

人工智能实验指导书

实验 1-搜索策略 pacman

2023 春

目录

人工智能实验指导书.....	1
1、 实验目的	3
2、 实验内容	3
3、 实验简介	3
4、 框架代码说明	5
5、 待解决的 8 个搜索问题	8
问题 1: 应用深度优先算法找到一个特定位置的豆子	9
问题 2: 应用宽度优先算法找到一个特定位置的豆子	10
问题 3: 应用代价一致算法找到一个特定位置的豆子	10
问题 4: 应用 A* 算法找到一个特定位置的豆子	10
问题 5: 找到所有的角落——基于 BFS 的角落问题 (CornersProblem Based on BFS)	11
问题 6: 找到所有的角落——基于 A* 的角落问题 (CornersProblem Based on A*)	11
问题 7: 吃掉所有的豆子——食物搜索问题 (FoodSearchProblem)	12
问题 8: 次最优搜索——任意食物搜索问题 (AnyFoodSearchProblem)	12
6、 autograder 测试	13
7、 Python 学习与调试	14
8、 实验结果提交	17

1、 实验目的

通过 pacman 实验，加深对课程介绍的各种搜索算法的理解。

2、 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，解决列出的 8 个搜索问题：

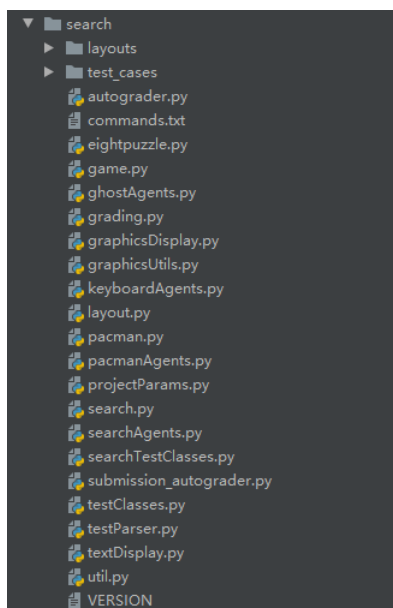
- 问题 1：应用深度优先算法找到一个特定位置的豆子
- 问题 2：应用宽度优先算法找到一个特定位置的豆子
- 问题 3：应用代价一致算法找到一个特定位置的豆子
- 问题 4：应用 A* 算法找到一个特定位置的豆子
- 问题 5：找到所有的角落——基于 BFS 的角落问题 (CornersProblem Based on BFS)
- 问题 6：找到所有的角落——基于 A* 的角落问题 (CornersProblem Based on A*)
- 问题 7：吃掉所有的豆子——食物搜索问题 (FoodSearchProblem)
- 问题 8：次最优搜索——任意食物搜索问题 (AnyFoodSearchProblem)

3、 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，实验地址如下

<https://inst.eecs.berkeley.edu/~cs188/sp22/project1/>。

基本代码和支持文件可以从 search-su22.zip 中获取，search-su22.zip 解压后有如下文件：



整个项目使用 python3 开发，支持 windows 和 linux 操作系统。在 linux 虚拟机或 wsl 环境需要做相关配置才能正常显示图形，故建议在 windows 下完成。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/su22/project0/>

Pacman 项目实现的功能比较多，在 search-su22 目录下，运行以下命令打开吃豆人游戏：

```
python pacman.py
```

运行效果如下图：



通过方向键可以控制吃豆人移动躲避 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。

4、 框架代码说明

项目使用 Tkinter 包作为 GUI 开发框架。几个支持文件和界面相关文件如下，这些文件不做修改，可先不研究：

文件名	功能
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
autograder.py	Project autograder
graphicsDisplay.py graphicsUtils.py textDisplay.py ghostAgents.py keyboardAgents.py PacmanAgents.py layout.py layouts/	界面实现相关的文件

以下几个文件比较重要，需要了解一下：

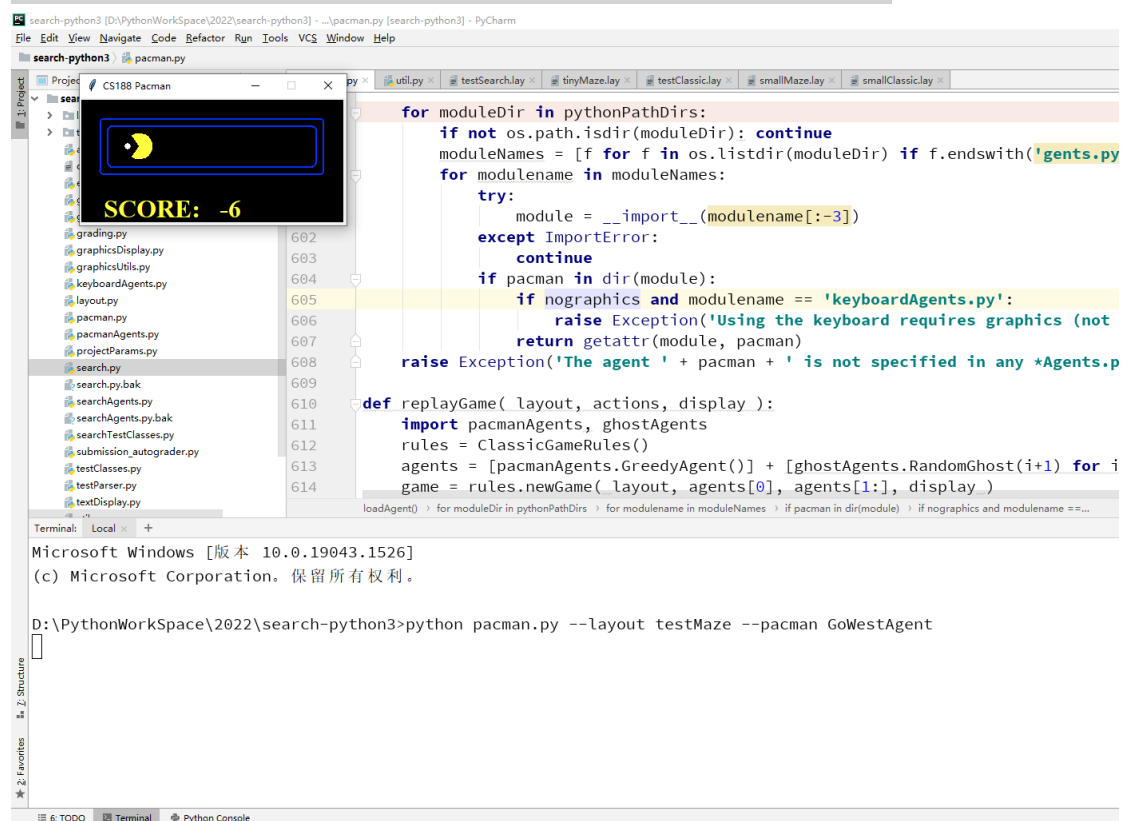
文件名	功能
pacman.py	吃豆人游戏的主程序。文件包括一个描述“吃豆人”GameState 的类型。
game.py	吃豆人游戏的运行逻辑。文件包含了像 AgentState、Agent、Direction、Grid 等几个起到支持作用的类。
util.py	提供一些可供搜索策略使用的数据结构。

需要编辑的文件：**search.py** 和 **searchAgents.py**，整个实验只有这 2 个文件需要修改，需要补充代码的地方以注释“***** YOUR CODE HERE *****”或 **util.raiseNotDefined()** 做了提示。其他文件无需修改，也不建议新建文件。

Pacman 项目实现的功能比较多，不同的任务用了不同的迷宫和不同的 Agent 实现。Agent 的实现在 **searchAgents.py** 中。**searchAgents.py** 中最简单的 Agent 叫做 **GoWestAgent**（一路向西），顾名思义，用 **GoWestAgent** 时吃豆人只能一直往左走，不能转弯。这种走法偶尔在某些地图中能实现目标。使用如下命令指定地图和 agent 运行：

(1) 地图简单，一路向西也能吃到豆子

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

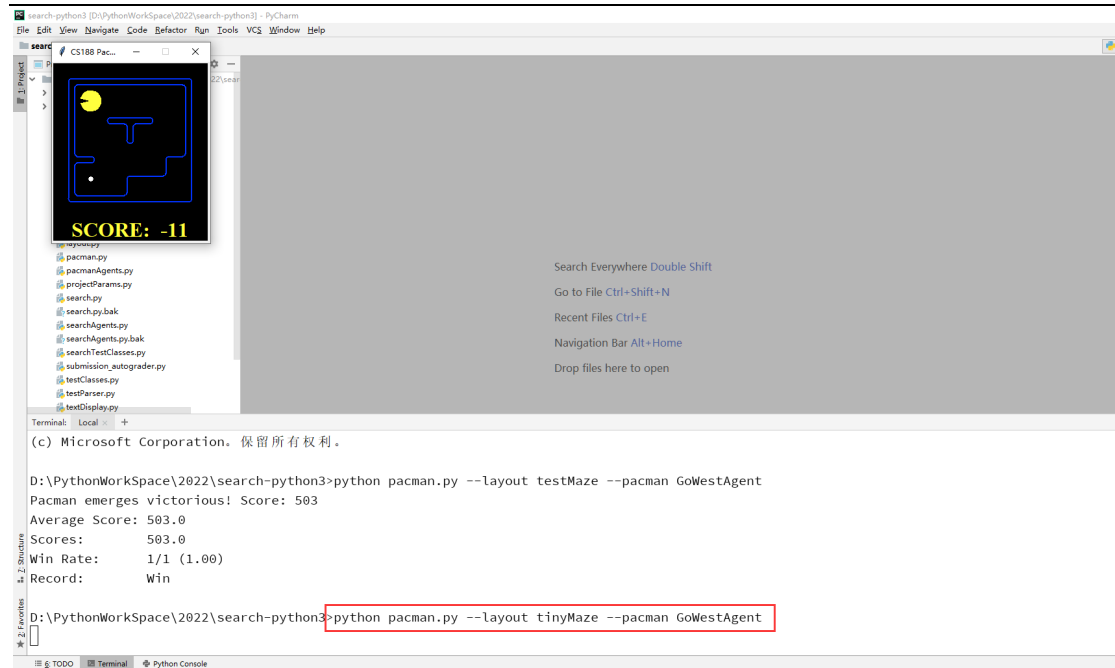


游戏结束后可以看到结果：

```
D:\PythonWorkspace\2022\search-python3>python pacman.py --layout testMaze --pacman GoWestAgent
Pacman emerges victorious! Score: 503
Average Score: 503.0
Scores:          503.0
Win Rate:        1/1 (1.00)
Record:          Win
```

(2) 因地图布局原因，吃不到豆会一直运行，可通过 **CTRL-c** 来终止

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



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`，指定地图选项，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 `searchAgents` 模块中的 agent 类型。比如：`GoWestAgent`、`SearchAgent`、`StayEastSearchAgent` 等。

--agentArgs, 简写为-a, 传递给 agent 的参数值。以字符串的形式, 如果有多个参数以逗号分隔, 比如"opt1=val1,opt2,opt3=val3"。

--zoom, 简写为-z, 设置图形窗口的缩放比例。默认是 1, 比 1 大则是放大, 比 1 则是缩小。

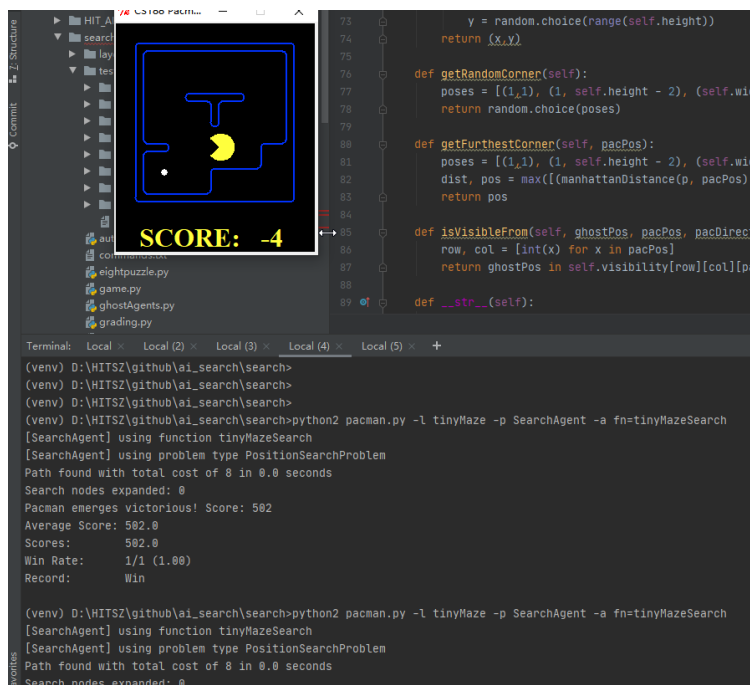
其他选项很少使用, 如需用到自行查看代码。

5、待解决的 8 个搜索问题

首先, 运行以下命令测试 SearchAgent 是不是正常工作:

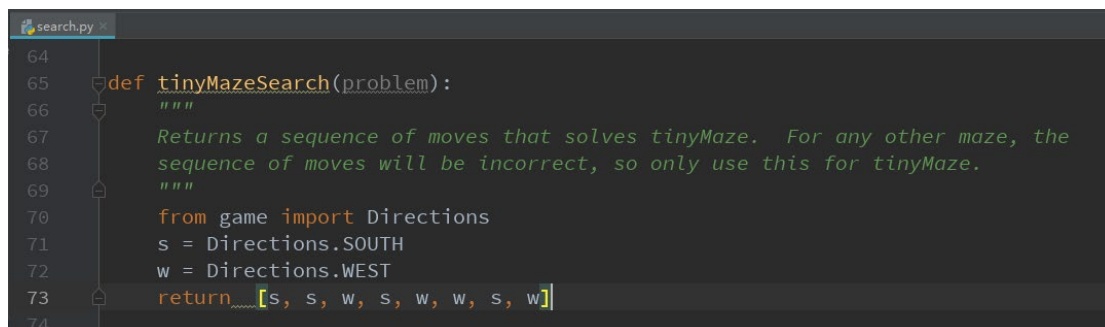
```
python pacman.py -l tinyMaze -p SearchAgent -a fn=tinyMazeSearch
```

正常情况会弹出如下所示的游戏界面, 说明环境正常:



这条命令的意思是用 tinyMazeSearch 算法在 tinyMaze 迷宫中找到特定位置的豆子。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, 另外提供了 bfs、dfs、astar、ucs 四个缩写。

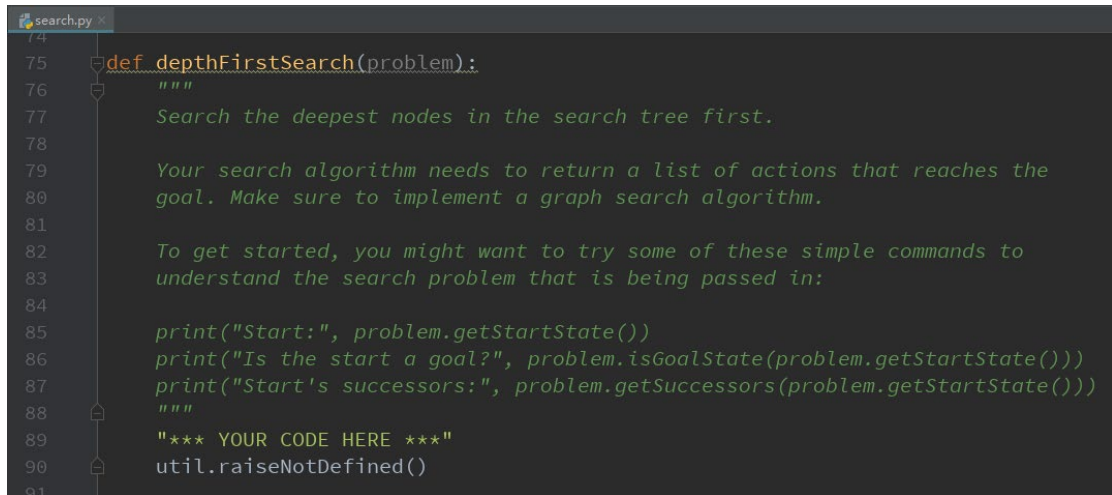
tinyMazeSearch 默认已实现, 大家看代码可以发现这个函数实际不是通过计算得到的结果, 而是直接 return 了吃豆人的路径:



接下来需要完成完整的搜索算法帮助吃豆人规划路线。

问题 1：应用深度优先算法找到一个特定位置的豆子

下面，我们需要实现深度优先搜索算法帮助吃豆人规划路线：



```
74 search.py
75 def depthFirstSearch(problem):
76     """
77     Search the deepest nodes in the search tree first.
78
79     Your search algorithm needs to return a list of actions that reaches the
80     goal. Make sure to implement a graph search algorithm.
81
82     To get started, you might want to try some of these simple commands to
83     understand the search problem that is being passed in:
84
85     print("Start:", problem.getStartState())
86     print("Is the start a goal?", problem.isGoalState(problem.getStartState()))
87     print("Start's successors:", problem.getSuccessors(problem.getStartState()))
88     """
89     "*** YOUR CODE HERE ***"
90     util.raiseNotDefined()
91
```

因为不同的搜索方法的不同之处仅仅在于 open 表的排序方法不同，**本实验要求定义一个通用的搜索算法解决问题 1-4**。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。通用搜索算法的伪代码见下图：

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
    end
```

提示：

- 1、所有的搜索函数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（比如不能翻越墙壁）。
- 2、利用 `util.py` 文件中提供的 `Stack`、`Queue` 和 `PriorityQueue` 数据结构。这是自动评分系统的兼容性要求。
- 3、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。

完成编码后，可以利用以下命令测试你的 code：

```
python pacman.py -l tinyMaze -p SearchAgent
```

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

以上 3 个测试都通过并不代表完全正确，需要再运行以下命令对问题 1 进行更完整的测试：

```
python autograder.py -q q1
```

-q 选项是指定对应的问题，其它 7 个问题建议也通过这种方法进行测试。若没有实现对应算法或者实现出错，执行之后会有错误提示。

问题 2：应用宽度优先算法找到一个特定位置的豆子

完成宽度优先算法（[search.py 文件中的 breadthFirstSearch 函数](#)），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
python autograder.py -q q2
```

问题 3：应用代价一致算法找到一个特定位置的豆子

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
python autograder.py -q q3
```

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
```

同样的地图，使用不同的代价函数，得到的路径和成本是不一样的，扩展的节点也不一样。

问题 4：应用 A* 算法找到一个特定位置的豆子

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
python autograder.py -q q4
```

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四个角落的路径。

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。

提示：新的状态只包含吃豆人的位置和角落的状态。

用以下命令测试你的 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
python autograder.py -q q5
```

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

使用 A*算法，构建合适的启发函数，找到一条访问地图四个角落的最短路径。

完成 `searchAgents.py` 文件中的 `cornersHeuristic` 角落搜索函数。用以下命令测试你的 code:

```
python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5
python autograder.py -q q6
```

注意：不同的启发函数扩展的节点数不一样，得分也会不一样。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，用尽可能少的步数吃掉所有的豆子。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（等价于代价一致算）的 A* 将很快求得 `testSearch` 问题的最优解。可用以下 2 条命令测试你的 code：

```
python pacman.py -l testSearch -p AStarFoodSearchAgent
python pacman.py -l trickySearch -p AStarFoodSearchAgent
```

现在请补充完成 `searchAgents.py` 文件中的 `foodHeuristic` 函数，完成后再运行刚才的 2 条命令，对比结果并思考原因。

最后用以下命令进行测试：

```
python autograder.py -q q7
```

注意：不同的启发函数扩展的节点会不一样，`autograder.py` 将根据扩展的节点数给分，对于 `trickySearch` 地图，节点扩展评分标准如下。

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，寻找一条吃掉所有豆子的最优路径也是困难的。定义一个优先吃离 pacman 最近的豆子的函数是提高搜索速度的一个好的办法。在本问题中，你需要实现一个智能体，它总是吃掉离它最近的豆。

在 `searchAgents.py` 中已经实现了 `ClosestDotSearchAgent`，但缺少关键方法 `findPathToClosestDot`，该方法搜索到最近豆的路径。也就是说你需要完成 `searchAgents.py` 的 `findPathToClosestDot` 方法。在实现 `findPathToClosestDot` 之前请先实现 `searchAgents.py` 文件中的 `AnyFoodSearchProblem` 类的目标测试方法 `isGoalState`。

用以下命令测试你的 code：

```
python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5
python autograder.py -q q8
```

大家还可以用刚才问题 7 用过的地图来测试，并与之前结果进行对比：

```
python pacman.py -l trickySearch -p ClosestDotSearchAgent
```

6、 autograder 测试

项目框架实现了对 8 个问题自动测评的脚本，运行 `python autograder.py` 帮助你对自己的程序打分，默认情况执行 `python autograder.py` 会有提示未通过原因。

```
(venv) D:\HITSZ\github\ai_search\search>python2 autograder.py
Starting on 10-15 at 16:50:06

Question q1 问题1的测试结果
=====

*** Method not implemented: depthFirstSearch at line 90 of search.py
*** FAIL: Terminated with a string exception.

### Question q1: 0/3 ### 默认dfs没有实现，无测试用例通过

Question q2
=====

*** Method not implemented: breadthFirstSearch at line 95 of search.py
*** FAIL: Terminated with a string exception.
```

修改代码后，再次运行 `python autograder.py`。当测试用例通过，可以看到测试过程：

```
(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).
```

7、 Python 学习与调试

Python 基础需自主学习，后续实验二深度学习继续使用 Python 完成，版本要求为 python3。实验要求的 python 内容不多，主要学习 python 安装、基本的数据类型（list, dict, tuple）、函数、类等基础即可。可参考以下内容：

[CS188 Python/Autograder Tutorial](#)

[python 官方教程中文版](#)

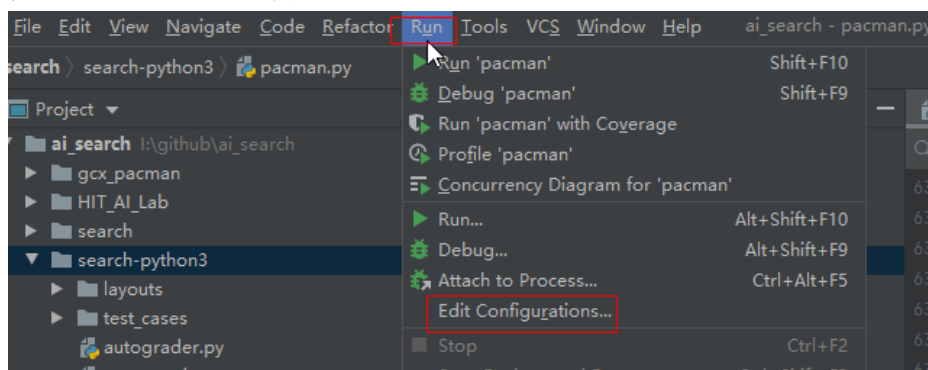
[廖雪峰 python 教程](#)

在提供的框架代码中，项目的入口代码是 `pacman.py` 中的 `runGames()`。

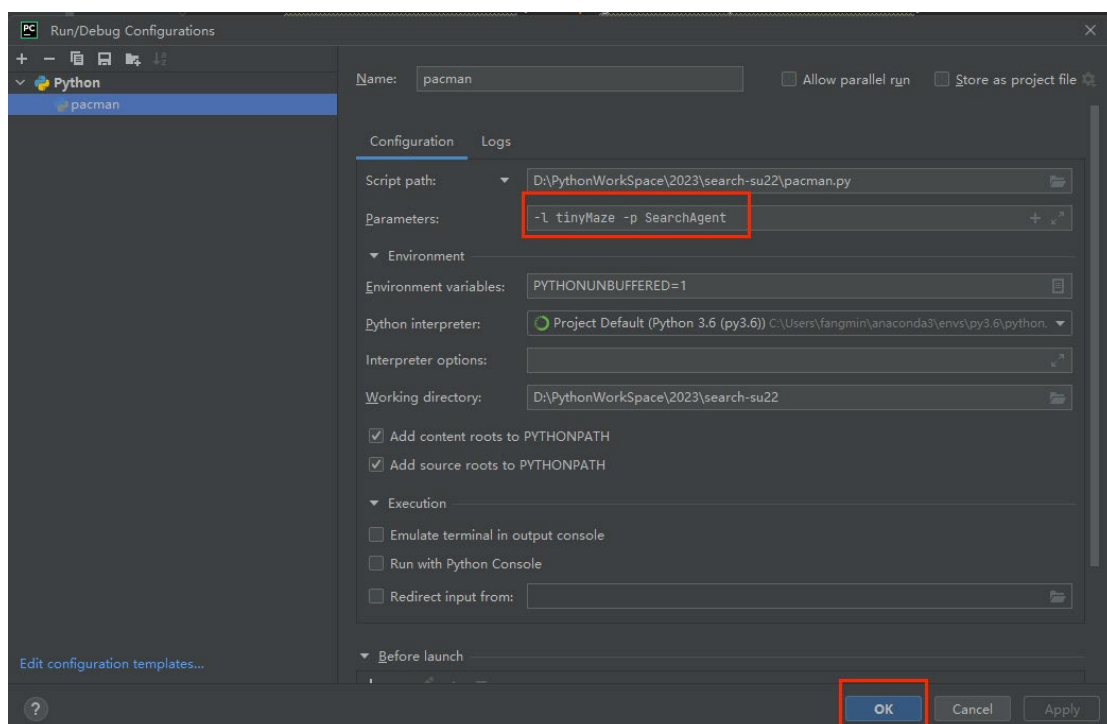
```
if __name__ == '__main__':  
    # The main function called when pacman.py is run  
    # from the command line:  
    > python pacman.py  
    See the usage string for more details.  
    > python pacman.py --help  
    args = readCommand(sys.argv[1:]) # Get game components based on input  
    runGames(**args)  
  
    # import cProfile  
    # cProfile.run("runGames( **args )")  
    pass
```

Python 自带调试工具 `pdb`，类似于 `gdb`。这里介绍下 `pycharm` 中怎么可视化调试 `pacman` ([debugging with pycharm](#))。

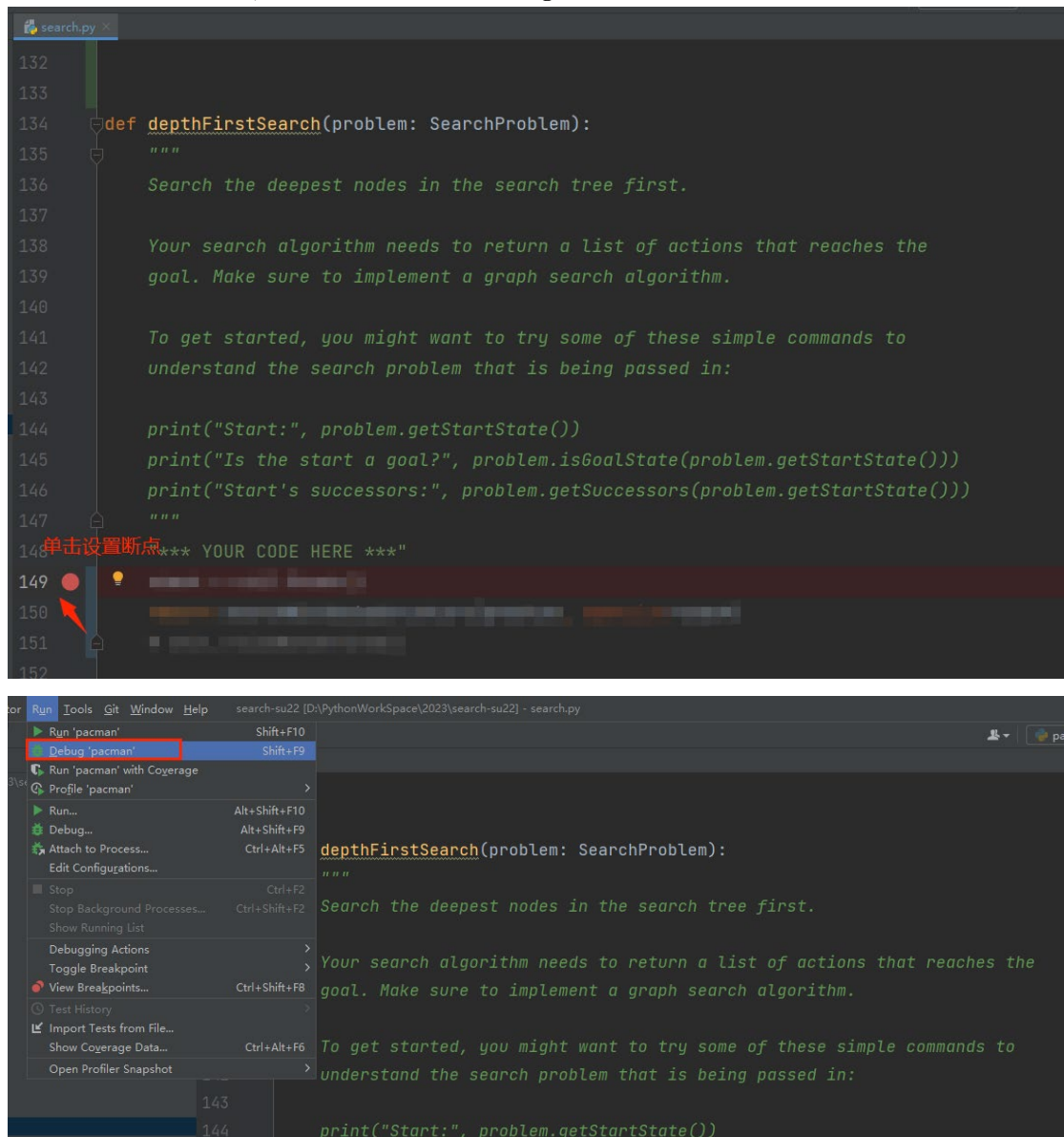
首先打开项目的配置编辑界面。



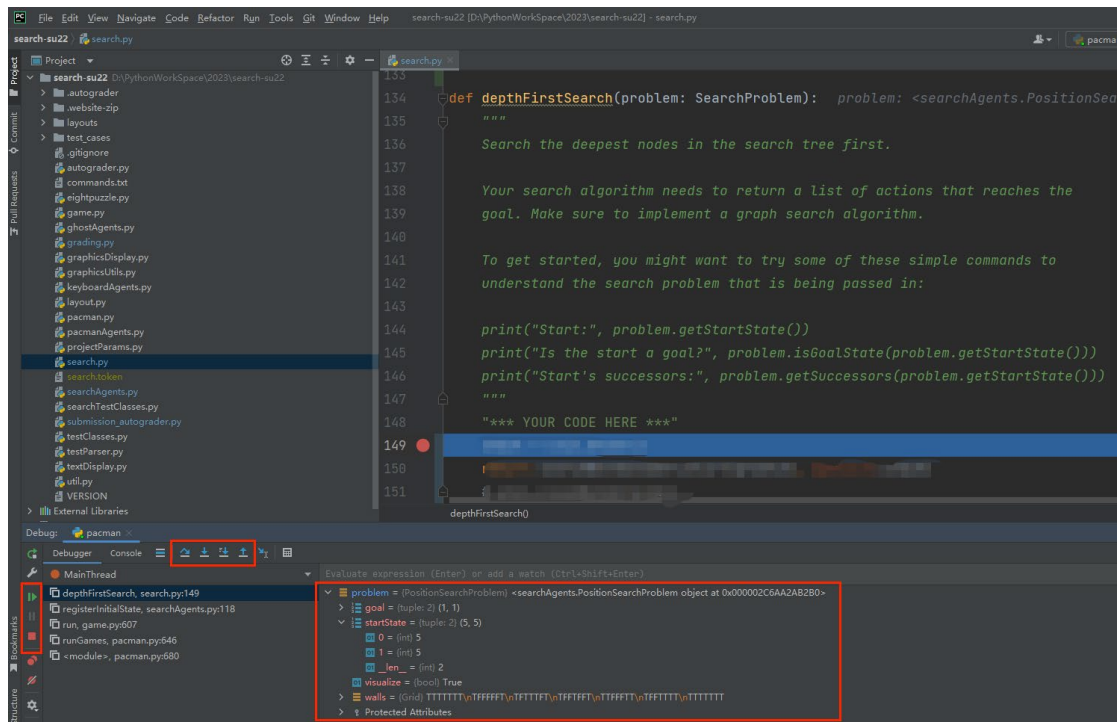
弹出如下界面，输入要调试的选项，比如要测试 `python pacman.py -l tinyMaze -p SearchAgent`，则在 `Parameters` 框中输入 `-l tinyMaze -p SearchAgent`，点击 `ok` 即完成配置。



在合适的位置添加断点，点击 Run→Debug，即可进行调试界面。



程序会自动在第一个断点处停止，并显示当前的变量信息，然后根据需要可以单步调试或者其他操作。



8、实验结果提交

- 提交物：
 - 整个工程（不要仅提交.py 文件）
 - 实验报告（一定要用模板）
 - 打成 zip 包提交
- 作业提交平台：<http://grader.tery.top:8000/#/courses>
- 截止时间：5 月 5 日