**《人工智能》**

**实验报告**

**实验名称： 搜索策略**

**成 员： 张文韬 胡 聪 李瑞吉 李乐航**

**完 成： 胡 聪**

**学年学期： 2020年秋季学期**

**任课教师： 苏敬勇**

哈尔滨工业大学（深圳）

计算机科学与技术学院

**目录**

1. 简介/问题描述
   1. 待解决问题的解释
   2. 问题的形式化描述
   3. 解决方案介绍（原理）
2. 算法介绍

2.1 所用方法的一般介绍

2.2 算法伪代码

1. 算法实现

3.1 实验环境与问题规模

3.2 数据结构

3.3 实验结果

3.4 系统中间及最终输出结果

1. 总结及讨论

参考文献

附录—源代码及其注释

1. **简介/问题描述**
   1. 待解决问题的解释

问题3：代价一致算法

很多情况下，路径中的代价是可以改变的。完成代价一致搜索方法(search.py文件中的uniformCostSearch函数)，并用命令测试你的code

问题8：次最优搜索

定义一个优先吃最近的豆子函数是提高搜索速度的一个好的办法。补充完成searchAgents.py文件中的AnyFoodSearchProblem目标测试函数，并完成searchAgents.py文件中的ClosestDotSearchAgent部分，在此Agent当中缺少一个关键的函数：找到最近豆子的函数。用命令测试你的code

1.2问题的形式化描述

UNIFORM-COST-SEARCH(problem) returns a solution,or failure

* 1. 解决方案介绍（原理）

（3）一致代价搜索是在广度优先搜索上进行扩展的，也被成为代价一致搜索，他的基本原理是：一致代价搜索总是扩展路径消耗最小的节点N。N点的路径消耗等于前一节点N-1的路径消耗加上N-1到N节点的路径消耗。代价一致搜索其实就是贪心搜索，取代扩展深度最浅的节点，代价一致搜索扩展的是路径消耗最低的节点n。如果所有单步耗散都相等，这种算法就和广度优先搜索算法一样。在uCS函数中，计算每条路径的总代价，将总代价作为优先级进行搜索，待搜索序列存储于队列中。对于每个节点，使用代价函数计算所产生的的代价，并且依次作为搜索的优先级进行搜索，同时对于每个节点添加是否被访问的标记。

（8）次最优搜索，定义一个优先吃最近豆子的函数，以此提高搜索速度。

1. **算法介绍**

2.1 所用方法的一般介绍

frontier：边缘。存储未扩展的节点。优先级队列，按路径消耗来排列

explored：探索集。存储的是状态

1.如果边缘为空，则返回失败。操作：EMPTY?(frontier)

2.否则从边缘中选择一个叶子节点。操作：POP(frontier)

3.目标测试：通过返回，否则将叶子节点的状态放在探索集

4.遍历叶子节点的所有动作

每个动作产生子节点，如果子节点的状态不在探索集或者边缘，则插入到边缘集合。操作：INSERT(child, frontier)，否则如果边缘集合中如果存在此状态且有更高的路径消耗，则用子节点替代边缘集合中的状态

2.2 算法伪代码

funtion UNIFORM-COST-SEARCH(problem) returns a solution,or failure

    node←a with STATE = problem.INITIAL-STATE,PATH-COST = 0

    frontier←a priority queue ordered by PATH-COST, with node as the only element

    explored← an empty set

    loop do

        if EMPTY?(frontier) then return failure

        node←POP(frontier)

        if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)

        add node.STATE to explored

        for each action in problem.ACTIONS(node.STATE) do

        if child.STATE is not in explored or frontier then

            frontier ← INSERT(child,frontier)

        else if child.STATE is in frontier with higher PATH-COST then

            replace that frontier node with child

1. **算法实现**

3.1 实验环境与问题规模

实验环境：

操作系统：Windows 10 Pro X64 Edition 版本号2004

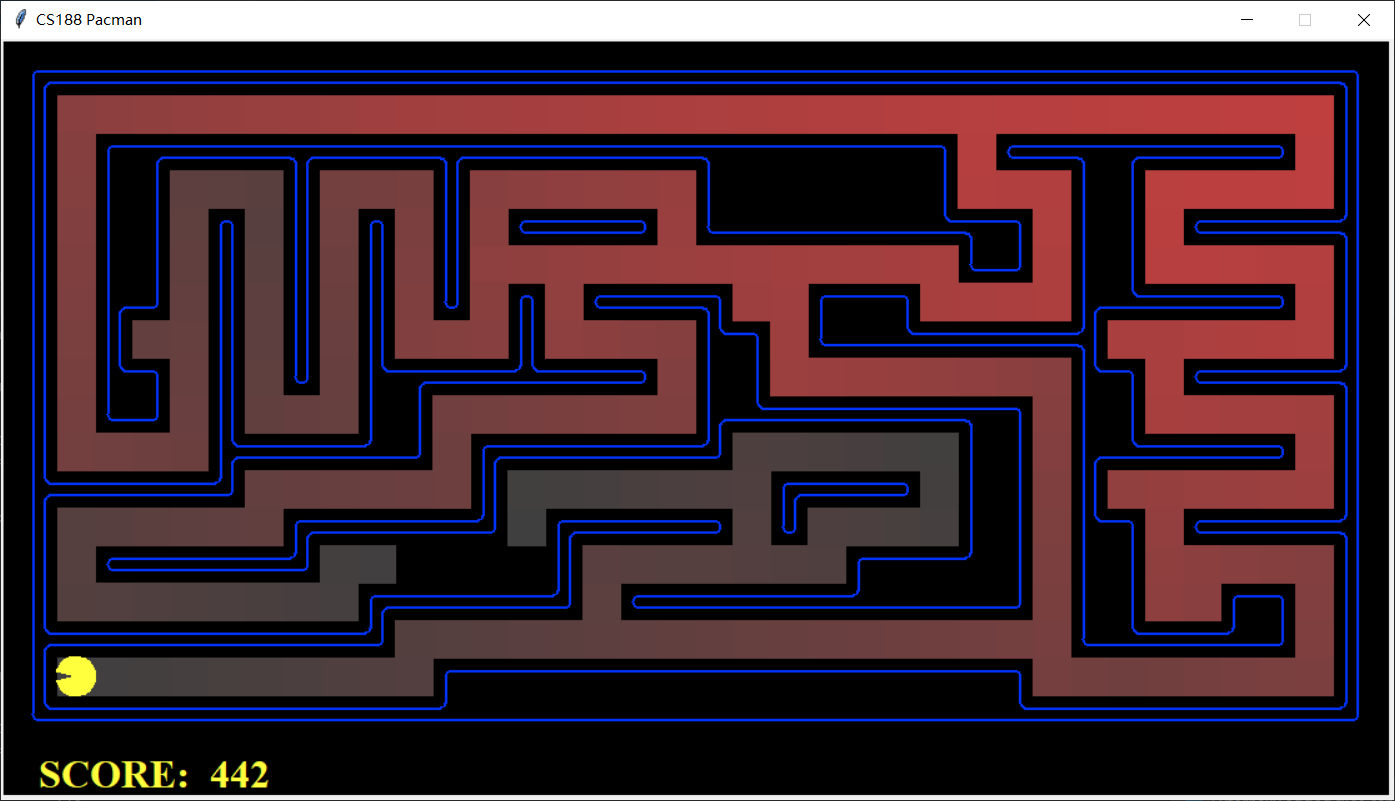
开发环境：PyCharm 2020.2.3 (Community Edition)

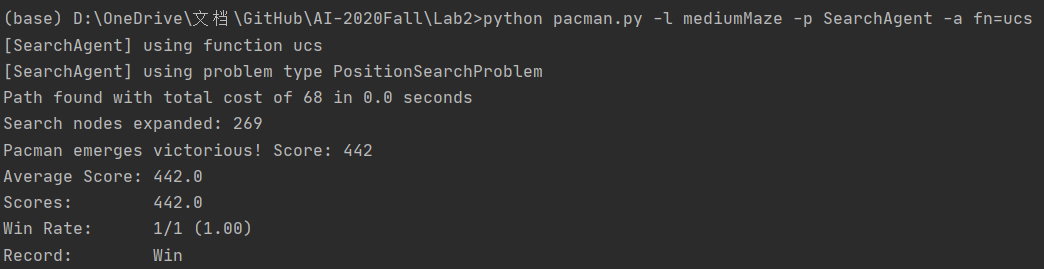
3.2 数据结构

util.PriorityQueue()

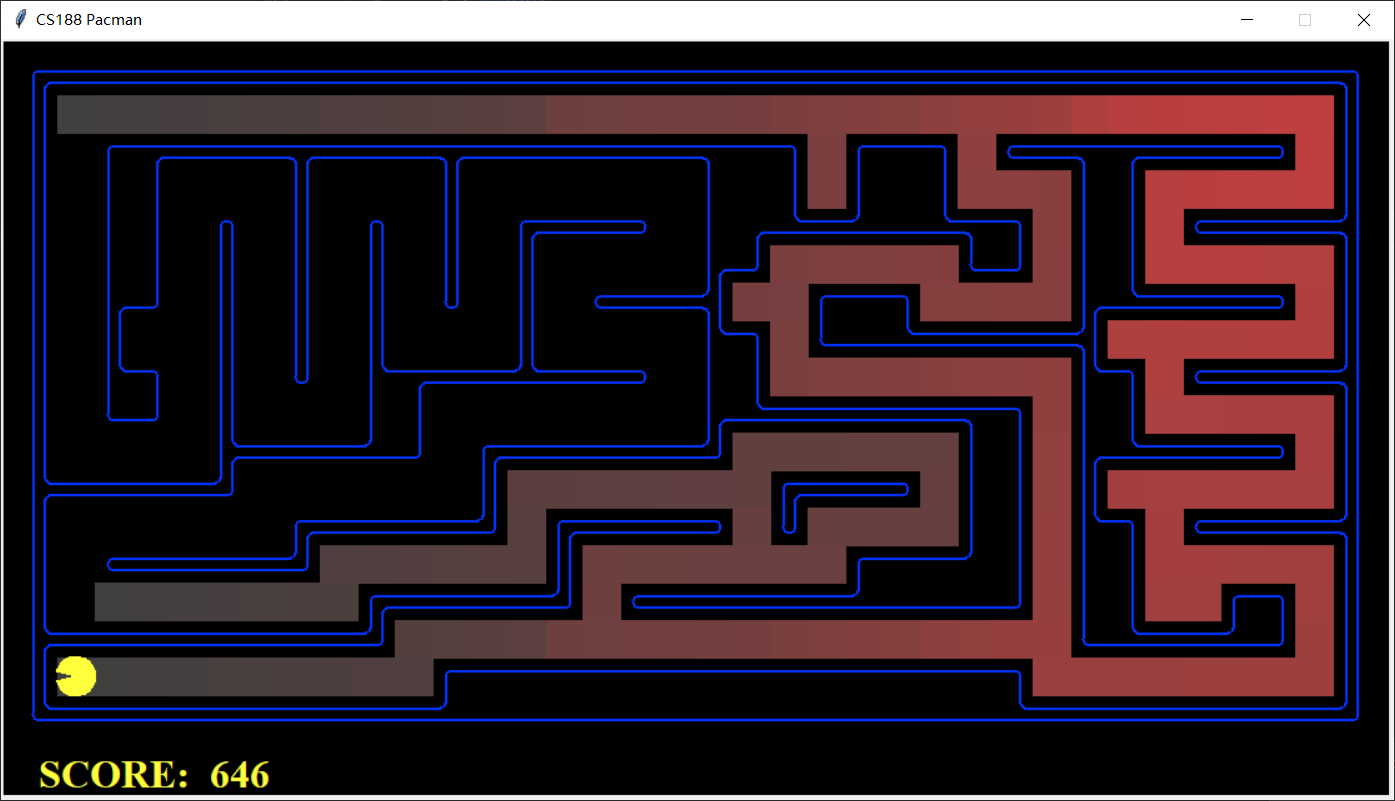
3.3 实验结果

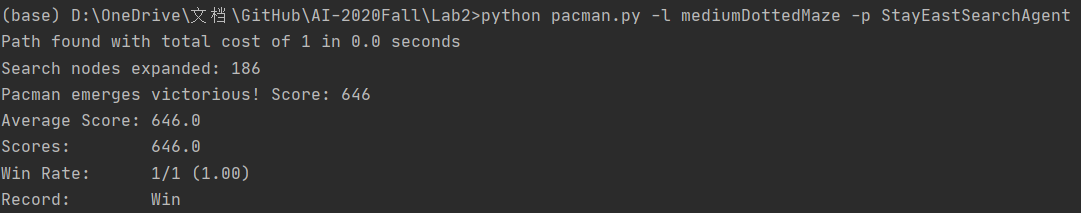
（3）python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs



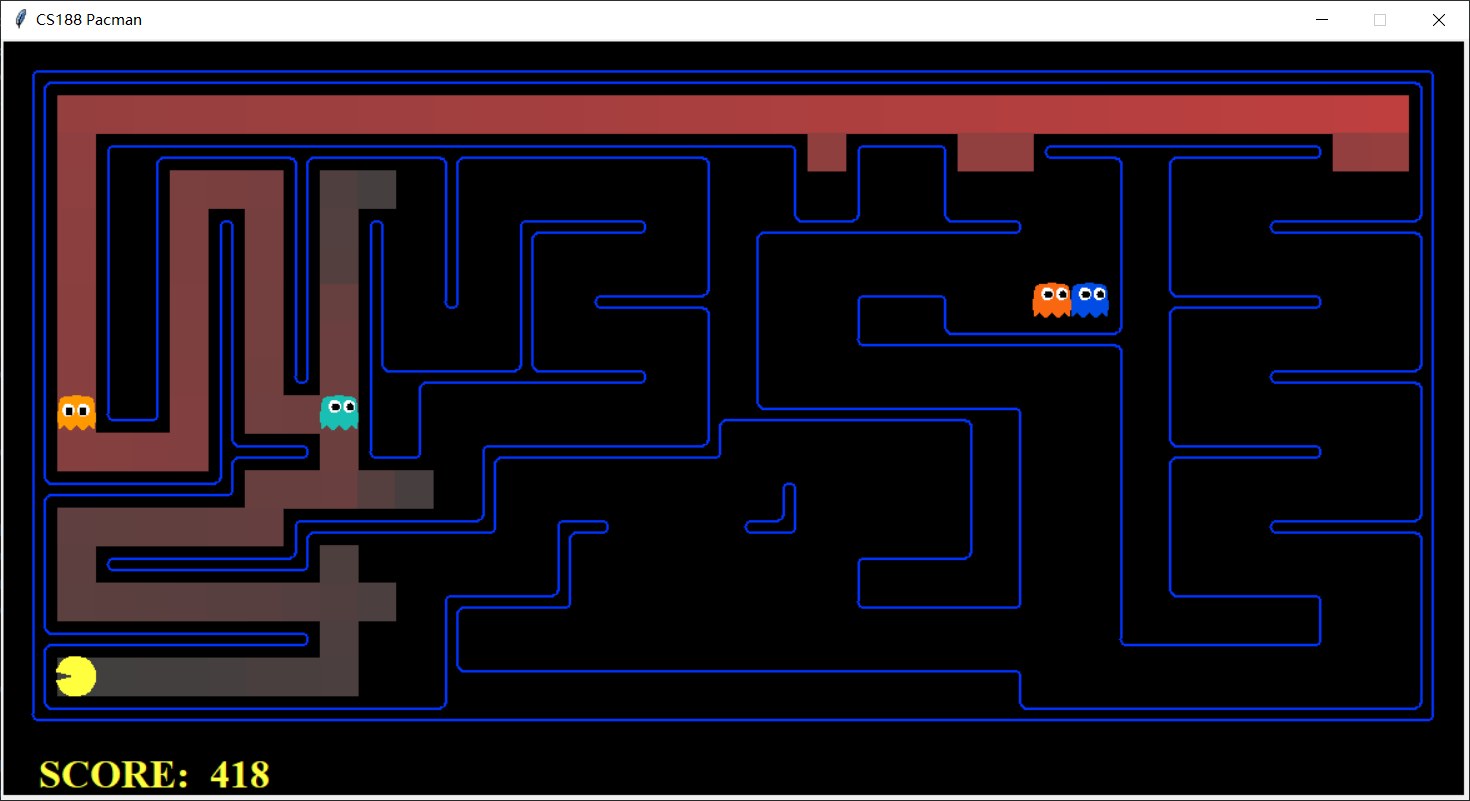


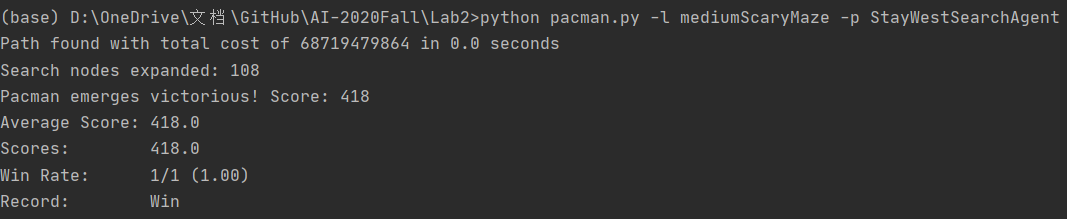
python pacman.py -l mediumDottedMaze -p StayEastSearchAgent





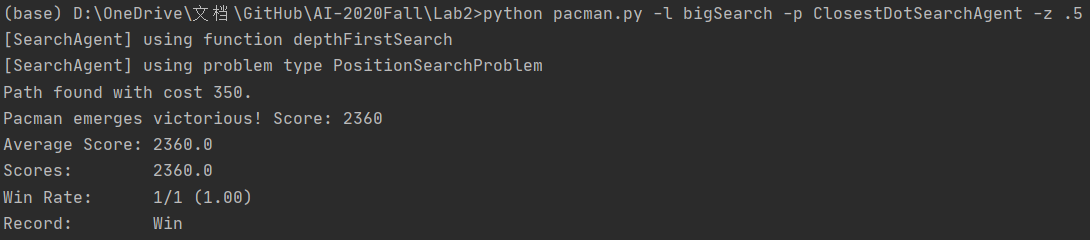
python pacman.py -l mediumScaryMaze -p StayWestSearchAgent





（8）python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5





1. **总结和讨论**

本次实验中对深度优先搜索、广度优先搜索、代价一致搜索和A\*算法等搜索方法使用Python进行了实现，加深了对于这四种方法的进一步了解。代价一致搜索效率较高，而A\*最为有效，能够节省搜索路径，提高效率。

**参考文献**

无

**附录—源代码及其注释**