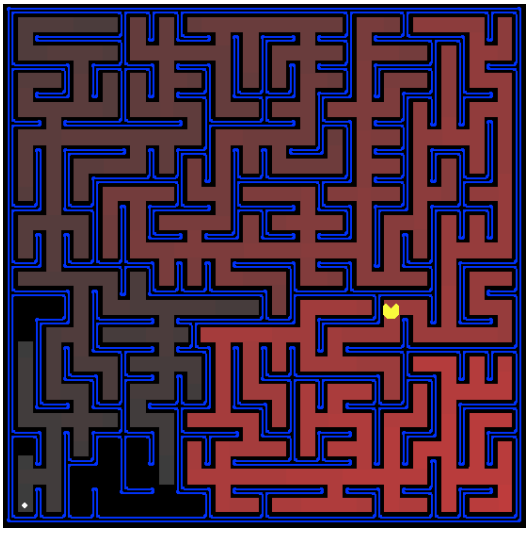
04 Search II



本节实验目标：练习实现有信息搜索策略

报告说明（重要）：本部分“搜索”包含两周内容，本节课为第二周内容，请留意该文档目录中标有“报告应实现任务[分值]”的部分，请在实现后及时保存文件和运行结果截图，在课程报告中提交。

介绍

在本节实验课中，你的Pacman将继续在他的迷宫世界中找到路径，既可以到达特定位置，又可以有效地收集食物。你将构建通用搜索算法并将其应用于 Pacman 场景。

与上节实验课一样，该项目包括一个自动评分器，供你对答案进行评分。这可以在命令终端（路径为该项目文件夹路径）使用以下命令运行：

```
python autograder.py
```

该项目的代码由几个 Python 文件组成，其中一些你需要阅读和理解才能完成作业，而其中一些你可以忽略。你可以在这个 对分易-课程资源-实验课课件，或从机房上电脑桌面（当堂课上） 下载所有代码和支持文件。

你要编辑的文件:	
search.py	你的搜索算法所在地
searchAgents.py	你的搜索Agent所在地
你可能想看一下的文件:	
pacman.py	运行 Pacman 游戏的主文件。这个文件描述了你在这个项目中使用的 Pacman

	GameState 类型。
game.py	Pacman 世界如何运作背后的逻辑。该文件描述了几种类型，如 AgentState、Agent、Direction 和 Grid
util.py	用于实现搜索算法的有用数据结构
你可以忽略的支持文件:	
graphicsDisplay.py	Graphics for Pacman
graphicsUtils.py	Support for Pacman graphics
textDisplay.py	ASCII graphics for Pacman
ghostAgents.py	Agents to control ghosts
keyboardAgents.py	Keyboard interfaces to control Pacman
layout.py	Code for reading layout files and storing their contents
autograder.py	Project autograder
testParser.py	Parses autograder test and solution files
testClasses.py	General autograding test classes
test_cases/	Directory containing the test cases for each question
searchTestClasses.py	Project-specific autograding test classes

请不要更改代码中提供的任何函数或类的名称，否则autograder很出bug。

如果你发现自己卡在某事上，请联系老师、助教或同学寻求帮助（现场、微信或邮件）。

从对分易-课程资源-实验课课件，或从机房电脑 下载代码文件压缩包，解压缩到桌面，在PyCharm中，点击左上角 File → Open... → 查找解压的文件夹（如果是解压到机房桌面，应该在Administrator → Desktop → 对应的文件夹）并切换到目录后，你应该可以通过在命令行中键入以下内容来玩 Pacman 游戏：

```
python pacman.py
```

报告应实现任务[3分]

- **说明：Question 4、5、6 各占1分，截图需要有3张，分别对应 python autograder.py -q q4 / python autograder.py -q q5 / python autograder.py -q q6 的结果截图，其中Question 4 和Question 5 获得评分器给的3分，才算得对应的报告总评的1分； Question 6获得评分器的2分或3分才能得到该小问占的1分。**

Question 4: A* search

任务说明

在 `search.py` 的空函数 `aStarSearch` 中实现 A* 图搜索。 A* 将启发式函数作为参数。启发式函数有两个参数：搜索问题中的状态（主要参数）和问题本身（用于参考信息）。 `search.py` 中的 `nullHeuristic` 启发式函数就是一个简单的例子。

你可以使用曼哈顿距离启发式（在 `searchAgents.py` 中已作为 `manhattanHeuristic` 实现），`aStarSearch`中的heuristic参数是一个类似 `manhattanHeuristic` 的函数，该函数以一个位置和搜索问题为输入，输出一个估计代价值。

针对原始问题测试你的 A* 实现，即找到穿过迷宫到固定位置的路径。

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

你应该看到 A* 找到最佳解决方案的速度略快于 BFS（在我们的实现中扩展了大约 549个 vs. 620 个搜索节点，但优先级的关系可能会使你的数字略有不同）。对于各种搜索策略，`openMaze` 会发生什么？

任务测试

请运行以下命令以查看你的实现是否通过了所有autograder测试用例:

```
python autograder.py -q q4
```

Question 5: Finding All the Corners

只有在更具挑战性的搜索问题中，A* 的真正威力才会显现出来。现在，是时候制定一个新问题并为它设计一个启发式方法了。

任务说明

在 *corner mazes*，有四个点，每个角落一个。**我们新的搜索问题是找到触及所有四个角（迷宫是否真的有食物）的最短路径。**请注意，对于像 `tinyCorners` 这样的迷宫，最短路径并不总是首先到达最近的食物！*Hint*: 通过 `tinyCorners` 的最短路径需要 28 步。

在 `searchAgents.py` 中实现 `CornersProblem` 类（可参考 `PositionSearchProblem` 理解各个函数在做什么）。你将需要选择一个状态表示来编码检测是否已到达所有四个角所需的所有信息。现在，Pacman应该能解决：

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

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

任务提示

要通过以上测试，你需要定义一个不编码无关信息（如鬼的位置、额外食物的位置等）的抽象状态表示。特别是，**不要**使用 Pacman `GameState` 作为搜索状态。如果你这样做，你的代码将非常非常慢（并且也是错误的）。

提示: 你在实现中唯一需要参考的游戏状态的部分是 Pacman 的起始位置和四个角落的位置（已在 `__init__` 中定义，但你可能需要在 `__init__` 定义一个Pacman的初始状态）。

提示: 你在实现 `getSuccessor` 时，需要通过判断每个可能动作执行后是否会碰到墙来筛选出所有后继结点集。successors作为一个列表，其列表元素应为你定义的Pacman状态形式，即跟 `getSuccessor` 输入参数state的形式一样。

我们的 `breadthFirstSearch` 在 `mediumCorners` 上扩展了不到2000 个搜索节点。而启发式（与 A* 搜索一起使用）可以减少所需的搜索节点。

任务测试

请运行以下命令以查看你的实现是否通过了所有autograder测试用例:

```
python autograder.py -q q5
```

Question 6: Corners Problem: Heuristic

任务说明

在 `cornersHeuristic` 中为 `CornersProblem` 实现一个重要的、一致的启发式。请认真查看函数的输入参数定义，输出应为一个从当前状态到目标状态的预估代价值。

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

注: `AStarCornersAgent` 是以下命令的简称:

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

任务提示

可采纳性 *vs.* 一致性: 请记住，启发式函数只是获取搜索状态并返回估计到达一个最近目标的代价。更有效的启发式函数将返回更接近实际目标成本的值。为了保证可采纳性，启发式值必须是到最近目标的实际最短路径代价的下限（并且非负数）。为了保持一致，它还必须认为如果一个动作的成本为 c，那么采取该动作只能导致最多 c 的启发式函数值下降。

请记住，可采纳性不足以保证图搜索的正确性——你需要更强的一致性条件。然而，可采纳的启发式通常也是一致的，特别是如果它们是从松弛问题中得来。因此，通常先从可接受的启发式方法开始。一旦你有一个运行良好的可接受启发式，你也可以检查它是否确实是一致的。确定一致性的唯一方法是使用证明。通过检查你展开的每个节点，其子节点的 f 值是否相等或更低，可以验证启发式的一致性。如果任何节点都违反了此条件，则你的启发式不一致。此外，如果 UCS（启发式为 0 的 A*）和 A* 返回了不同长度的路径，则你的启发式是不一致的。这东西是有点难理解哈哈！

重要的启发式: 不重要的启发式指每次都返回0（UCS）的启发式或计算真实路径代价的启发式。前者不会为你节省任何时间，而后者会使autograder运行很慢。你需要一种减少总计算时间的启发式方法。

任务测试

Autograder的评分机制: 你的启发式必须是重要的非负的一致的启发式才能获得任何分数。确保你的启发式在每个目标状态下都返回 0，并且永远不会返回负值。根据你的启发式所需扩展的节点数，你将被按一下标准评分

扩展的节点数	评分
> 2000	0/3
≤ 2000	1/3
≤ 1600	2/3
≤ 1200	3/3

注意: 如果你的启发式不一致，你将不会获得任何分数。

测试: 请运行以下命令以查看你的实现是否通过了所有autograder测试用例:

```
python autograder.py -q q6
```

报告2 “Search” 提交说明

- 按照第三周和本周的Question 2-6的要求实现代码和获得运行结果截图
- **提交压缩包命名为”姓名_学号_报告序号.zip”** (如“彭振辉_2106666_报告2.zip”)
- 压缩包应**包含内容：**
  - **已实现的完整项目文件夹”Project_2_Search_full“**
    - 其中 search.py 中有Question 2-4要求的函数实现
  - **一个doc或pdf说明文档，上面需要有：**
    - 开头一段说明 “**整体实现参考 + 2-3句简要体会（如教训、思路、拓展应用等）**”，如：-
      - “自行实现。挑战最大的是xxx内容，初始时报了什么错，通过什么方式解决，该部分的实现思路为xxx”
      - “xxx内容参考xxx同学/xxx网址。思考不出算法思路，探究后学习到了什么方法。”
    - **要求实现的五个任务的成功运行截图，说明截图对应任务。**