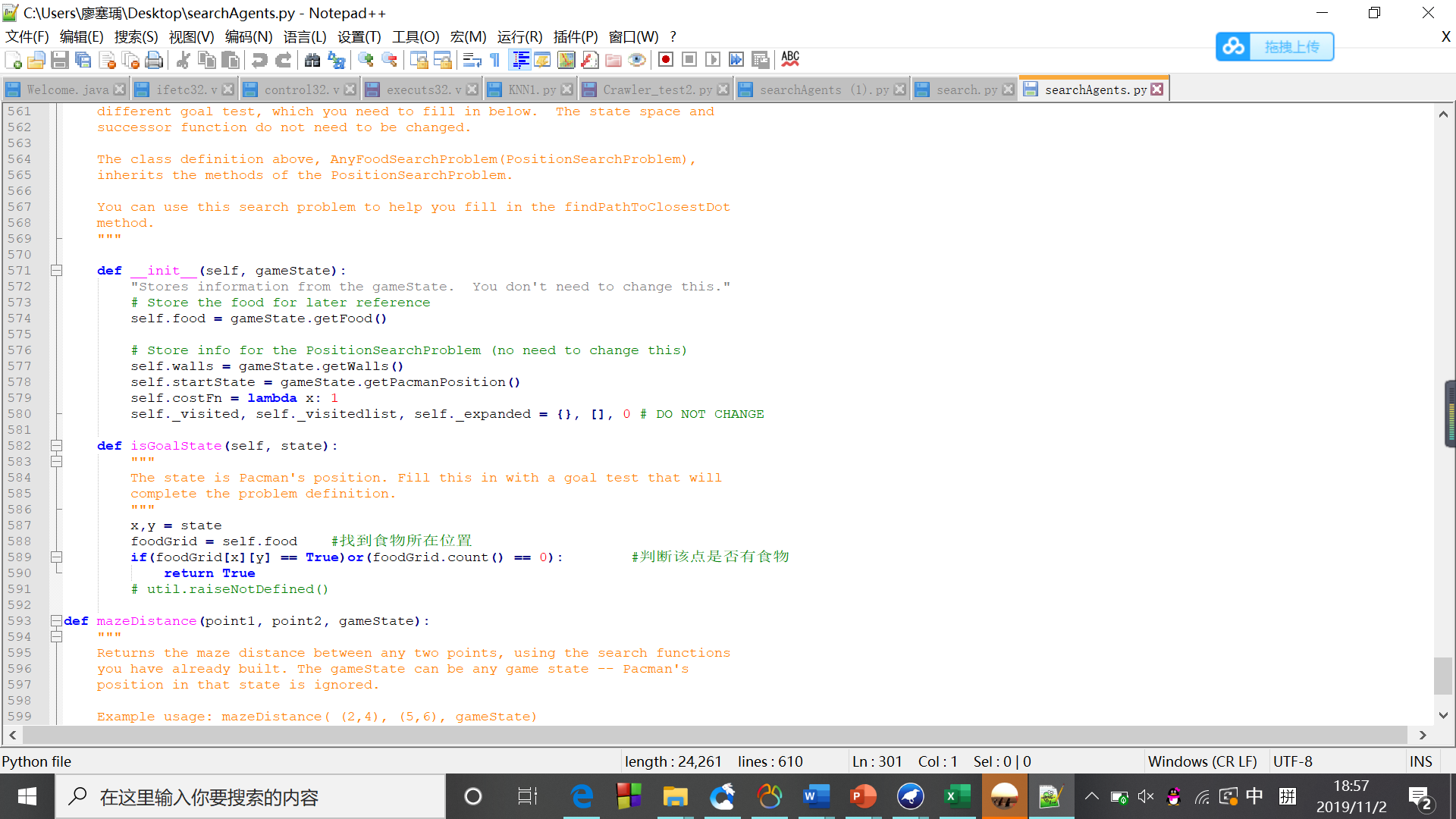


# 问题七：吃掉所有的“豆”



问题八：次优先搜索





**三．算法实现**

**3.1 实验环境与问题规模**

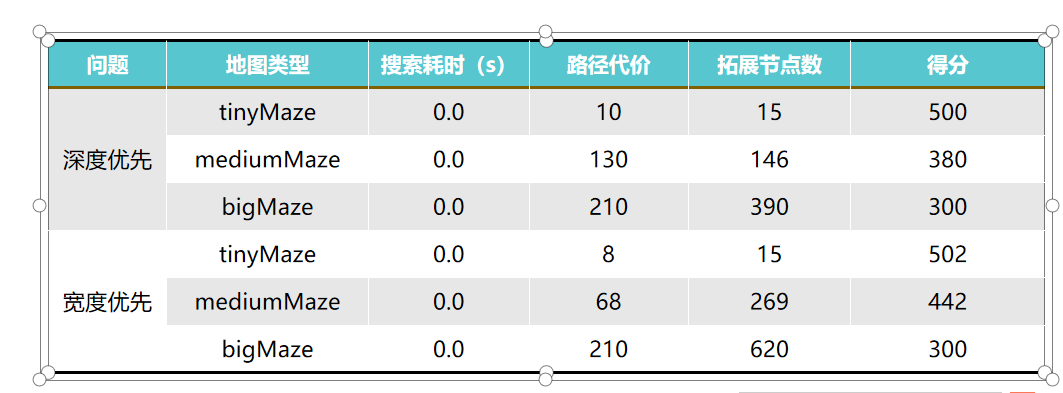
python2.7

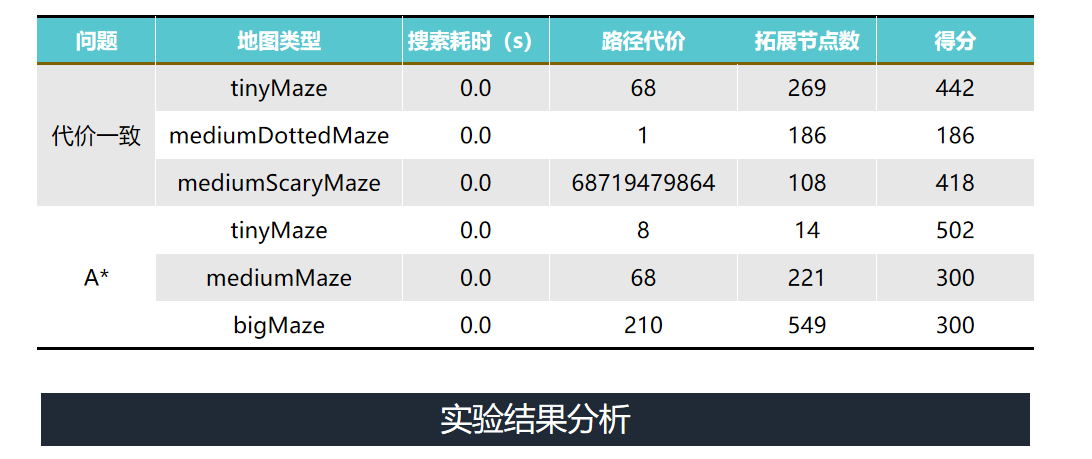
**3.2 数据结构**

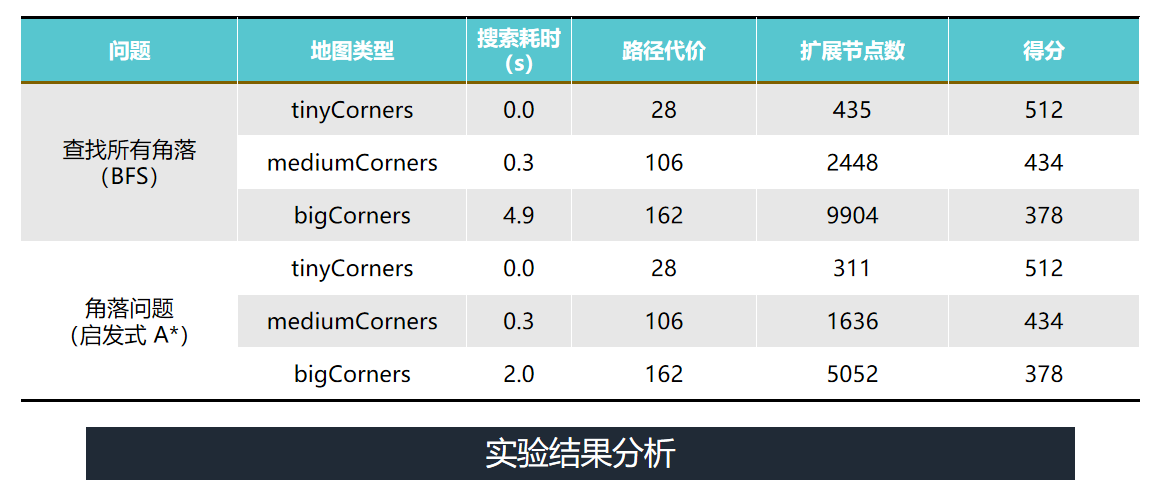
堆栈：用于类似深度优先搜索以及回溯

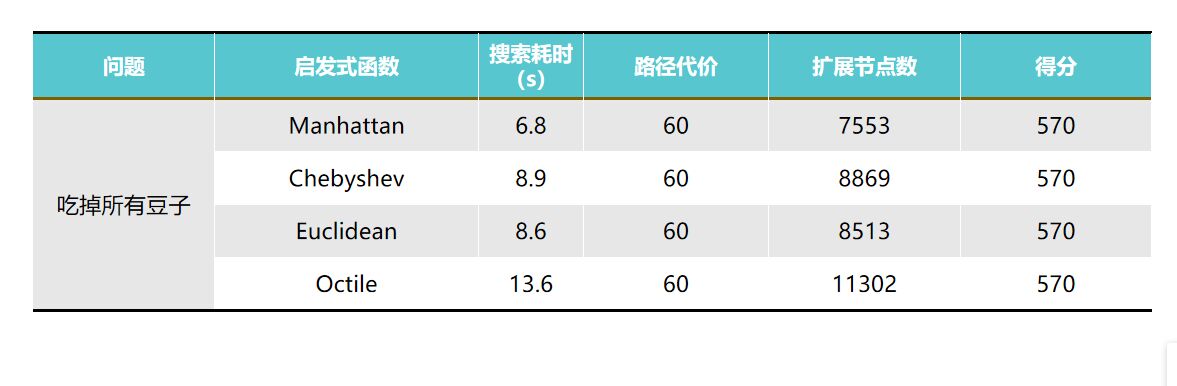
队列：用于类似广度优先搜索

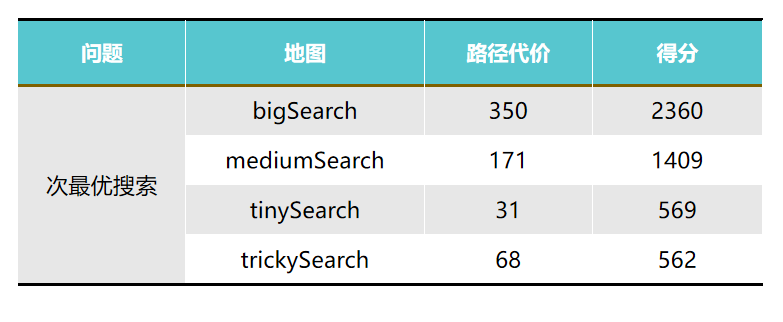
**3.3 实验结果**

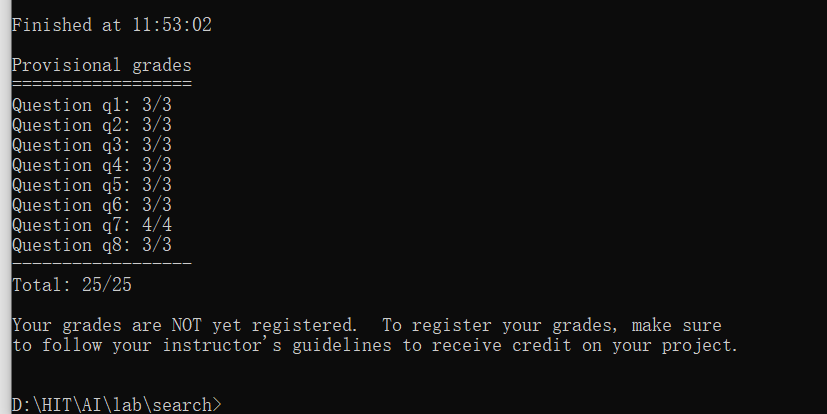
****

****

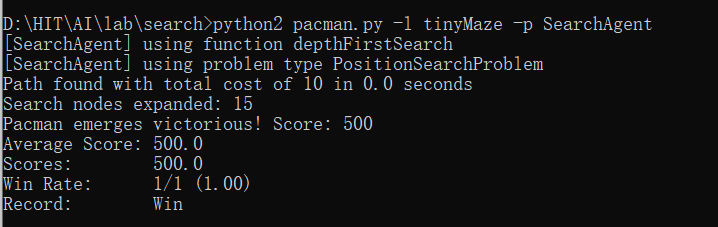
****

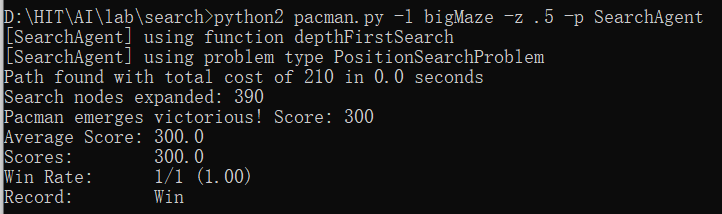
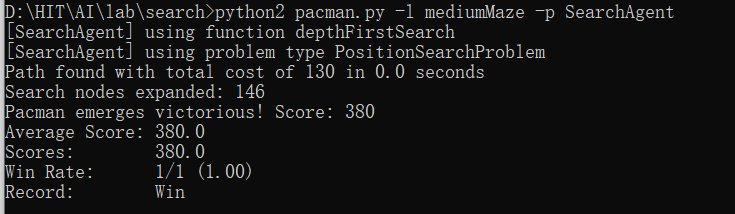
****

****

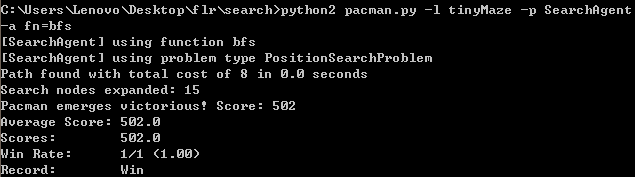


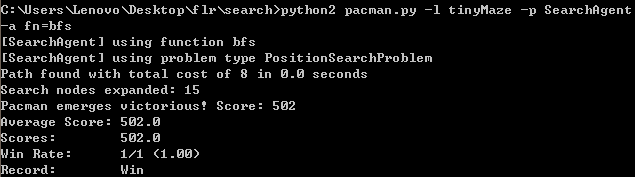
**3.4 系统中间及最终输出结果（要求有屏幕显示）**

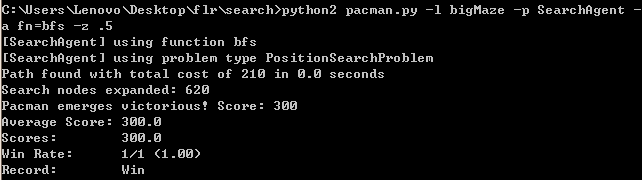
问题一：广度优先搜索——三种地图分别的测试结果



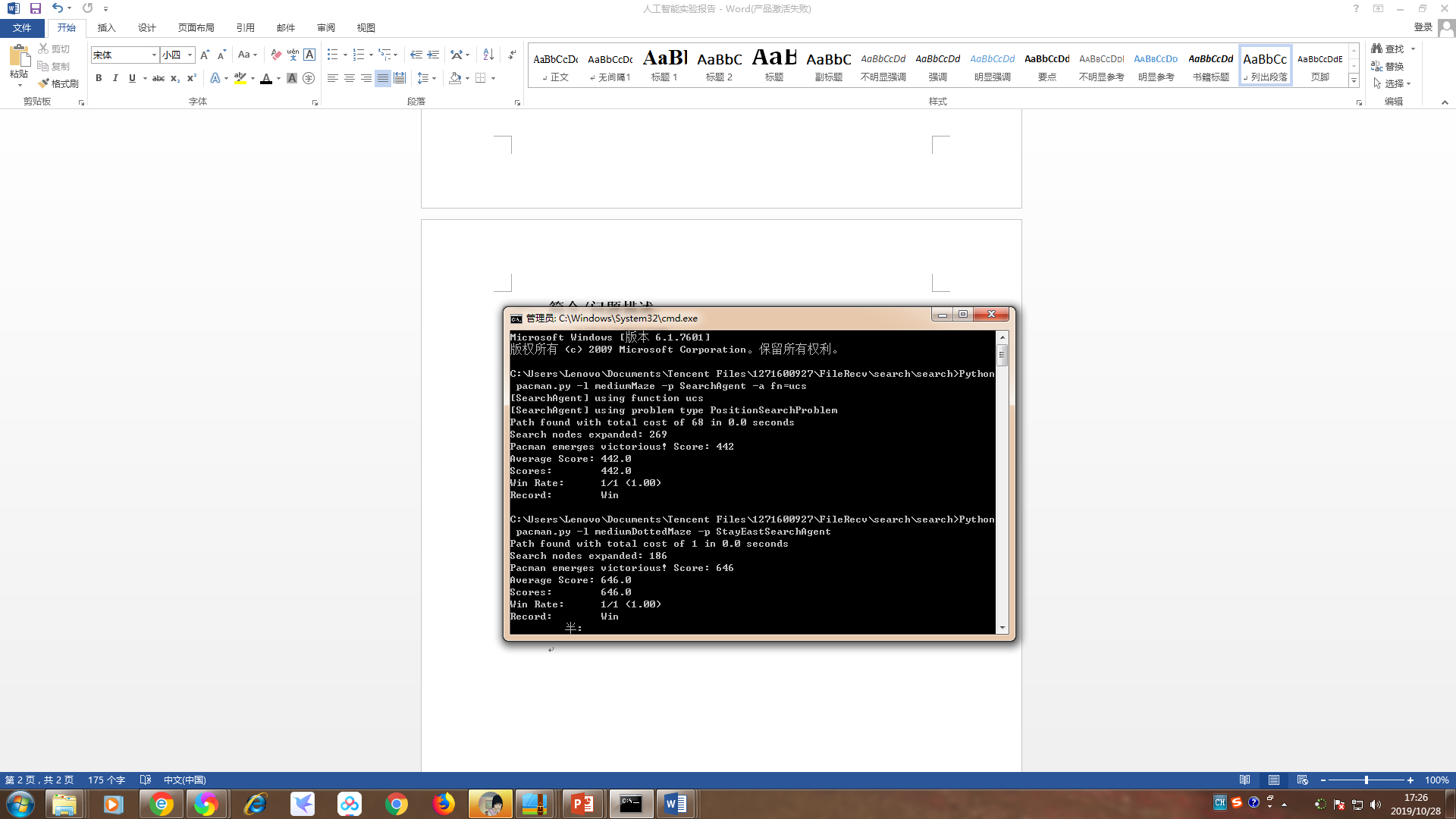
问题二：宽度优先搜索——三种地图分别的测试结果

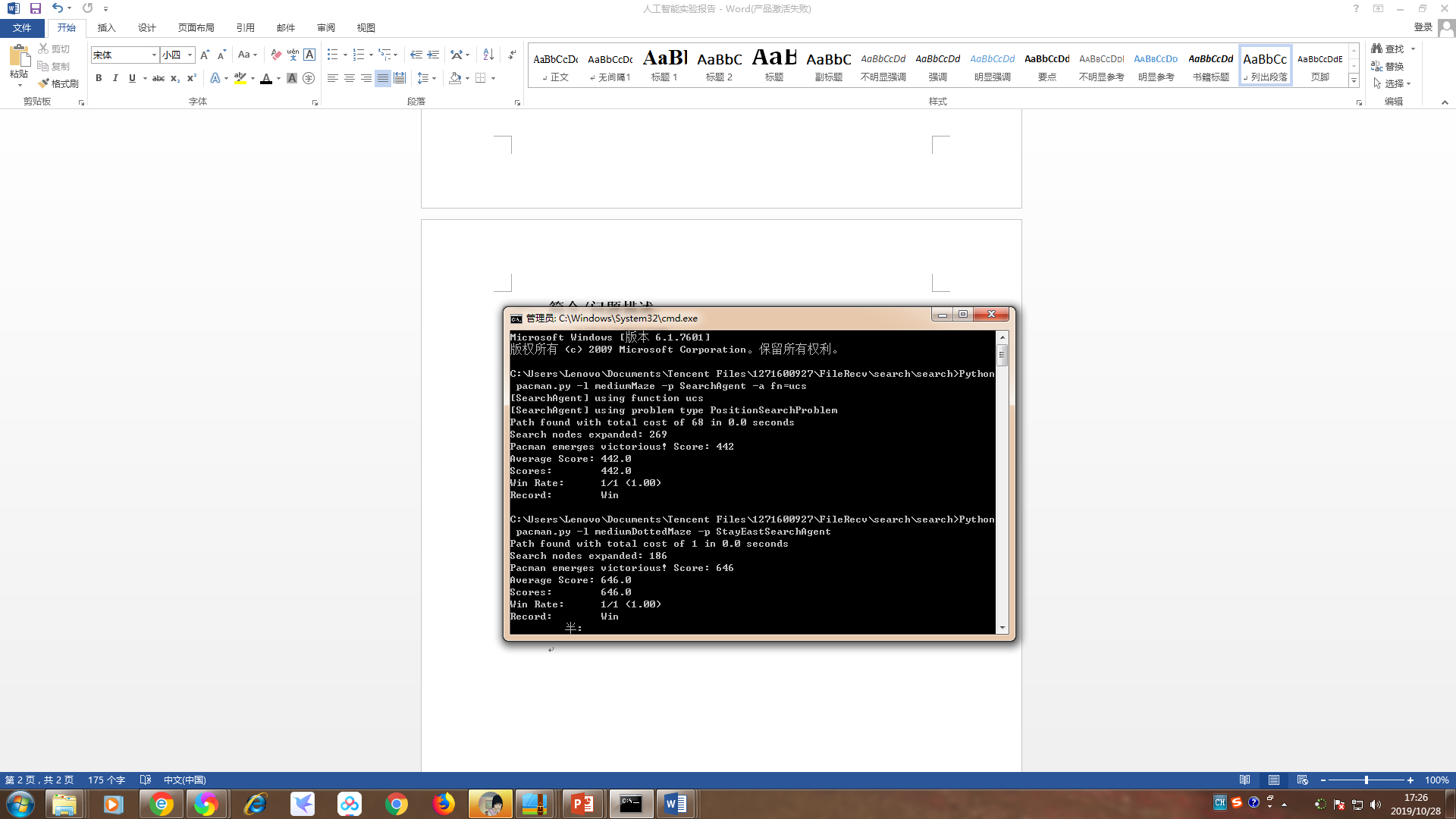


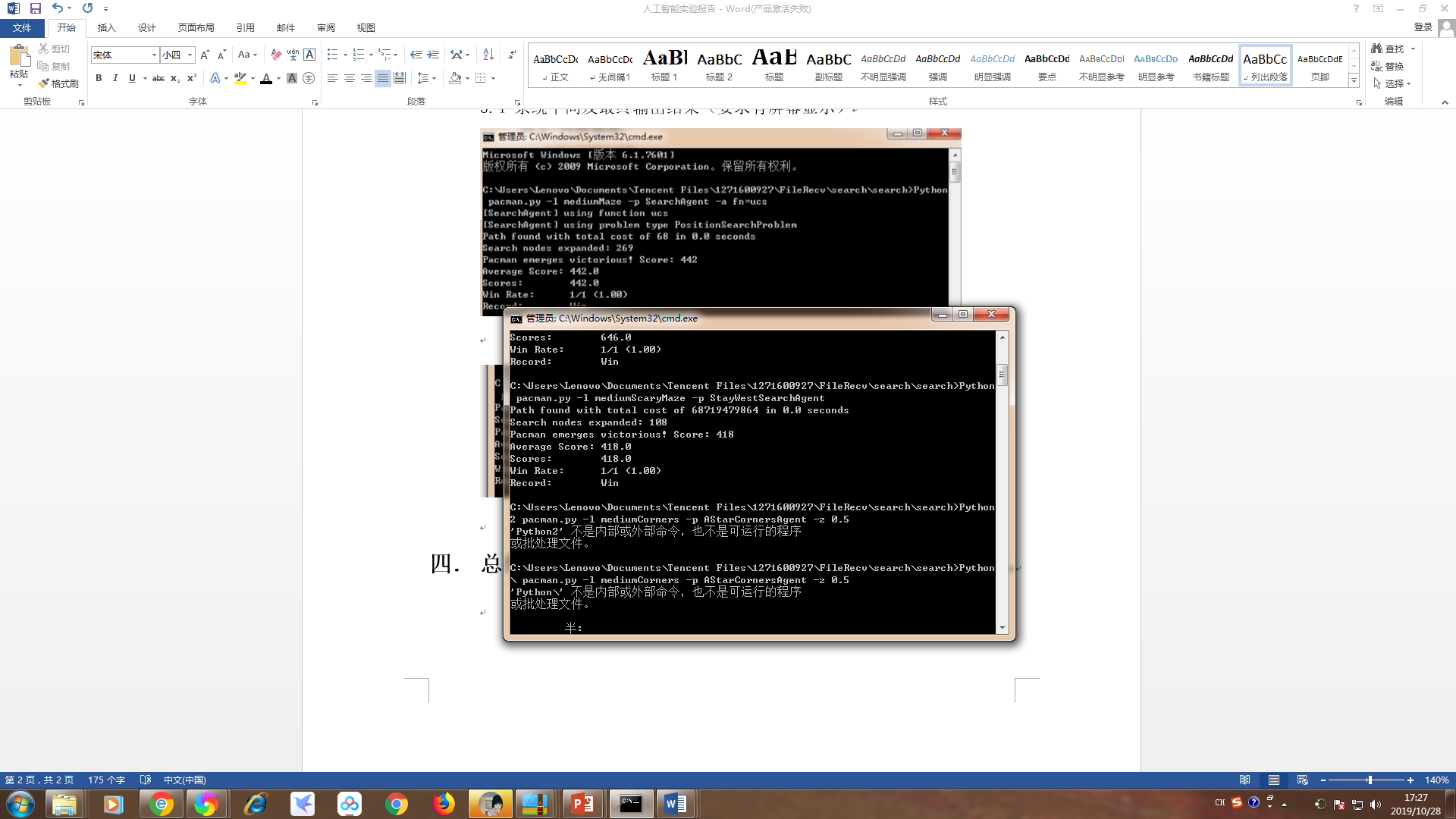


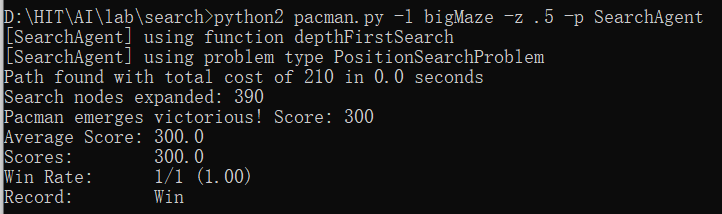


问题三：代价一致算法——三种地图分别的测试结果

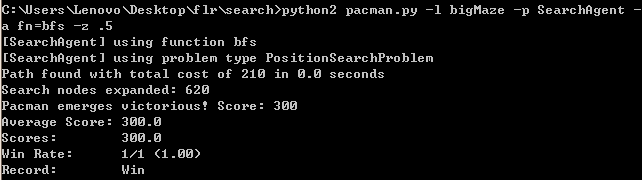


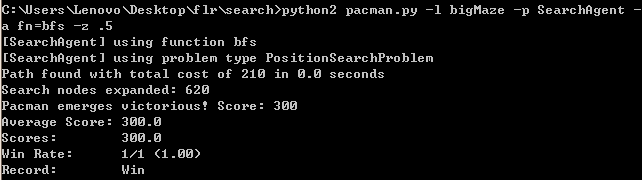


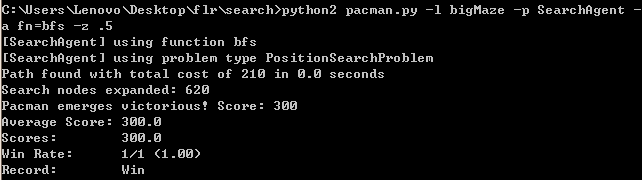


问题四：A\*算法

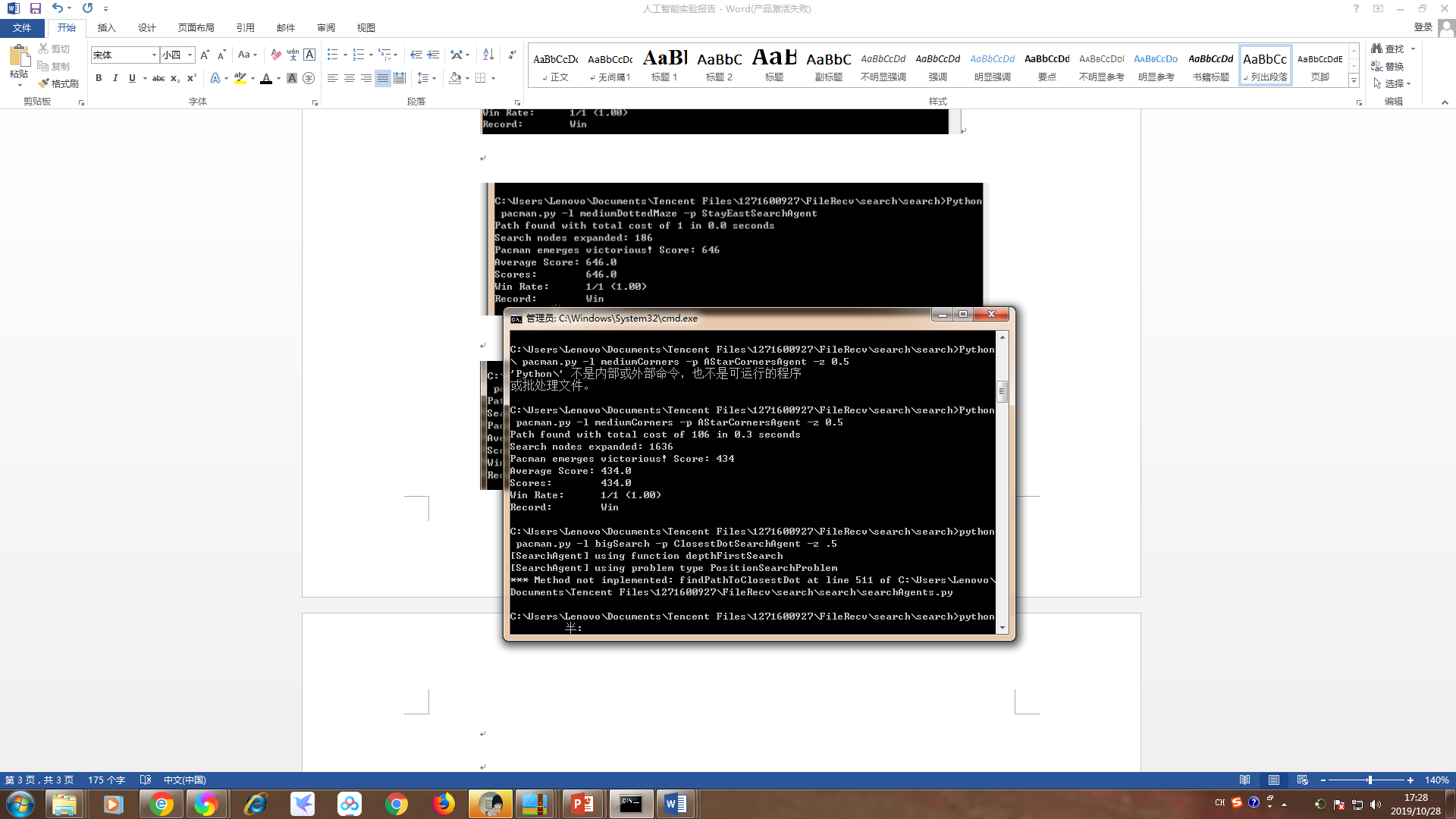
问题五：找到所有的角落——三种地图分别的测试结果

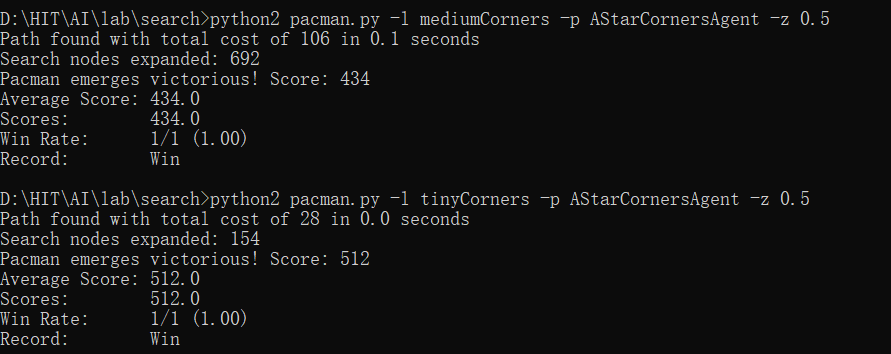


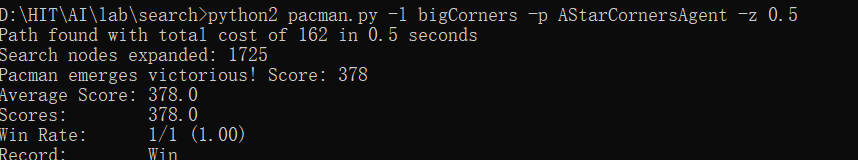


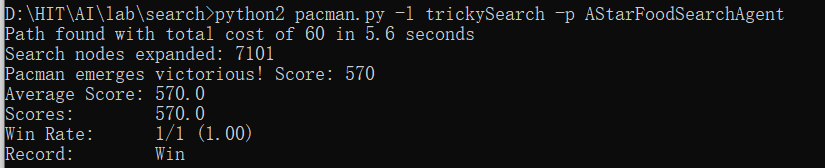


问题六：角落问题——启发式

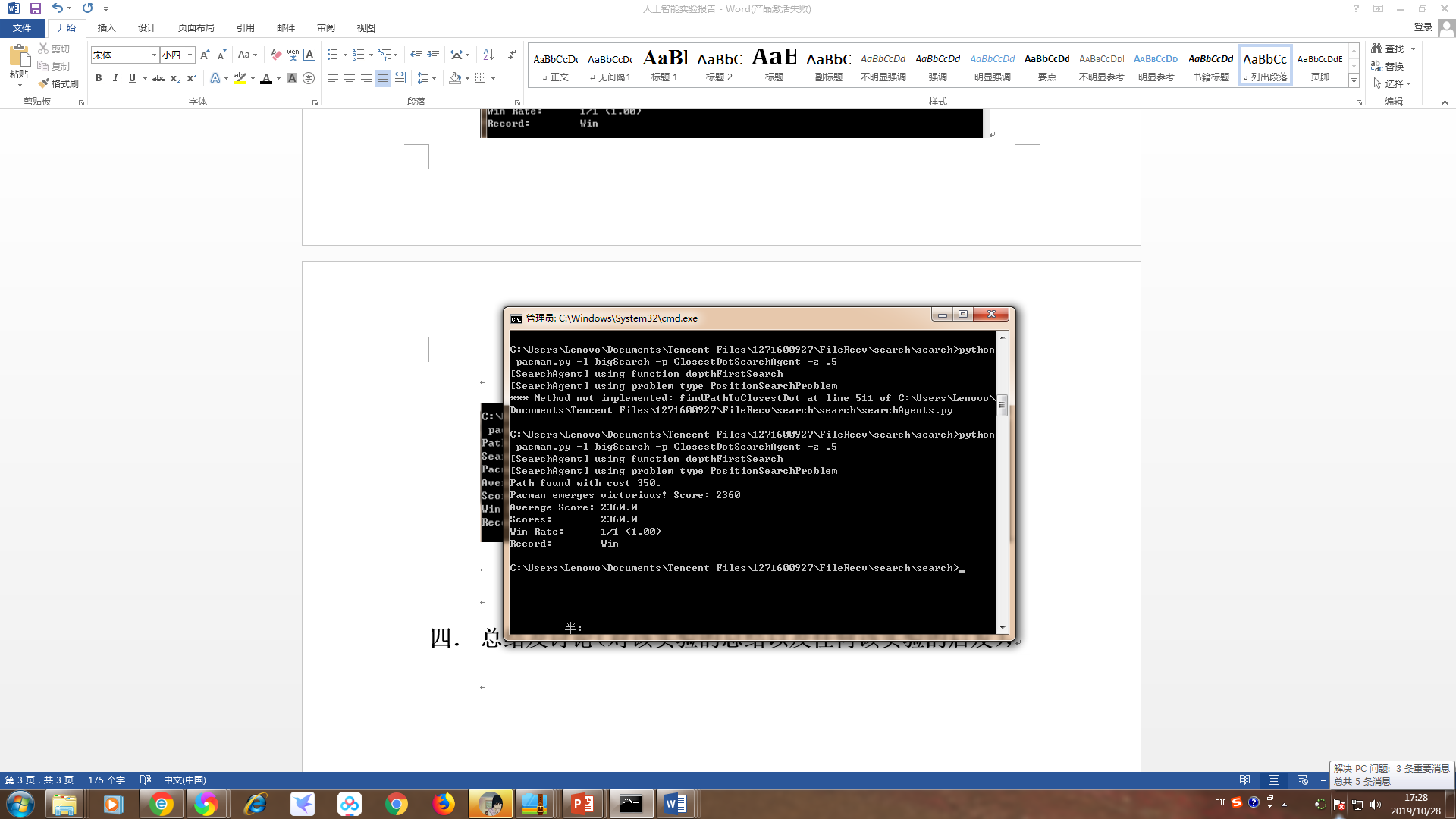






问题七：吃掉所有的“豆”

问题八：次优先搜索



**四．总结及讨论（对该实验的总结以及任何该实验的启发）**

通过这次实验，了解到了搜索的基本方式是广度优先以及深度优先，对于不同的情况两者不同的搜索路径和搜索方式，会影响时间得长短，比如当目标就在出发点周围时，深度优先搜索选择了一条错误的路，会浪费很多时间，而对于广度优先搜索，会立即搜索周围所有的结点，容易发现目标。而对于远离出发点的点，使用广度优先搜索会耗费非常多的时间，深度则耗费较少时间。在这两种基础的搜索上，使用代价函数以及估值，进行优化，尽量避免上述两种比较极端的情况出现，并且选择的代价函数不同，以及优化方法不同，开销会不同，代价也会不同。选择合适的代价估计函数能优化搜索，在实际实验中，需要考虑各种方面的因素，不仅仅只看最终目标，求最优路线，要结合时间空间代价，综合考虑，选择搜索方式。