**人工智能实验1-知识表示**

**实验内容：**

参照课程第二部分讲授的知识表示方法完成，包括产生式系统、框架系统、语义网络等（还可以选择其他方法）解决以下问题（不限于此），必要时上网查找有关参考文献。

**猴子摘香蕉问题：**

一个房间里，天花板上挂有一串香蕉，有一只猴子可在房间里任意活动（到处走动，推移箱子，攀登箱子等）。设房间里还有一只可被猴子移动的箱子，且猴子登上箱子时才能摘到香蕉，问猴子在某一状态下（设猴子位置为A，香蕉位置在B，箱子位置为C），如何行动可摘取到香蕉。

**人工智能实验2-搜索策略**

**实验内容：**

实验要求采用且不限于课程第四章内各种搜索算法此编写一系列吃豆人程序解决以下列出的问题1-7 （填充search.py和searchAgents.py这两个文件中的空缺代码），包括到达指定位置以及有效的吃豆等。

**简介：**

基本代码和支持文件可以从search.zip中获取。其中，一些需要参考的文件如下：

## 需要编辑的文件：search.py和searchAgents.py。这两个文件中有一些空缺代码需要大家补充。

## 需要参考的文件：

|  |  |
| --- | --- |
| [pacman.py](https://s3-us-west-2.amazonaws.com/cs188websitecontent/projects/release/search/v1/001/docs/pacman.html) | 吃豆人游戏的程序。 文件包括一个描述”吃豆人”gamestate的类型。 |
| [game.py](https://s3-us-west-2.amazonaws.com/cs188websitecontent/projects/release/search/v1/001/docs/game.html) | 吃豆人游戏的运行逻辑. 文件包括以下类型AgentState, Agent, Direction, and Grid. |
| [util.py](https://s3-us-west-2.amazonaws.com/cs188websitecontent/projects/release/search/v1/001/docs/util.html) | 搜索策略可以用到的数据结构. |

## 可以忽略的支持性文件：graphicsDisplay.py graphicsUtils.py textDisplay.py ghostAgents.py keyboardAgents.py layout.pyautograder.py testParser.py testClasses.py test\_cases/ searchTestClasses.py

**解压缩search.zip，在此目录下，运行以下指令可打开吃豆人游戏。**

***python pacman.py***

**运行*python autograder.py*可以帮助你对自己的程序打分**。

searchAgents.py中最简单的Agent叫做GoWestAgent，一路向西，偶尔能实现目标：

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

但是其不能实现转弯：

*Python pacman.py --layout tinyMaze --pacman GoWestAgent*

如果程序卡死，可通过CTRL-c来终止。

此项目中用到的指令也都储存在commands.txt文件中，可用于复制和粘贴。

## 问题1：应用深度优先算法找到一个特定的位置的豆

首先，运行一下命令测试SearchAgent是不是正常工作：

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

然后，完成完整的通用算法帮助吃豆人规划路线。搜索算法的伪代码见附录。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。

注意：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（不能翻墙）。

注意：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。

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*

注意：因为不同的搜索方法的不同之处仅仅在于open表的排序不同，因此请尽量定义一个通用的搜索算法解决问题1-3。提示：问题1-3的不同之处在于用不同的数据结构对open表进行排序。

## 问题2：宽度优先算法

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：A\* 算法

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，用以下命令测试你得code：

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

## 问题4：找到所有的角落

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*

提示：新的状态只包含吃豆人的位置和角落的状态。

## 问题5：角落问题（启发式）

构建合适的启发函数，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*

## 问题6：吃掉所有的豆子

用尽可能少的步数吃掉所有的豆子。完成searchAgents.py文件中的FoodSearchProblem豆子搜索问题。此问题利用之前A\*算法可以很容易找到解，可用以下命令测试：

*Python pacman.py -l testSearch -p AStarFoodSearchAgent*

构建合适的启发函数，完成searchAgents.py文件中的foodHeuristic豆子搜索（启发式）问题。用以下命令测试你的code：

*Python pacman.py -l trickySearch -p AStarFoodSearchAgent*

## 问题7：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。用以下命令测试你得code：

*Python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5*

# 附录

参考网址<http://ai.berkeley.edu/search.html>的实验内容。

**搜索算法的伪代码**

