人工智能实验一：搜索策略 pacman

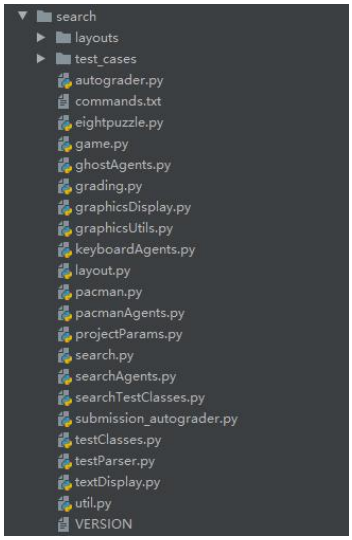
1、实验内容

要求采用但不限于课程第四章内各种搜索算法编写一系列吃豆人程序解决以下列出的问题 1-8，包括找到一个指定位置的豆子、到达四个角落、吃掉所有豆子。

实验目的，加深对课程介绍的各种搜索算法的理解。

2、实验简介

pacman 是加州大学伯克利分校开源的人工智能实验项目，实验的初衷是在有趣的图形化可视化游戏界面中编写 AI 策略，课程地址如下 https://inst.eecs.berkeley.edu/~cs188/fa20/project1/（建议看一遍英文原文）。下载 search.zip 压缩包，解压后有如下文件。



整个项目使用 python 开发，支持 windows 和 linux 操作系统，因历史原因有 python2 和 python3 两个版本，建议只用 python3 版本。若在自己电脑上运行，需提前安装好 python 环境。无 python 基础的同学请先自行学习，可网上搜索教程或参考 https://inst.eecs.berkeley.edu/~cs188/su21/project0/。

Pacman 项目实现的功能比较多，在 search 目录下，运行以下命令可以进入交互式的吃豆人游戏，如下图，通过方向键可以控制 Agent(智能体)移动躲避 ghost，吃完所有的豆子就是胜利。本实验只在非交互式模式下让智能体自动完成任务，无需方向键控制。

python pacman.py

3、search.zip 文件说明

项目使用 Tkinter 包作为 GUI 开发框架，界面实现相关的文件如下，具体实现原理、怎么实现比较复杂，不需做了解。

```
graphicsDisplay.py
graphicsUtils.py
textDisplay.py
ghostAgents.py
keyboardAgents.py
PacmanAgents.py
layout.py    # 加载地图模块
layouts/     # 地图文件
```

几个主要文件及功能如下

文件	主要功能
pacman.py	吃豆人游戏的主程序，文件包括一个描述”吃豆人”gamestate 的类。
game.py	吃豆人游戏的运行逻辑，文件包括以下类 AgentState, Agent, Direction, and Grid.
util.py	搜索策略可以用到的数据结构。

几个支持文件及功能如下

autograder.py	Project autograder
testParser.py	Parses autograder test and solution files

testClasses.py	General autograding test classes
searchTestClasses.py	Project 1 specific autograding test classes
test_cases/	Directory containing the test cases for each question
commands.txt	常用的测试命令

完成实验所要修改的文件只有 search.py 和 searchAgents.py，其他文件无需修改，不建议新建文件。需要补充代码的地方以注释"*** YOUR CODE HERE ***"或 util.raiseNotDefined()做了提示。

searchAgents.py 中最简单的 Agent 叫做 GoWestAgent，一路向西，偶尔能实现目标，但不能转弯，使用如下命令指定地图可以测试 GoWestAgent。

（1）地图简单，GoWestAgent 能吃到豆子。

python pacman.py --layout testMaze --pacman GoWestAgent

（2）因地图有转弯，GoWestAgent 将吃不到豆一直运行，可通过 CTRL-c 来终止。

python pacman.py --layout tinyMaze --pacman GoWestAgent

python pacman.py 命令后面可以跟的选项及取值可以通过看 pacman.py 的代码进行了解，如下图，

```
parser = OptionParser(usageStr)

parser.add_option('-n', '--numGames', dest='numGames', type='int',
                  help=default('the number of GAMES to play'), metavar='GAMES', default=1)
parser.add_option('-l', '--layout', dest='layout',
                  help=default('the LAYOUT_FILE from which to load the map layout'),
                  metavar='LAYOUT_FILE', default='mediumClassic')
parser.add_option('-p', '--pacman', dest='pacman',
                  help=default('the agent TYPE in the pacmanAgents module to use'),
                  metavar='TYPE', default='KeyboardAgent')
parser.add_option('-t', '--textGraphics', action='store_true', dest='textGraphics',
                  help='Display output as text only', default=False)
parser.add_option('-q', '--quietTextGraphics', action='store_true', dest='quietGraphics',
                  help='Generate minimal output and no graphics', default=False)
parser.add_option('-g', '--ghosts', dest='ghost',
                  help=default('the ghost agent TYPE in the ghostAgents module to use'),
                  metavar = 'TYPE', default='RandomGhost')
parser.add_option('-k', '--numghosts', type='int', dest='numGhosts',
                  help=default('The maximum number of ghosts to use'), default=4)
parser.add_option('-z', '--zoom', type='float', dest='zoom',
                  help=default('Zoom the size of the graphics window'), default=1.0)
parser.add_option('-f', '--fixRandomSeed', action='store_true', dest='fixRandomSeed',
                  help='Fixes the random seed to always play the same game', default=False)
parser.add_option('-r', '--recordActions', action='store_true', dest='record',
                  help='Writes game histories to a file (named by the time they were played)', default=False)
parser.add_option('--replay', dest='gameToReplay',
                  help='A recorded game file (pickle) to replay', default=None)
parser.add_option('-a','--agentArgs',dest='agentArgs',
                  help='Comma separated values sent to agent. e.g. "opt1=val1,opt2,opt3=val3"')
parser.add_option('-x', '--numTraining', dest='numTraining', type='int',
                  help=default('How many episodes are training (suppresses output)'), default=0)
parser.add_option('--frameTime', dest='frameTime', type='float',
                  help=default('Time to delay between frames; <0 means keyboard'), default=0.1)
parser.add_option('-c', '--catchExceptions', action='store_true', dest='catchExceptions',
                  help='Turns on exception handling and timeouts during games', default=False)
parser.add_option('--timeout', dest='timeout', type='int',
                  help=default('Maximum length of time an agent can spend computing in a single game'), default=30)

options, otherjunk = parser.parse_args(argv)
```

也可以通过 python pacman.py -h 查看，主要的选项说明如下:

--layout，简写为“-l”，地图选项，从 layouts/目录下加载指定类型地图，地图文件内容可以尝试修改。

--pacman，简写为-p,指定 agent 类型，默认是 KeyboardAgent 即交互式操作，通过键盘方向键能控制的智能体。

--zoom，简写为-z，图形窗口的大小，默认是 1，比 1 大则是放大，比 1 小则是缩小。

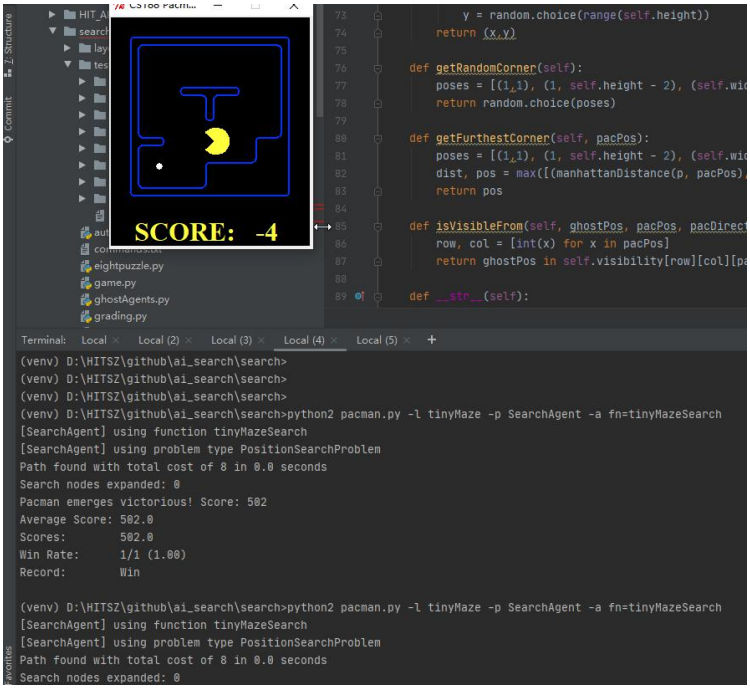
--agentArgs，简写为-a,传递给 agent 的 args 取值，字符串的形式，如果有多个取值以逗号分隔，比如"opt1=val1,opt2,opt3=val3"。

4、待解决的 8 个搜索问题

问题 1：应用深度优先算法找到一个特定的位置的豆

首先，运行以下命令测试 SearchAgent 是不是正常工作，正常情况会弹出如下所示的游戏界面。命令中 fn 是 function 的简写，能够使用的函数是在 search.py 文件中定义的函数，提供了 bfs,dfs,astar,ucs 四个缩写。

python pacman.py -l tinyMaze -p SearchAgent -a fn=tinyMazeSearch



tinyMazeSearch 默认已实现，只能在 tinyMaze 地图中成功找到豆子。接下来需要完成完整的搜索算法帮助吃豆人规划路线。注意，一个搜索节点不仅包含节点的状态，而且要包含构建搜索路径所需要的信息。

你的 code 应该能顺利解决以下问题：

python pacman.py -l tinyMaze -p SearchAgent

python pacman.py -l mediumMaze -p SearchAgent

python pacman.py -l bigMaze -z .5 -p SearchAgent

问题 2：应用宽度优先算法找到一个特定的位置的豆

利用宽度优先算法实现解决以上问题，并利用以下命令测试你的 code：

python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs

python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5

问题 3：应用代价一致算法找到一个特定的位置的豆

通过修改代价函数，我们鼓励 Pacman 发现不同路径。例如，有恶魔的区域，我们增加每步的代价，而在食物丰富的区域减少每步的代价，一个理性的 Pacman 应该相应地调整它的行为。

完成代价一致搜索方法(search.py 文件中的 uniformCostSearch 函数)，并用以下命令测试你的 code：

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

再与以下两种实现进行对比，找出与 StayEastSearchAgent，StayWestSearchAgent 的区别。

python pacman.py -l mediumDottedMaze -p StayEastSearchAgent

python pacman.py -l mediumScaryMaze -p StayWestSearchAgent

问题 4：应用 A* 算法找到一个特定的位置的豆

完成 A*搜索方法(search.py 文件中的 aStarSearch 函数)，利用曼哈顿距离作为启发函数，用以下命令测试你的 code：

python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic

注意：所有的搜索函数最后必须返回一个从初始状态到目标状态的操作序列，所有操作必须合法（不能翻墙）。实现搜索算法用到的数据结构必须使用 util.py 文件中提供的 Stack, Queue 和 PriorityQueue，这是自动评分系统的兼容性要求。

以上四个问题对应的不同搜索算法可以单独实现，但建议统一到通用的搜索算法框架当中来。通用搜索算法的伪代码见下图，不同的搜索方法的不同之处仅仅在于 open 表的排序不同，问题 1-4 的不同之处在于用不同的数据结构对 open 表进行排序。

Graph Search Pseudo-Code

```
function GRAPH-SEARCH(problem, fringe) return a solution, or failure
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        if STATE[node] is not in closed then
            add STATE[node] to closed
            for child-node in EXPAND(STATE[node], problem) do
                fringe ← INSERT(child-node, fringe)
            end
    end
```

问题 5：找到所有的角落——基于 BFS 的角落问题（CornersProblem Based on BFS）

在角落迷宫的四个角上面有四个豆，这个搜索问题要求找到一条访问所有四个角落的最短的路径。完成 searchAgents.py 文件中的 CornersProblem 搜索问题，你需要重新定义状态，使其能够表示角落是否被访问。用以下命令测试你得 code：

python pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem

python pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem

提示：新的状态只包含吃豆人的位置和角落的状态。

问题 6:找到所有的角落——基于 A*的角落问题(CornersProblem Based on A*)

在问题 5 完成的 CornersProblem 基础上，使用 A*算法，构建合适的启发函数，找到一条访问地图所有四个角落的最短路径。完成 searchAgents.py 文件中的 cornersHeuristic 角落搜索问题。用以下命令测试你的 code：

python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5

AStarCornersAgent 等同于以下选项

-p SearchAgent -a fn=aStarSearch,prob=CornersProblem,heuristic=cornersHeuristic

不同的启发函数扩展的节点不一样得分也会不一样，要求启发函数是非平凡非负一致的，一致性即单调性。autograder.py 会跟根据扩展的节点数给分，标准如下。

Number of nodes expanded	Grade
more than 2000	0/3
at most 2000	1/3
at most 1600	2/3
at most 1200	3/3

问题 7：吃掉所有的豆子——食物搜索问题（FoodSearchProblem）

构造恰当的启发式函数，利用 A*算法，以尽可能少的步数吃掉地图中存在的所有豆子,即完成 searchAgents.py 文件中的 FoodSearchProblem 搜索问题。如果你已经正确地完成了前面的搜索算法，不需要改代码，使用 null heuristic 的 A*（等价于代价一致算法） 将很快求得 testSearch 问题的最优解，可用以下命令测试：

python pacman.py -l testSearch -p AStarFoodSearchAgent

现在，请补充完成 searchAgents.py 文件中的 foodHeuristic 函数，用以下命令测试你的 code：

python pacman.py -l trickySearch -p AStarFoodSearchAgent

不同的启发函数扩展的节点会不一样，要求启发函数是非平凡非负一致的，autograder.py 会跟根据扩展的节点数给分，标准如下。

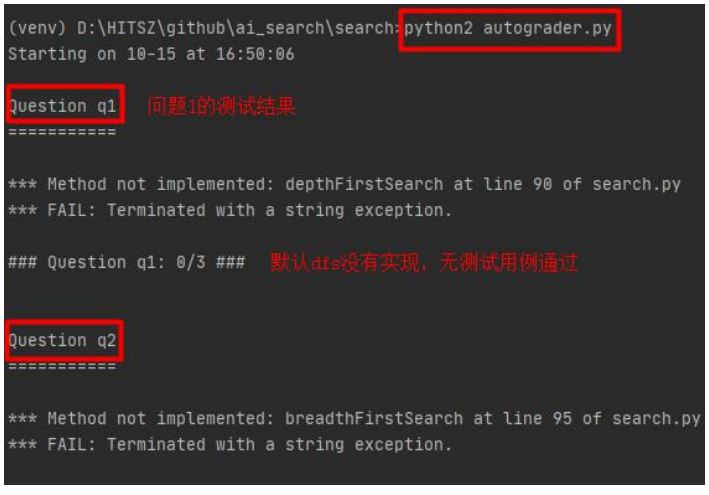
Number of nodes expanded	Grade
more than 15000	1/4
at most 15000	2/4
at most 12000	3/4
at most 9000	4/4 (full credit; medium)
at most 7000	5/4 (optional extra credit; hard)

问题 8：次最优搜索——任意食物搜索问题（AnyFoodSearchProblem）。

有的时候，即使使用 A* 加上好的启发式，吃掉所有豆子的最优路径也是困难的。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。在本节中，你需要实现一个智能体，它总是吃掉最近的豆。在 searchAgents.py 中已经实现了 ClosestDotSearchAgent，但缺少关键函数 findPathToClosestDot，该函数搜索到最近豆的路径，补充其实现。在实现 findPathToClosestDot 之前请先实现 AnyFoodSearchProblem 目标测试函数 isGoalState。用以下命令测试你的 code：

python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5

实现对应的代码之后，运行 python autograder.py 对 8 个问题进行评分，或者添加选项 python autograder.py -q q1 对特定的问题评分。若没有实现对应算法或者实现出错，执行 python autograder.py 会有提示未通过原因。



如果代码实现正确，运行 python autograder.py，会看到测试用例输出。

```
(venv) D:\HITSZ\github\ai_search\search>python2 autograder.py
Starting on 10-16 at 9:12:01

Question q1
===========

*** PASS: test_cases\q1\graph_backtrack.test
***     solution:               ['1:A->C', '0:C->G']
***     expanded_states:        ['A', 'D', 'C']
*** PASS: test_cases\q1\graph_bfs_vs_dfs.test
***     solution:               ['2:A->D', '0:D->G']
***     expanded_states:        ['A', 'D']
*** PASS: test_cases\q1\graph_infinite.test
***     solution:               ['0:A->B', '1:B->C', '1:C->G']
***     expanded_states:        ['A', 'B', 'C']
*** PASS: test_cases\q1\graph_manypaths.test
***     solution:               ['2:A->B2', '0:B2->C', '0:C->D', '2:D->E2', '0:E2->F', '0:F->G']
***     expanded_states:        ['A', 'B2', 'C', 'D', 'E2', 'F']
```

如果运行 python autograder.py 出现以下错误，参考 https://stackoverflow.com/questions/62470666/getting-this-error-with-py2-7-as-well-as-with-py3-7 解决，import html，用 html.escape 代替 cgi.escape。

```
Traceback (most recent call last):
  File "D:\code\AI_2022\ref-search-python3\autograder.py", line 355, in <module>
    evaluate(options.generateSolutions, options.testRoot, moduleDict,
  File "D:\code\AI_2022\ref-search-python3\autograder.py", line 311, in evaluate
    grades.grade(sys.modules[__name__], bonusPic = projectParams.BONUS_PIC)
  File "D:\code\AI_2022\ref-search-python3\grading.py", line 81, in grade
    self.addExceptionMessage(q, inst, traceback)
  File "D:\code\AI_2022\ref-search-python3\grading.py", line 149, in addExceptionMessage
    self.fail('FAIL: Exception raised: %s' % inst)
  File "D:\code\AI_2022\ref-search-python3\grading.py", line 272, in fail
    self.addMessage(message, raw)
  File "D:\code\AI_2022\ref-search-python3\grading.py", line 294, in addMessage
    message = cgi.escape(message)
AttributeError: module 'cgi' has no attribute 'escape'
```

最后提醒，项目采用的是伯克利的开源成果，在 pacman.py 文件开头的 license 信息做了说明，严禁将解法、报告公布，请尊重自己和伯克利的知识产权。

```
# pacman.py
# ---------
# Licensing Information:  You are free to use or extend these projects for
# educational purposes provided that (1) you do not distribute or publish
# solutions, (2) you retain this notice, and (3) you provide clear
# attribution to UC Berkeley, including a link to http://ai.berkeley.edu.
#
# Attribution Information: The Pacman AI projects were developed at UC Berkeley.
# The core projects and autograders were primarily created by John DeNero
# (denero@cs.berkeley.edu) and Dan Klein (klein@cs.berkeley.edu).
# Student side autograding was added by Brad Miller, Nick Hay, and
# Pieter Abbeel (pabbeel@cs.berkeley.edu).
```

5、实验报告与提交说明

实验分小组完成，每小组 3 或 4 人，每人提交一份报告和代码，实验报告独

立完成（内容只需写自己完成的部分），严禁抄袭，代码提交小组最终完整版（只需要提交修改后的 search.py、searchAgents.py 文件或其他新增代码），提交截止时间答辩前一天，具体见作业提交系统。最后一次课小组答辩所有成员都要上台。http://grader.tery.top:8000/#/courses。（初始账号密码都是学号）

实验报告首页至少包括以下信息，格式不限：

实验名称、姓名、学号、班级、上课学期、实验地点、授课教师、同组成员

实验报告内容包含但不限于以下内容。

一. 实验内容

概述你对 8 个问题及所用算法的理解、之间的关联。

概述你所完成的是哪几个问题。

二. 算法介绍

算法文字介绍、状态空间描述、流程图等；若涉及到启发函数，给出启发函数的设计说明、相关证明等；如同一问题实验了多种方案，可分别列出实验结果。

三. 算法实现

3.1 实验环境与问题规模

3.2 关键代码、主要数据结构及说明

3.3 实验结果

测试命令、输出结果，要求有屏幕截图及说明，包括 pacman.py 和 autograder.py 评分运行截图。同一类问题不同算法在相同地图实现效果的对比分析，如扩展节点、用时、cost 等。

四. 总结

4.1 碰到的问题及解决方法

4.2 实验的启发、总结、建议

参考文献