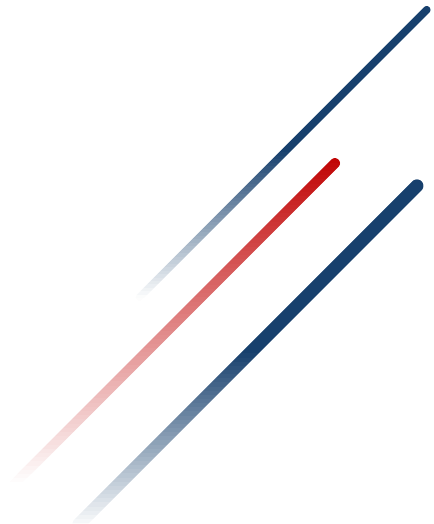


人工智能实验

2022春



实验安排

- ◆ 实验课程共**10**个学时，**2**个实验项目，总成绩为**30**分

实验内容	分值
答辩	实验1（5分）、实验2（5分）
实验1	10分（报告、代码、考勤等）
实验2	10分（报告、代码、考勤等）

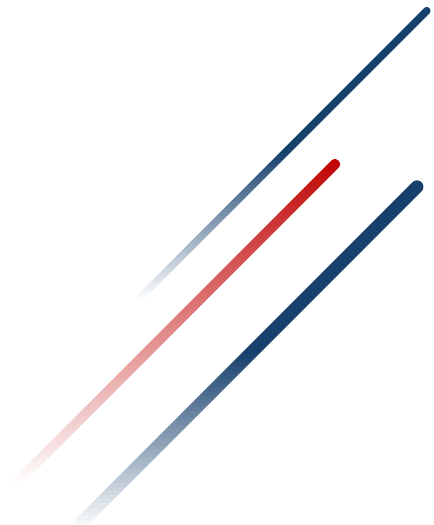
- ◆ 分组完成，每组3或4人
- ◆ 实验课第7~10周，最后一次课每组有5分钟的答辩时间



人工智能

实验1-搜索策略pacman

2022春



实验背景

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可视化游戏界面中加入AI策略。

实验地址如下<https://inst.eecs.berkeley.edu/~cs188/su21/project1/>



对python的掌握情况

- Ⓐ 没学过，完全不会
- Ⓑ 会一点，看的懂代码，没有独立开发过项目
- Ⓒ 较熟练，用python开发过小项目
- Ⓓ 非常熟练，用python开发过很多项目

- ◆ Python语法可以参考“廖雪峰Python教程”
- ◆ 学习重点：python安装、数据类型、函数、模块等基础即可

实验内容

要求使用但不限于课程讲的各种搜索算法，编写一系列吃豆人程序，解决以下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

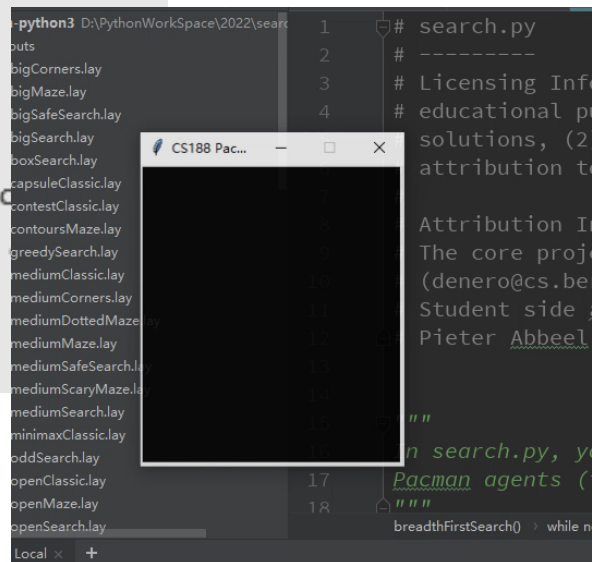
问题4：应用A* 算法找到一个特定位置的豆子

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

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

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)



The screenshot shows a code editor with a file named `search.py` open. The file contains a docstring with licensing information and a comment about attribution. Below the docstring, there is a function definition for `breadthFirstSearch0`. A terminal window titled "CS188 Pacman" is open in the foreground, showing a black screen. The code editor also shows a list of files on the left side, including `bigCorners.lay`, `bigMaze.lay`, `bigSafeSearch.lay`, `bigSearch.lay`, `boxSearch.lay`, `capsuleClassic.lay`, `contestClassic.lay`, `contoursMaze.lay`, `greedySearch.lay`, `mediumClassic.lay`, `mediumCorners.lay`, `mediumDottedMaze.lay`, `mediumMaze.lay`, `mediumSafeSearch.lay`, `mediumScaryMaze.lay`, `mediumSearch.lay`, `minimaxClassic.lay`, `oddSearch.lay`, `openClassic.lay`, `openMaze.lay`, and `openSearch.lay`.

实验内容

要求使用但不限于课程讲的各种搜索算法，编写一系列吃豆人程序，解决以下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 (CornerFinder)

问题6：找到所有的角落——基于A*的角落问题 (CornerFinder)

问题7：吃掉所有的豆子——食物搜索问题 (FoodSearcher)

问题8：次最优搜索——任意食物搜索问题 (AnyFoodSearcher)

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
```


实验内容

要求使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

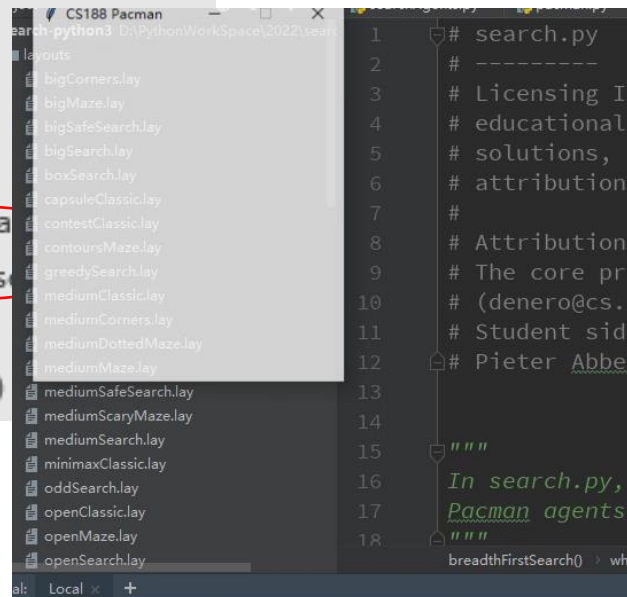
问题4：应用A* 算法找到一个特定位置的豆子

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

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

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)



实验内容

要求使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

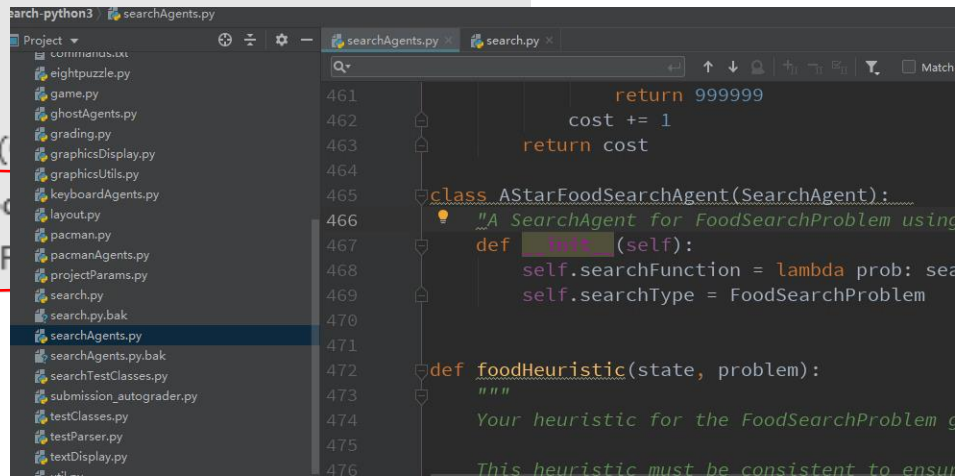
问题4：应用A* 算法找到一个特定位置的豆子

问题5：找到所有的角落——基于BFS的角落问题

问题6：找到所有的角落——基于A*的角落问题

问题7：吃掉所有的豆子——食物搜索问题 (Food Search)

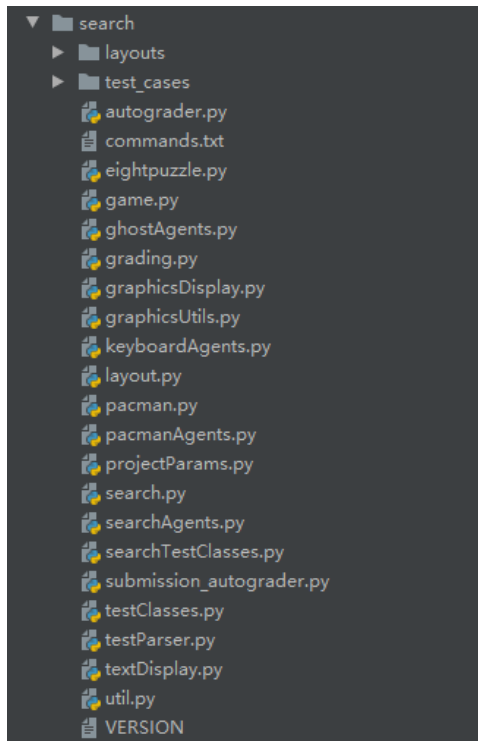
问题8：次最优搜索——任意食物搜索问题 (Any Food Search)



```
search-python3 | searchAgents.py
Project | searchAgents.py | search.py
461 |         return 999999
462 |         cost += 1
463 |         return cost
464 |
465 | class AStarFoodSearchAgent(SearchAgent):
466 |     """A SearchAgent for FoodSearchProblem using
467 |     def __init__(self):
468 |         self.searchFunction = lambda prob: searchFunction
469 |         self.searchType = FoodSearchProblem
470 |
471 |
472 | def foodHeuristic(state, problem):
473 |     """
474 |     Your heuristic for the FoodSearchProblem g
475 |
476 |     This heuristic must be consistent to ensure
```

初始代码

整个项目使用Python开发，有python2和python3两个版本



需要读懂代码的文件

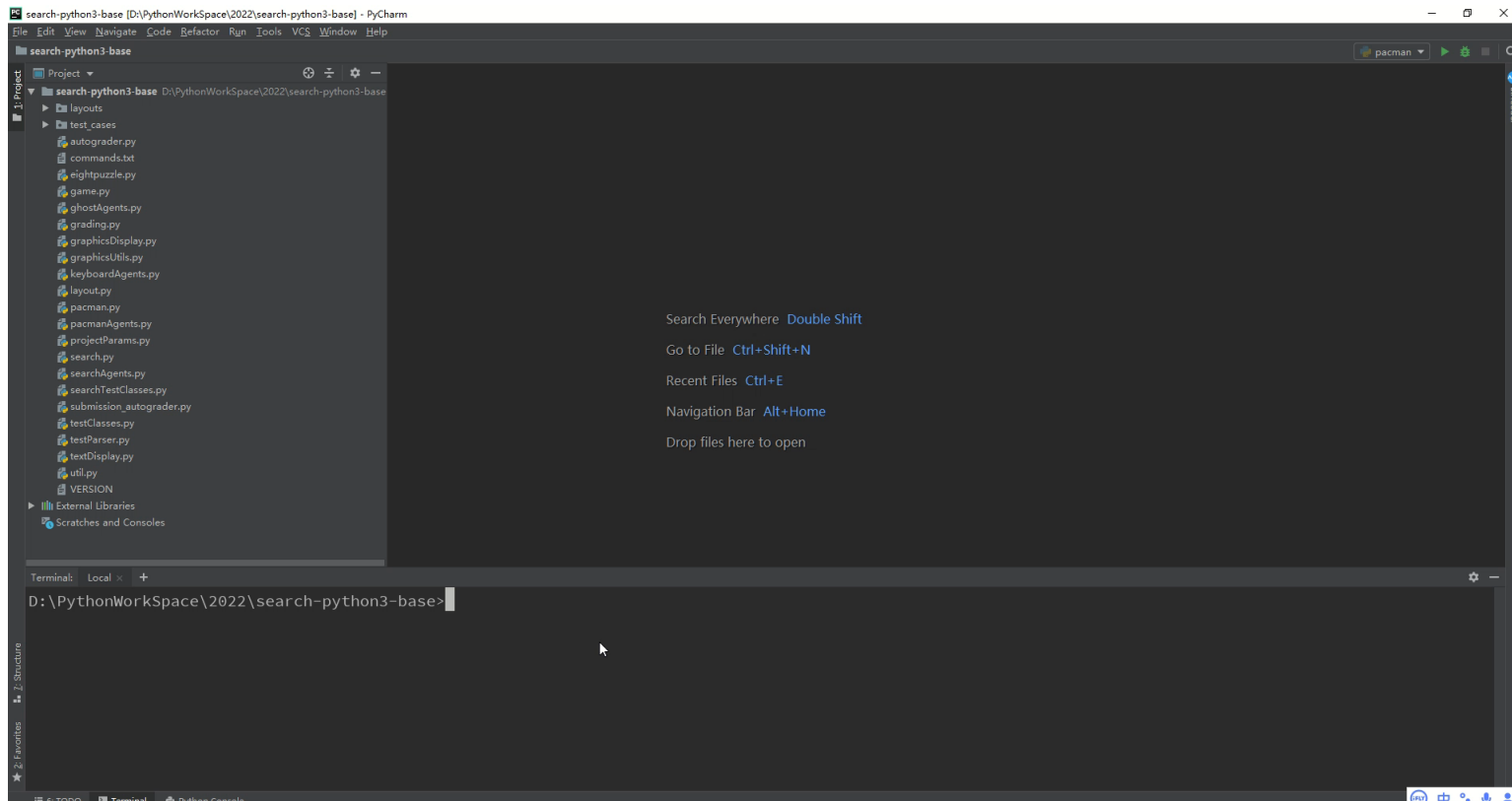
文件	主要功能
pacman.py	吃豆人游戏的主程序
game.py	吃豆人游戏的运行逻辑
util.py	提供一些可用的数据结构，如Stack、Queue、PriorityQueue。这是自动评分系统的兼容性要求。

需要完善代码的文件

文件	主要功能
search.py	待实现的搜索算法
searchAgents.py	待实现的智能体和相关Problem

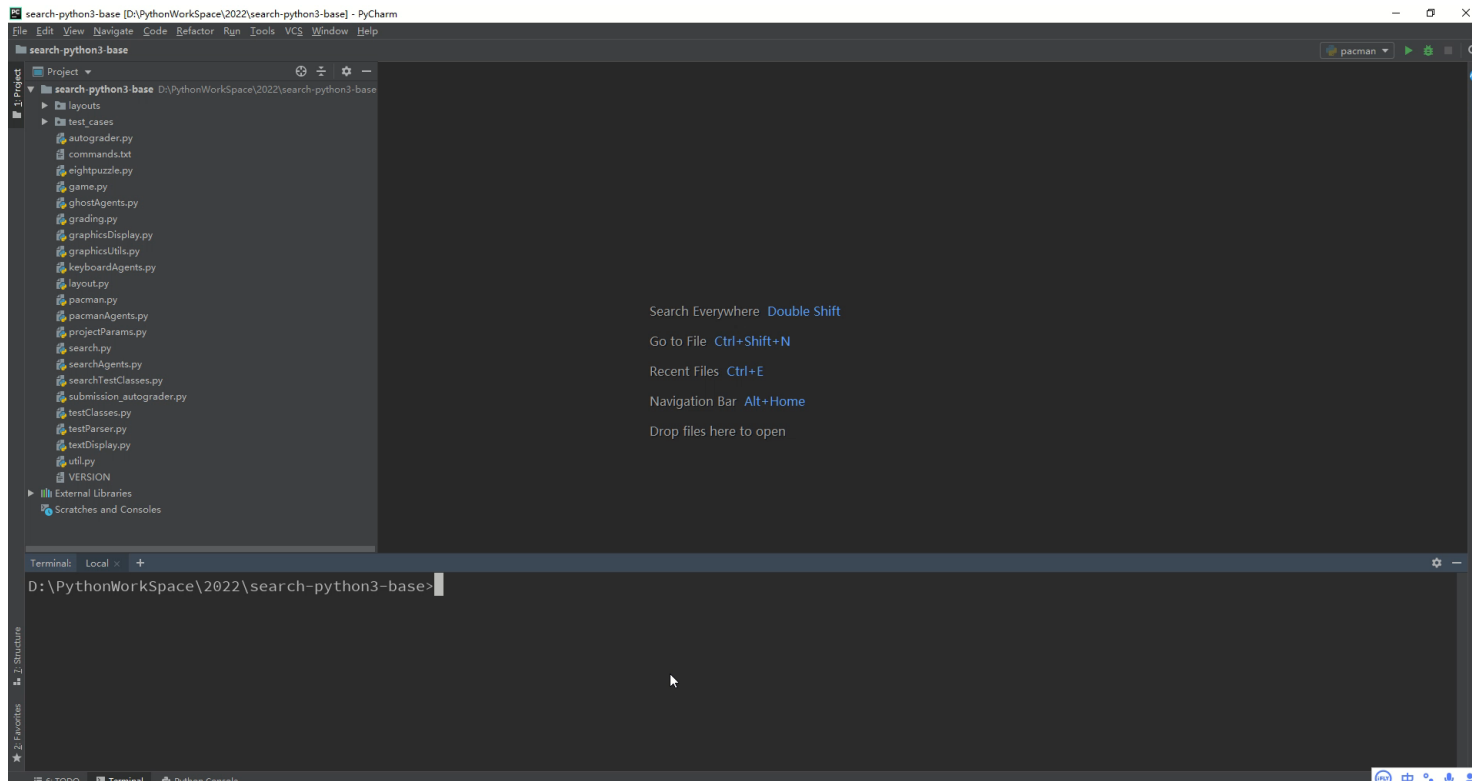
运行和测试代码

python pacman.py



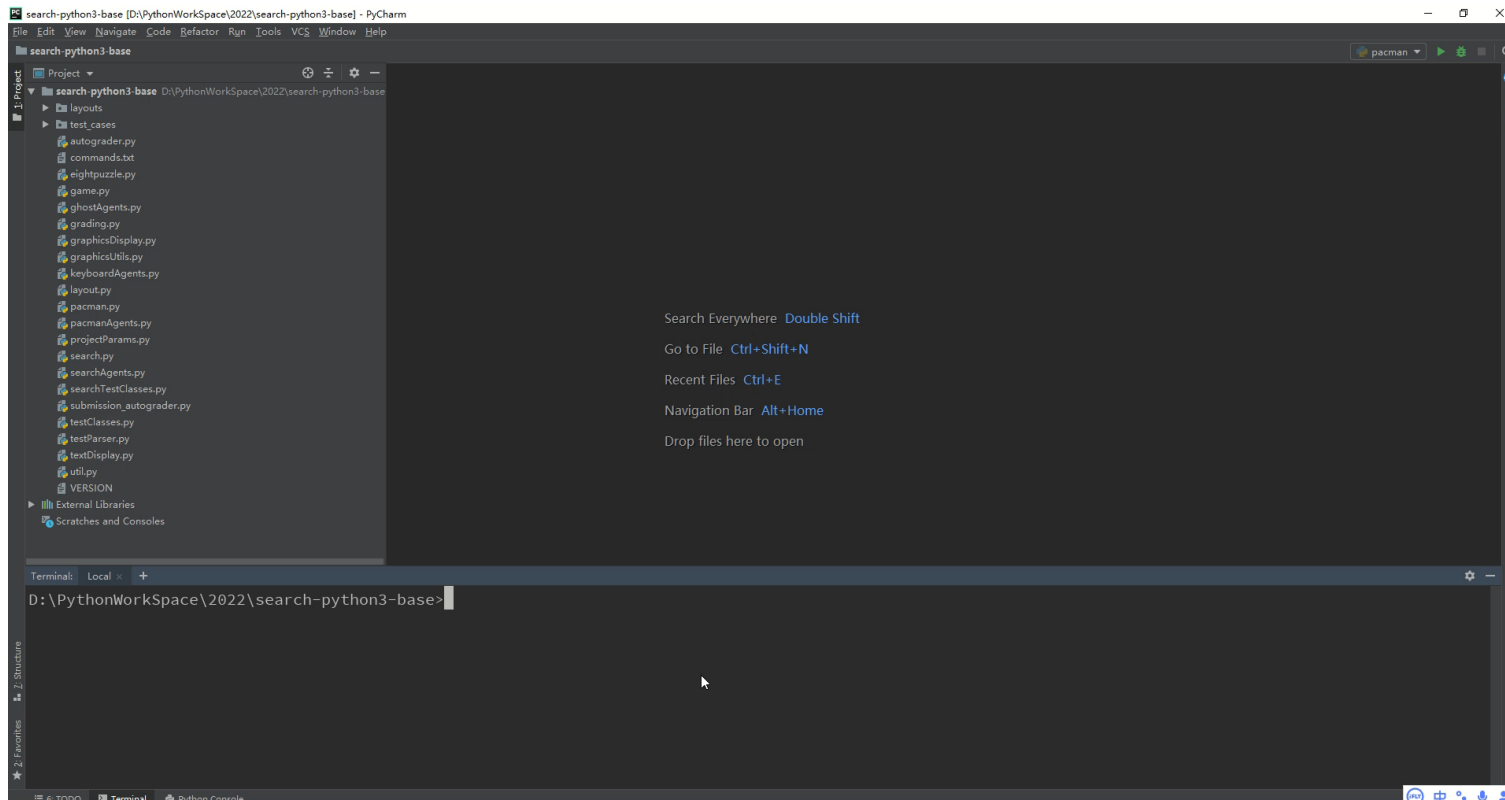
运行和测试代码

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



运行和测试代码

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



测试命令

The image shows a screenshot of an IDE interface. On the left, a project tree for 'search-python3' is visible, with 'commands.txt' highlighted. The main editor window displays the contents of 'commands.txt', which is a list of 22 numbered lines of Python commands for testing the Pacman game. The commands use various flags like --layout, --pacman, -l, -p, -a, and -z to run different search algorithms on different maze layouts.

```
1 python pacman.py
2 python pacman.py --layout testMaze --pacman GoWestAgent
3 python pacman.py --layout tinyMaze --pacman GoWestAgent
4 python pacman.py -h
5 python pacman.py -l tinyMaze -p SearchAgent -a fn=tinyMazeSearch
6 python pacman.py -l tinyMaze -p SearchAgent
7 python pacman.py -l mediumMaze -p SearchAgent
8 python pacman.py -l bigMaze -z .5 -p SearchAgent
9 python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs
10 python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5
11 python eightpuzzle.py
12 python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs
13 python pacman.py -l mediumDottedMaze -p StayEastSearchAgent
14 python pacman.py -l mediumScaryMaze -p StayWestSearchAgent
15 python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic
16 python pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
17 python pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
18 python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5
19 python pacman.py -l testSearch -p AStarFoodSearchAgent
20 python pacman.py -l trickySearch -p AStarFoodSearchAgent
21 python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5
22
```

Terminal: Local x +

常用的命令参数

指定使用的agent类型。默认是1，比1大则是放大，比1则是缩小。

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

地图选项参数，从layouts/目录下加载指定类型地图

传递给agent的参数值。以字符串的形式，如果有多个取值以逗号分隔

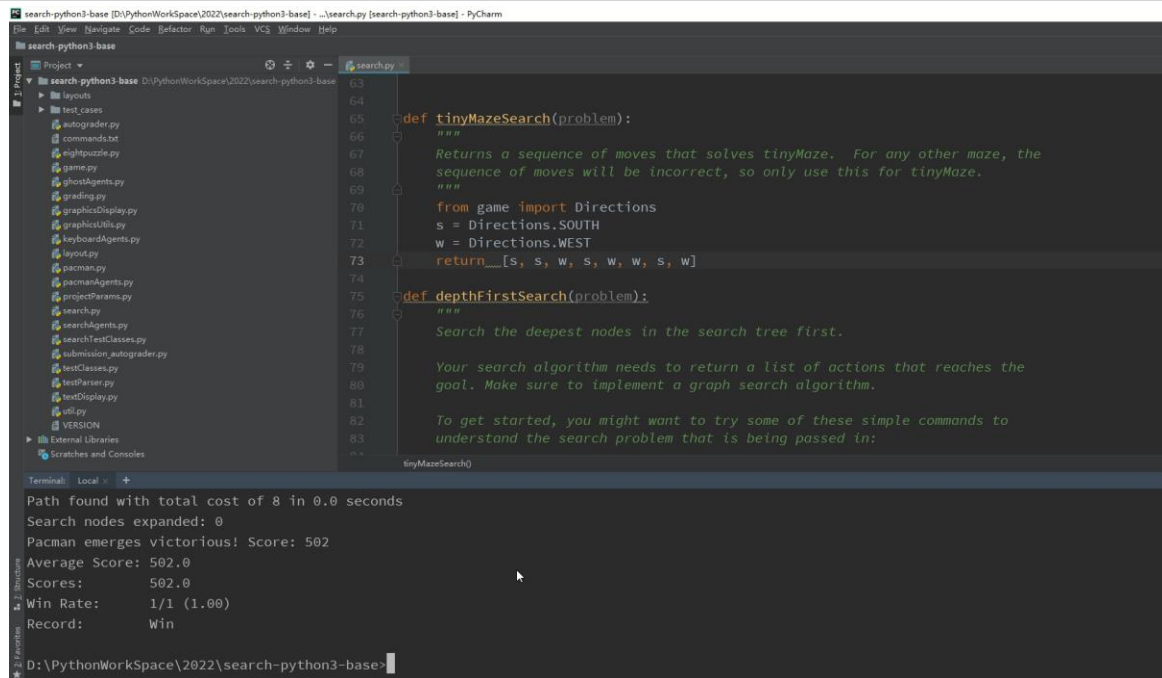


如何开始.....

问题1：应用深度优先算法找到一个特定位置的豆子

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
```



The screenshot shows the PyCharm IDE with the `search.py` file open. The code defines two functions: `tinyMazeSearch` and `depthFirstSearch`. The `tinyMazeSearch` function returns a sequence of moves that solves the tinyMaze. The `depthFirstSearch` function is a placeholder for a depth-first search algorithm.

```
def tinyMazeSearch(problem):  
    """  
    Returns a sequence of moves that solves tinyMaze. For any other maze, the  
    sequence of moves will be incorrect, so only use this for tinyMaze.  
    """  
    from game import Directions  
    s = Directions.SOUTH  
    w = Directions.WEST  
    return [s, s, w, s, w, s, w, s, w]  
  
def depthFirstSearch(problem):  
    """  
    Search the deepest nodes in the search tree first.  
  
    Your search algorithm needs to return a list of actions that reaches the  
    goal. Make sure to implement a graph search algorithm.  
  
    To get started, you might want to try some of these simple commands to  
    understand the search problem that is being passed in:  
    """
```

The terminal output shows the results of running the command:

```
Path found with total cost of 8 in 0.0 seconds  
Search nodes expanded: 0  
Pacman emerges victorious! Score: 502  
Average Score: 502.0  
Scores: 502.0  
Win Rate: 1/1 (1.00)  
Record: Win
```

如何开始.....

问题1：应用深度优先算法找到一个特定位置的豆子

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

Terminal: Local x +

```
D:\PythonWorkspace\2022\search-python3-base>python pacman.py -l tinyMaze -p SearchAgent -a fn=dfs
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
*** Method not implemented: depthFirstSearch at line 90 of D:\PythonWorkspace\2022\search-python3-base\search.py
```



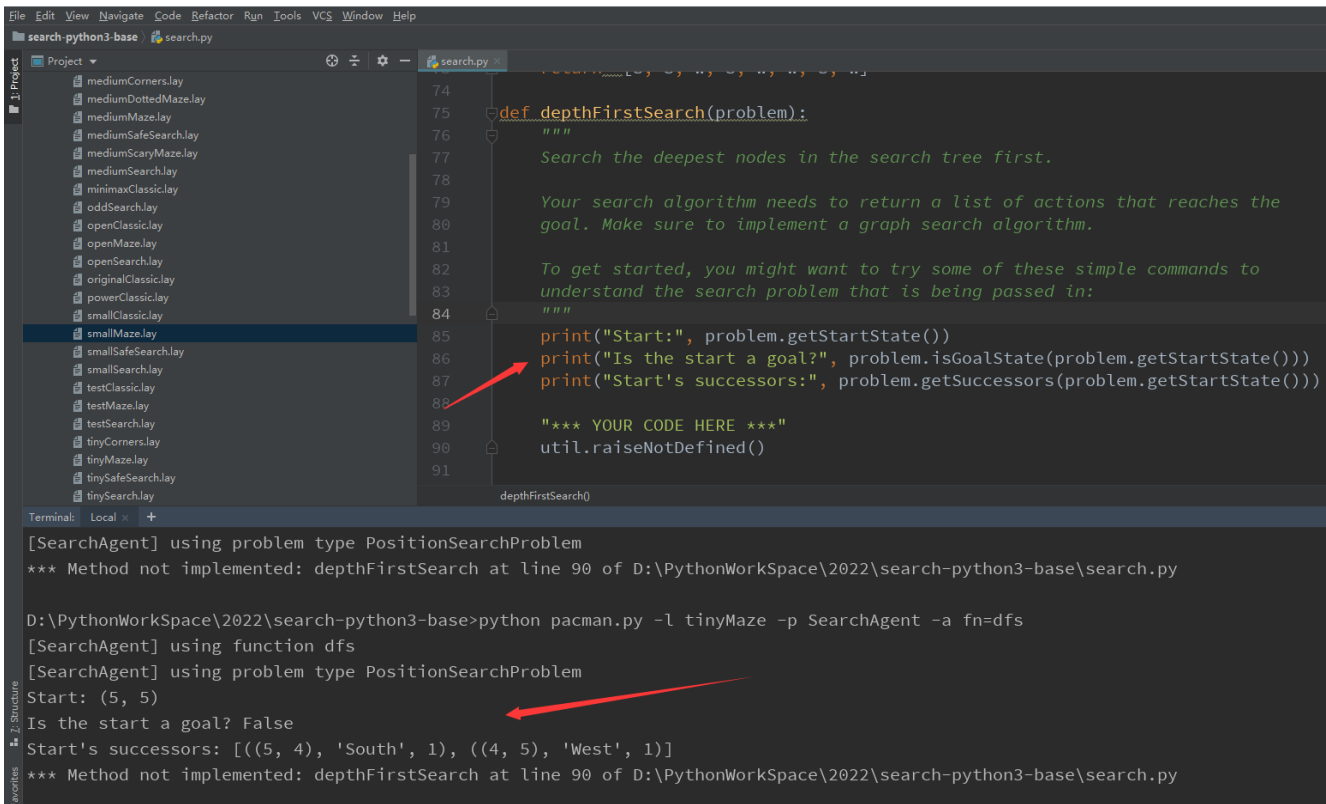
如何开始.....

问题1：应用深度优先算法找到一个特定位置的豆子

```
search.py x
72     w = Directions.WEST
73     return [s, s, w, s, w, w, s, w]
74
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
92 def breadthFirstSearch(problem):
93     """Search the shallowest nodes in the search tree first."""
94     "*** YOUR CODE HERE ***"
95     util.raiseNotDefined()
96
97 def uniformCostSearch(problem):
98     """Search the node of least total cost first."""
99     "*** YOUR CODE HERE ***"
```

如何开始.....

问题1：应用深度优先算法找到一个特定位置的豆子



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
search-python3-base search.py

Project
  mediumCorners.lay
  mediumDottedMaze.lay
  mediumMaze.lay
  mediumSafeSearch.lay
  mediumScaryMaze.lay
  mediumSearch.lay
  minimaxClassic.lay
  oddSearch.lay
  openClassic.lay
  openMaze.lay
  openSearch.lay
  originalClassic.lay
  powerClassic.lay
  smallClassic.lay
  smallMaze.lay
  smallSafeSearch.lay
  smallSearch.lay
  testClassic.lay
  testMaze.lay
  testSearch.lay
  tinyCorners.lay
  tinyMaze.lay
  tinySafeSearch.lay
  tinySearch.lay

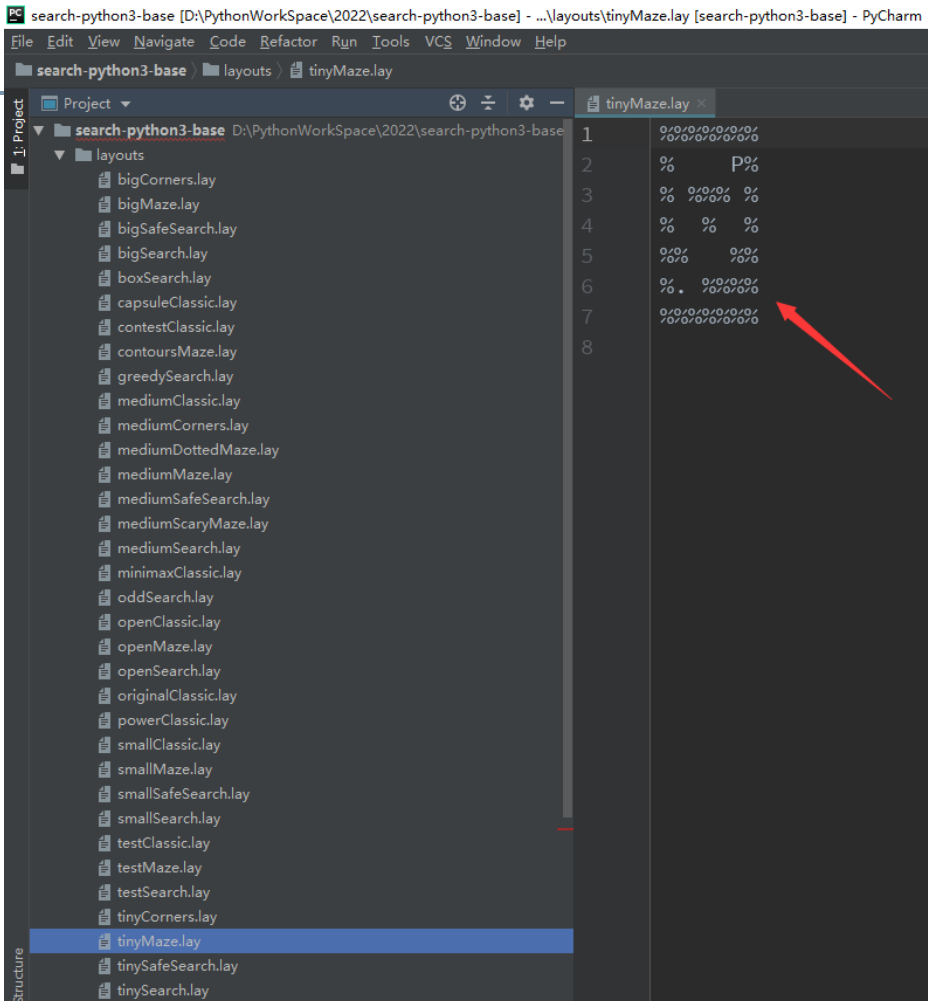
search.py
74
75 def depthFirstSearch(problem):
76     """
77     Search the deepest nodes in the search tree first.
78
79     Your search algorithm needs to return a list of actions that reaches the
80     goal. Make sure to implement a graph search algorithm.
81
82     To get started, you might want 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
depthFirstSearch()

Terminal: Local x +
[SearchAgent] using problem type PositionSearchProblem
*** Method not implemented: depthFirstSearch at line 90 of D:\PythonWorkSpace\2022\search-python3-base\search.py

D:\PythonWorkSpace\2022\search-python3-base>python pacman.py -l tinyMaze -p SearchAgent -a fn=dfs
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Start: (5, 5)
Is the start a goal? False
Start's successors: [((5, 4), 'South', 1), ((4, 5), 'West', 1)]
*** Method not implemented: depthFirstSearch at line 90 of D:\PythonWorkSpace\2022\search-python3-base\search.py
```

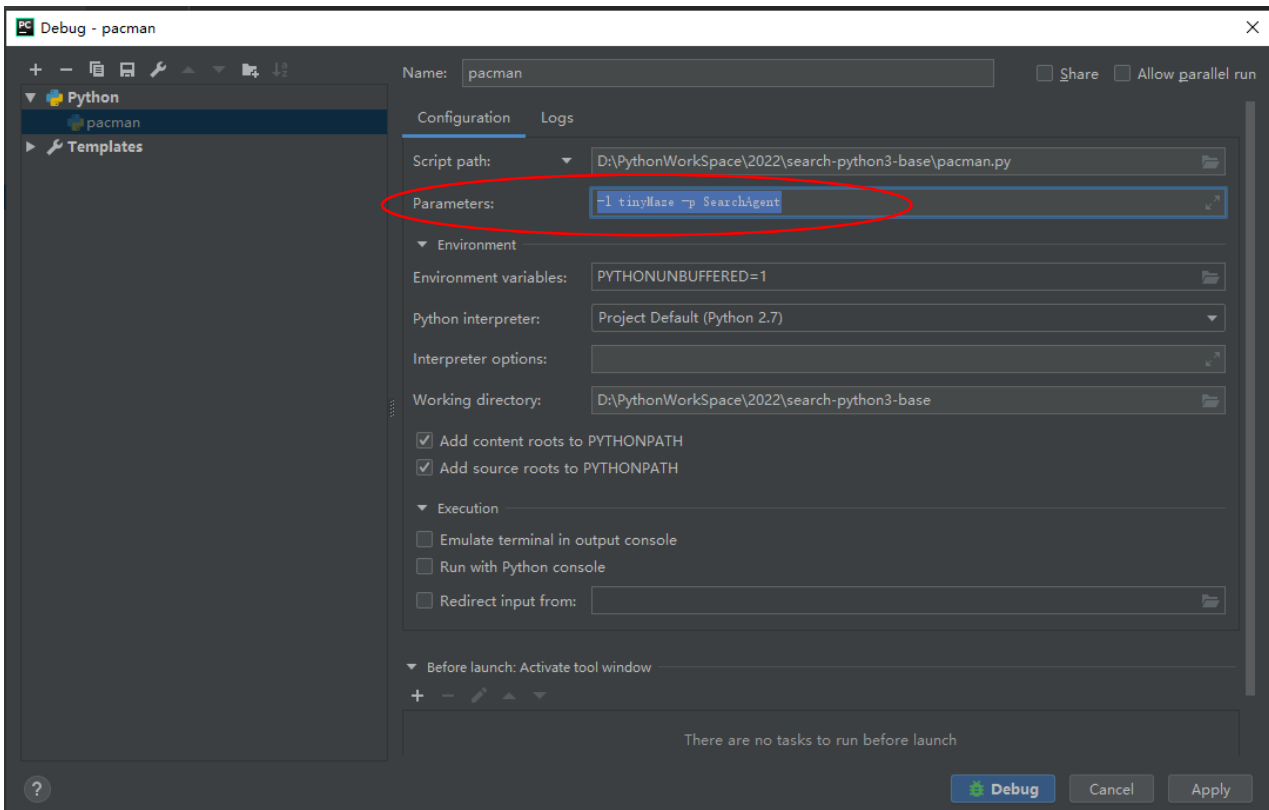
如何开始.....

问题1：应用深度优先算法找到一个特定位置的豆子



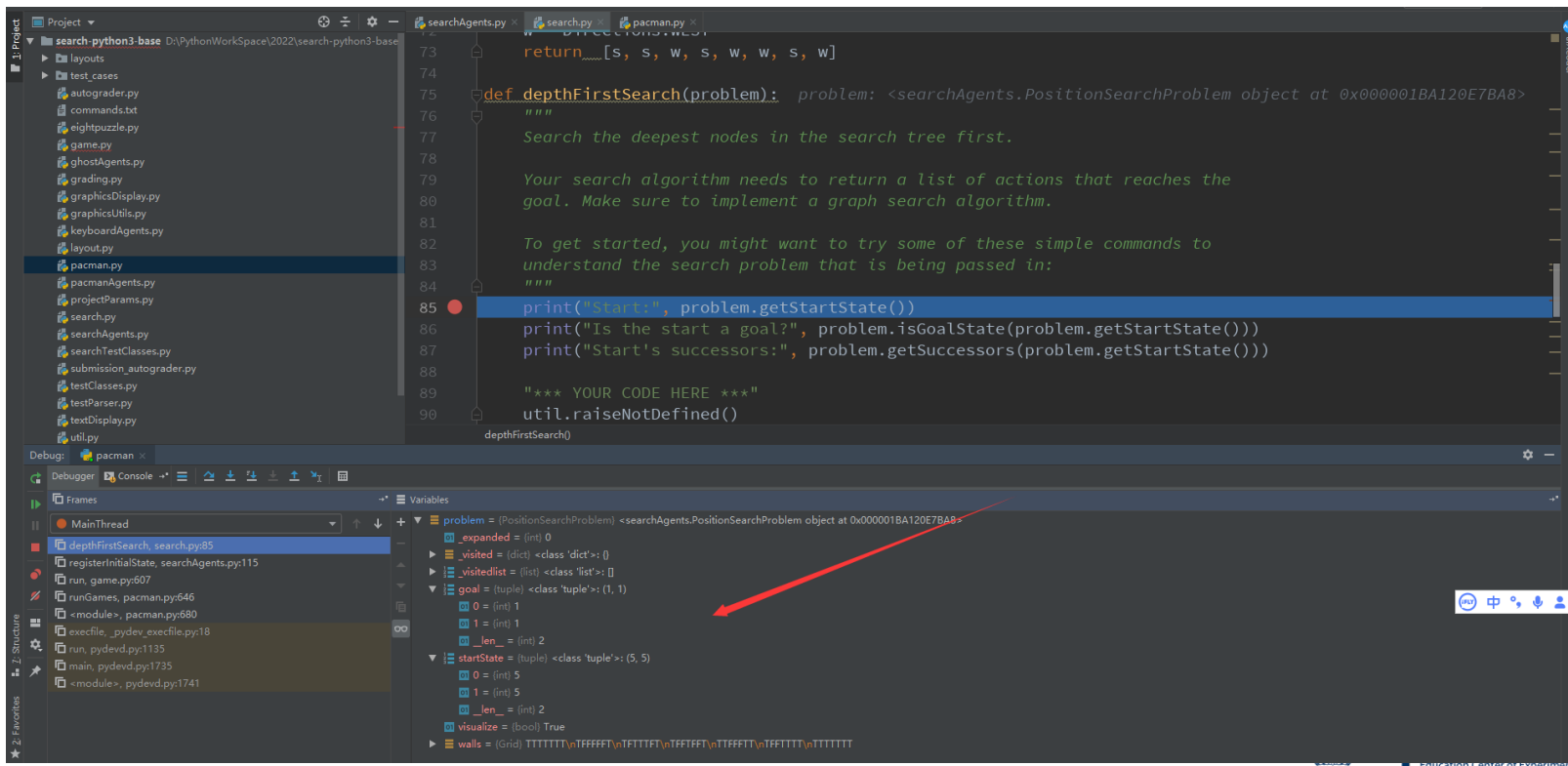
如何开始.....

问题1：应用深度优先算法找到一个特定位置的豆子



如何开始.....

问题1：应用深度优先算法找到一个特定位置的豆子



自动评测

运行python autograder.py

```
D:\PythonWorkSpace\2022\search-python3>python autograder.py
autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  import imp
Starting on 3-29 at 17:10:40

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']
*** PASS: test_cases\q1\pacman_1.test
***      pacman layout:      mediumMaze
```


自动评测

运行python autograder.py

```
Finished at 17:11:00
```

```
Provisional grades
```

```
=====
```

```
Question q1: 3/3
```

```
Question q2: 3/3
```

```
Question q3: 3/3
```

```
Question q4: 3/3
```

```
Question q5: 3/3
```

```
Question q6: 3/3
```

```
Question q7: 5/4
```

```
Question q8: 3/3
```

```
-----
```

```
Total: 26/25
```

■ 7. Structure

```
Your grades are NOT yet registered. To register your grades, make sure  
to follow your instructor's guidelines to receive credit on your project.
```

实验要求

- 所有操作必须合法（比如不能翻越墙壁）。
- 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。



提交方式

- ◆ 每位同学书写自己完成功能的实验报告，小组合并后提交一份完整的实验报告（注意标注每部分的作者名）；
- ◆ 代码以小组为单位提交最终完整版；
- ◆ 作业提交系统：<http://grader.tery.top:8000/#/courses>



同学们
请开始实验吧！