## 人工智能实验一实验报告

PB19151769 马宇骁

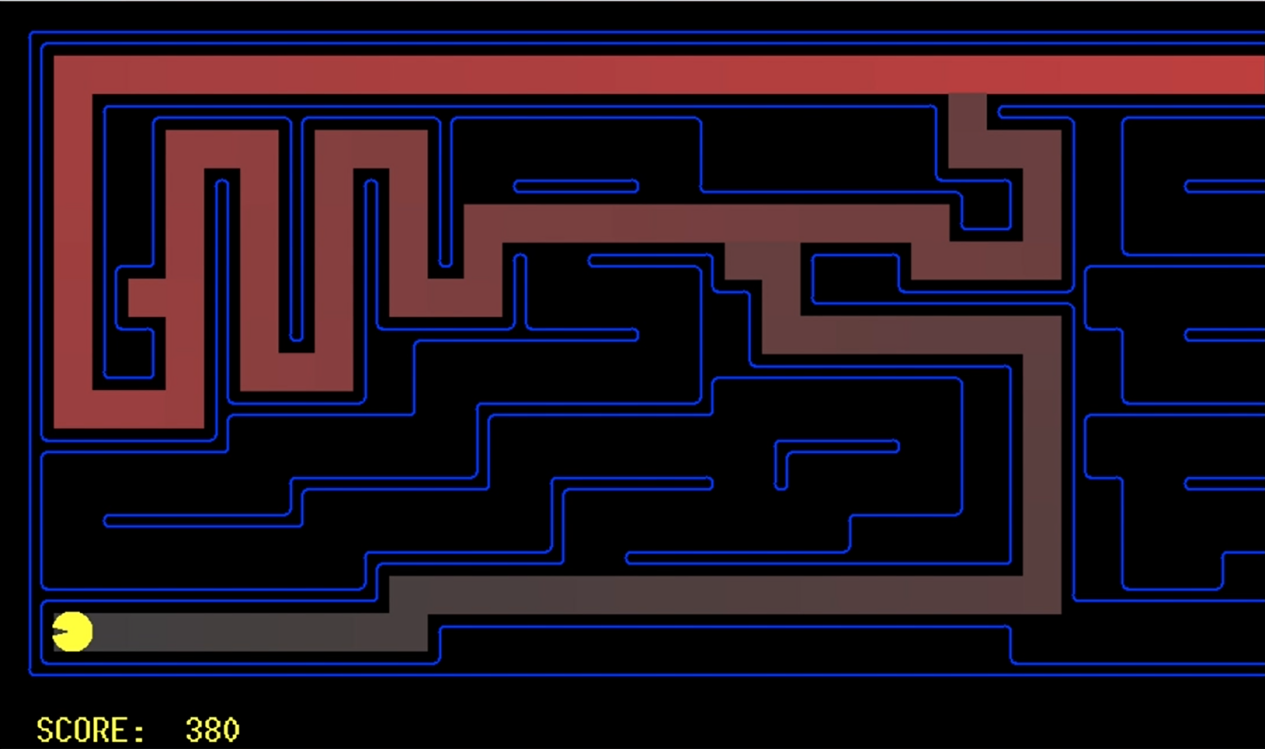
###### 实验说明：

本次实验有2个部分，分别是Search和Multiagent。具体而言，Search的目标是吃豆人仅仅是寻找食物；Multiagent的目标是吃完所有食物，同时避开鬼。抽象而言，Search实现的静态查找算法，Multiagent的问题是在有对手的情况下做出下一步决策使自己的利益最大化。Search部分需要你实现BFS算法和A\*算法。Multiagent部分需要你实现minimax算法和alpha-beta剪枝。你只需要并且只能修改并向助教提交myImpl.py文件，阅读其他代码对完成实验没有意义。请不要在myImpl.py文件中import其他模块，否则会造成测试失败。实验代码量大约为100行以内。实验需要使用Python 3.6版本，建议使用anaconda来管理Python环境。本实验推荐使用Linux，测试只需要在命令行中运行./test.sh。正确代码应该PASS所有的测试。如果实现的代码有误，请善用报错信息和print()函数。

###### Vlab实现截图：

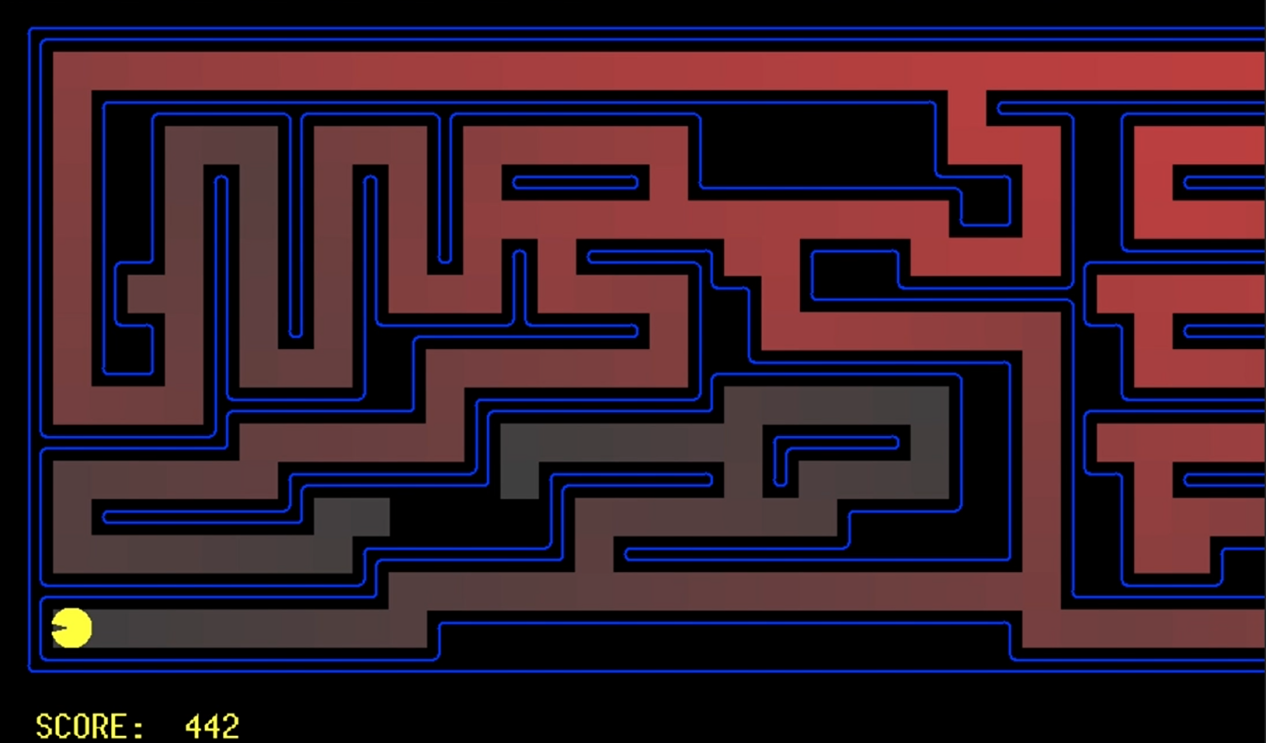
深度优先搜索：

深度优先搜索采用堆栈寻找路径，首先从起始结点出发，判断是否为目标结点，若否，寻找与该结点的邻接点，先搜索一条分支上的所有节点，然后再去搜索起始节点的其它分支结点，找出并存进待扩展结点表，等待扩展，每次先判断待扩展结点表是否为空，若否，则从待扩展结点表中取出一个结点进行扩展，并将扩展后的结点存进该表，若是，则返回失败。



广度优先搜索：

属于一种盲目搜寻法，目的是系统地展开并检查图中的所有节点，以找寻结果。换句话说，它并不考虑结果的可能位置，彻底地搜索整张图，直到找到结果为止，且搜索出来的路径为最短路径。



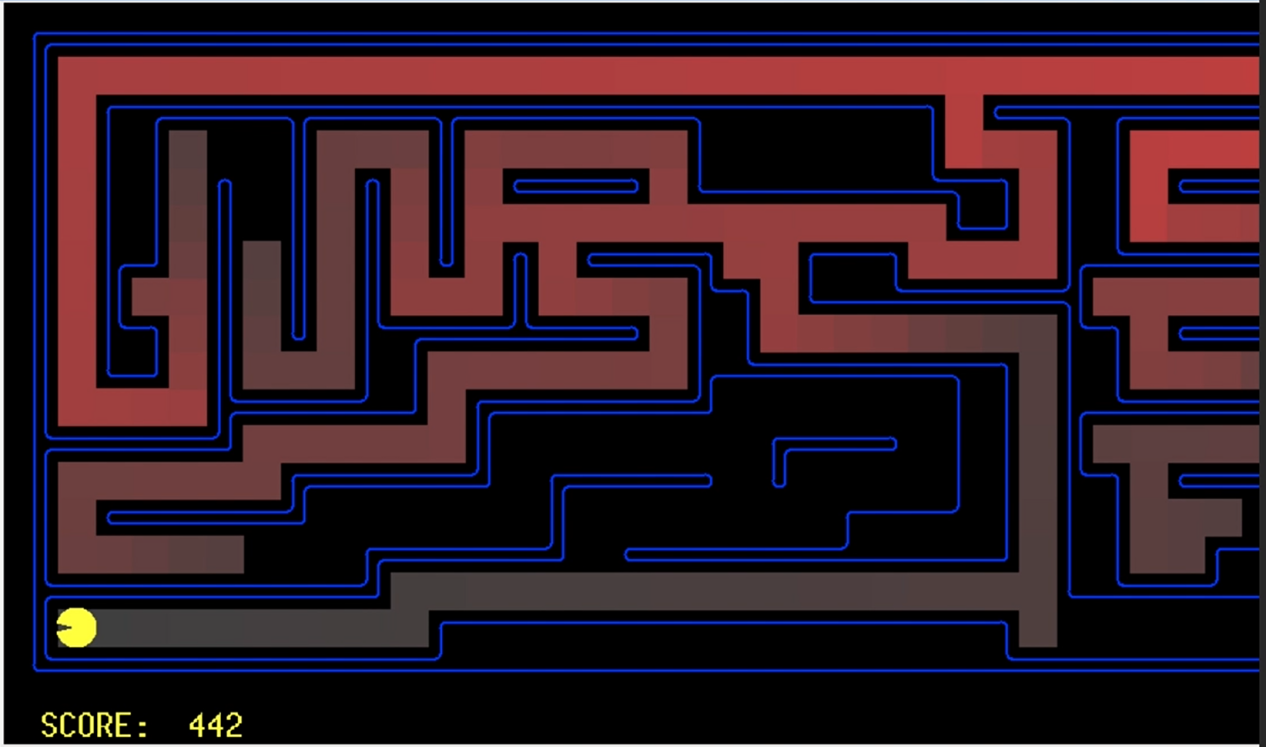
A\*算法：

公式表示为： f(n)=g(n)+h(n),其中 f(n) 是从初始点经由节点n到目标点的估价函数，g(n) 是在状态空间中从初始节点到n节点的实际代价，h(n) 是从n到目标节点最佳路径的估计代价。保证找到最短路径（最优解的）条件，关键在于估价函数f(n)的选取：首先算法开始时：

1、如果表不为空，从表头取一个结点n，如果为空算法失败。

2、n是目标解吗？是，找到一个解（继续寻找，或终止算法）。

3、将n的所有后继结点展开，就是从n可以直接关联的结点（子结点），如果不在表中，就将它们放入表，同时计算每一个后继结点的估价值f(n)，将OPEN表按f(x)排序，最小的放在表头，重复算法，回到1。



MinMax:

利用MinMax博弈树，这里模拟的Ghost可能不止一个，在计算Min节点的时候增加了对多Ghost的支持。

实际运行游戏时候，可以模拟多个Ghost，从而选择威胁最大的那个作为最终的min节点。

AlphaBate剪枝:

ɑ路径上发现的MAX 方的最佳值；β路径上发现的MIN 方的最佳值；在搜索的过程中，ɑ-β 算法不断地更新MAX 方的\_ 值和MIN 方的ɑ值，并且一旦条件成熟时即进行剪枝：MAX 方发现回传值低于自己的当前最佳值，即进行β剪枝；MIN 方发现回传值高于自己的当前最佳值，即进行ɑ剪枝。

结果如下：

\*\*\* PASS: test\_cases/q2/0-eval-function-lose-states-1.test

\*\*\* PASS: test\_cases/q2/0-eval-function-lose-states-2.test

\*\*\* PASS: test\_cases/q2/0-eval-function-win-states-1.test

\*\*\* PASS: test\_cases/q2/0-eval-function-win-states-2.test

\*\*\* PASS: test\_cases/q2/0-lecture-6-tree.test

\*\*\* PASS: test\_cases/q2/0-small-tree.test

\*\*\* PASS: test\_cases/q2/1-1-minmax.test

\*\*\* PASS: test\_cases/q2/1-2-minmax.test

\*\*\* PASS: test\_cases/q2/1-3-minmax.test

\*\*\* PASS: test\_cases/q2/1-4-minmax.test

\*\*\* PASS: test\_cases/q2/1-5-minmax.test

\*\*\* PASS: test\_cases/q2/1-6-minmax.test

\*\*\* PASS: test\_cases/q2/1-7-minmax.test

\*\*\* PASS: test\_cases/q2/1-8-minmax.test

\*\*\* PASS: test\_cases/q2/2-1a-vary-depth.test

\*\*\* PASS: test\_cases/q2/2-1b-vary-depth.test

\*\*\* PASS: test\_cases/q2/2-2a-vary-depth.test

\*\*\* PASS: test\_cases/q2/2-2b-vary-depth.test

\*\*\* PASS: test\_cases/q2/2-3a-vary-depth.test

\*\*\* PASS: test\_cases/q2/2-3b-vary-depth.test

\*\*\* PASS: test\_cases/q2/2-4a-vary-depth.test

\*\*\* PASS: test\_cases/q2/2-4b-vary-depth.test

\*\*\* PASS: test\_cases/q2/2-one-ghost-3level.test

\*\*\* PASS: test\_cases/q2/3-one-ghost-4level.test

\*\*\* PASS: test\_cases/q2/4-two-ghosts-3level.test

\*\*\* PASS: test\_cases/q2/5-two-ghosts-4level.test

\*\*\* PASS: test\_cases/q2/6-tied-root.test

\*\*\* PASS: test\_cases/q2/7-1a-check-depth-one-ghost.test

\*\*\* PASS: test\_cases/q2/7-1b-check-depth-one-ghost.test

\*\*\* PASS: test\_cases/q2/7-1c-check-depth-one-ghost.test

\*\*\* PASS: test\_cases/q2/7-2a-check-depth-two-ghosts.test

\*\*\* PASS: test\_cases/q2/7-2b-check-depth-two-ghosts.test

\*\*\* PASS: test\_cases/q2/7-2c-check-depth-two-ghosts.test

\*\*\* Running MinimaxAgent on smallClassic 1 time(s).

\*\*\* PASS: test\_cases/q3/0-eval-function-lose-states-1.test

\*\*\* PASS: test\_cases/q3/0-eval-function-lose-states-2.test

\*\*\* PASS: test\_cases/q3/0-eval-function-win-states-1.test

\*\*\* PASS: test\_cases/q3/0-eval-function-win-states-2.test

\*\*\* PASS: test\_cases/q3/0-lecture-6-tree.test

\*\*\* PASS: test\_cases/q3/0-small-tree.test

\*\*\* PASS: test\_cases/q3/1-1-minmax.test

\*\*\* PASS: test\_cases/q3/1-2-minmax.test

\*\*\* PASS: test\_cases/q3/1-3-minmax.test

\*\*\* PASS: test\_cases/q3/1-4-minmax.test

\*\*\* PASS: test\_cases/q3/1-5-minmax.test

\*\*\* PASS: test\_cases/q3/1-6-minmax.test

\*\*\* PASS: test\_cases/q3/1-7-minmax.test

\*\*\* PASS: test\_cases/q3/1-8-minmax.test

\*\*\* PASS: test\_cases/q3/2-1a-vary-depth.test

\*\*\* PASS: test\_cases/q3/2-1b-vary-depth.test

\*\*\* PASS: test\_cases/q3/2-2a-vary-depth.test

\*\*\* PASS: test\_cases/q3/2-2b-vary-depth.test

\*\*\* PASS: test\_cases/q3/2-3a-vary-depth.test

\*\*\* PASS: test\_cases/q3/2-3b-vary-depth.test

\*\*\* PASS: test\_cases/q3/2-4a-vary-depth.test

\*\*\* PASS: test\_cases/q3/2-4b-vary-depth.test

\*\*\* PASS: test\_cases/q3/2-one-ghost-3level.test

\*\*\* PASS: test\_cases/q3/3-one-ghost-4level.test

\*\*\* PASS: test\_cases/q3/4-two-ghosts-3level.test

\*\*\* PASS: test\_cases/q3/5-two-ghosts-4level.test

\*\*\* PASS: test\_cases/q3/6-tied-root.test

\*\*\* PASS: test\_cases/q3/7-1a-check-depth-one-ghost.test

\*\*\* PASS: test\_cases/q3/7-1b-check-depth-one-ghost.test

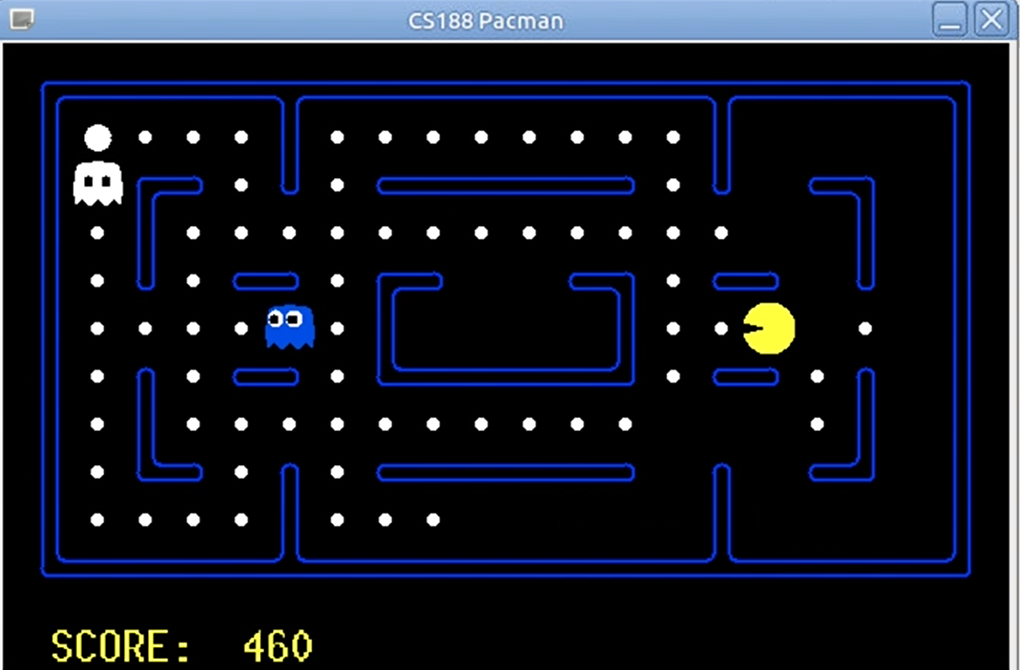
\*\*\* PASS: test\_cases/q3/7-1c-check-depth-one-ghost.test

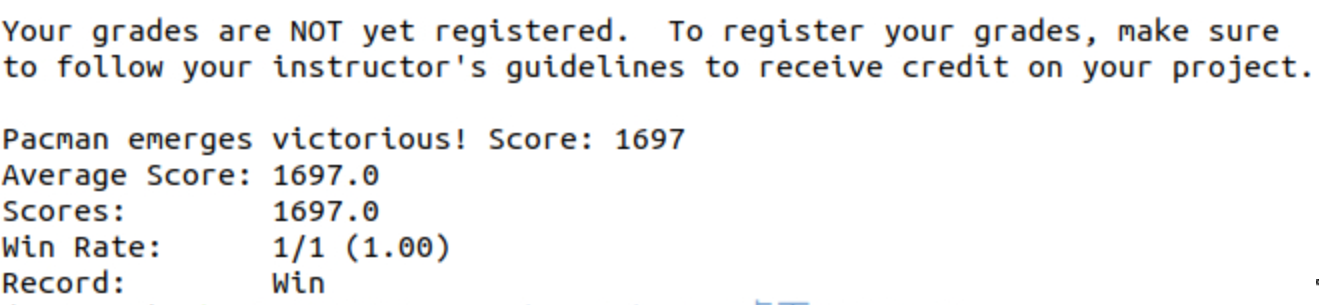
\*\*\* PASS: test\_cases/q3/7-2a-check-depth-two-ghosts.test

\*\*\* PASS: test\_cases/q3/7-2b-check-depth-two-ghosts.test

\*\*\* PASS: test\_cases/q3/7-2c-check-depth-two-ghosts.test

\*\*\* Running AlphaBetaAgent on smallClassic 1 time(s).





小结：

经过多次运行，大部分情况下Agent都能战胜取得胜利，少数输掉游戏的原因经过理论分析应该为：前期结点出现的顺序不符合剪枝条件，导致树出现不能取胜的情况。

###### 遇到的问题与解决：

在初次完成运行发现A\*算法的收益过大，其实是算法中开始写的代码实现时本质上只加了step\_cost和h，也就是说对于每一个节点并没有把已经经过的代价全部计算进去。在修改传参g进入函数且每次迭代g = g+step\_cost后发现编译结果出现问题，仔细考虑代码的问题再次修改函数内部为g\_。证明参数名和引用之间冲突的小问题易被忽略。

同时，最终修改完善代码解决所有问题pass之前，在包括上述一些小问题出现尽管在test有部分未通过的前提下，仍然发现运行结果总能正常运行，且Agent保持很高胜率。这证明了几个事情：

· Ghost其实并没有博弈的概念，所以大部分猜想是浪费时间的

· Pacman游戏其实只需要局部考虑，无需过多全局考虑，也就是说：

· 当Ghost离Agent足够远的时候，其实Ghost的行动对于Pacman影响不大，没必要过多考虑。