《人工智能》Project1 搜索：代码介绍

课程代码：CS50020

2025 年 10 月

项目介绍

在本课程项目中，吃豆人智能体将在迷宫世界中寻找路径，躲避幽灵并到达特定的位置并有效地收集食物。本课程项目的目标是构建通用的搜索算法并将其应用于吃豆人场景。

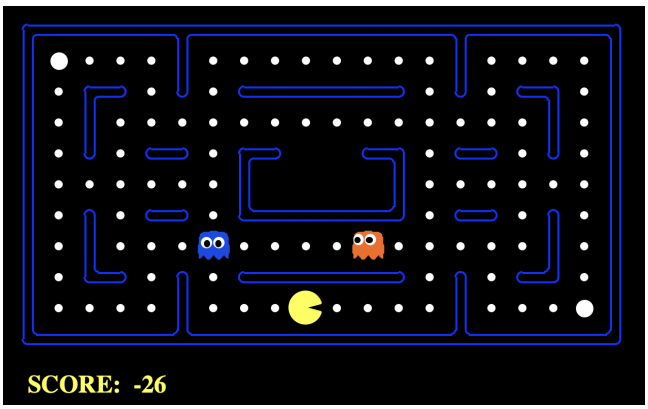

图 1: 吃豆人游戏截图

环境准备

课程项目使用 Python（3.9 ~ 3.11）完成，终端运行 `python -V` 确认已经安装的 python 版本，如果不符合需要下载合适的版本：

1. 官网下载：`https://www.python.org/downloads/`
2. 使用 conda 管理不同的 python 环境：`https://www.anaconda.com/docs/getting-started/miniconda/install`（推荐）

下载并配置好适配 python 编写的 IDE，如 VSCode（推荐）、PyCharm 等，并下载好对应的扩展（如果需要）。

代码介绍

本项目的代码由多个 Python 文件组成，在实现过程中，只需要关注其中的几个关键文件，可以忽视其他的支持文件。所有代码和支持文件均包含在压缩包 pj1-search.zip 中。

需要编辑的文件	
search.py	包含需要编写有信息/无信息搜索算法。
searchAgents.py	包含需要编写的有信息/无信息搜索智能体。
multiAgents.py	包含需要编写的对抗搜索智能体。
可以参考的文件	
pacman.py	运行吃豆人游戏的主文件。该文件描述了一个 GameState 类型，你将在项目中使用它。
game.py	游戏的逻辑文件，定义了一些辅助类：如 AgentState、Agent、Direction 和 Grid。
util.py	提供用于实现搜索算法的常用数据结构。
可忽略的支持文件	
graphicsDisplay.py	吃豆人的图形显示部分。
graphicsUtils.py	为吃豆人图形显示提供支持的工具。
textDisplay.py	吃豆人的 ASCII 文本图形显示。
ghostAgents.py	控制幽灵的智能体。
keyboardAgents.py	用于键盘控制吃豆人的接口。
layout.py	用于读取并存储地图布局的代码。
autograder.py	项目自动评分程序。
testParser.py	用于解析自动评分的测试与答案文件。
testClasses.py	自动评分的通用测试类定义。
test_cases/	存放每个问题测试用例的目录。
layouts/	存放不同地图布局的目录。
searchTestClasses.py	项目中有信息/无信息搜索部分的自动评分的测试类。
multiagentTestClasses.py	项目中对抗搜索搜索部分的自动评分的测试类。

表 1: 项目文件说明

你只需要编辑表格中提及的“需要编辑的文件”，不要改动其他文件的

任何内容。部分辅助类的使用可能需要参考“可以参考的文件”中的定义。在每个问题中，会指明需要编写的部分，请在"*** YOUR CODE HERE ***"后写入相应的内容。

你的代码将会通过自动评分程序（autograder.py）进行技术正确性的评估。请不要修改代码中提供的任何函数或类的名称，否则将会导致自动评分系统出错。不过，分数最终将依据你实现的正确性本身来判定，而不仅仅依赖于自动评分的结果。

主函数和评分函数使用

pacman.py 可以运行吃豆人游戏，使用不同的地图设定和吃豆人算法设定，你可以通过这个函数来简单测试编写的函数是否正常。pacman.py 支持多种选项，每个选项都可以用长格式（例如 –layout）或短格式（例如 -l）来表示。在根目录下，你可以通过以下命令查看所有选项及其默认值：

```
python pacman.py -h
```

autograder.py 是自动评分函数，运行多个测试案例测试算法编写是否正确。你可以所有问题一起评分：

```
python autograde.py
```

也可以单独对某一个问题进行评分测试，根据输出发现问题：

```
python autograde.py -q q1
```

autograder.py 也支持多种选项，可以通过以下命令查看所有选项及其默认值：

```
python autograde.py -h
```

重要的相关类

pacman.GameState ：吃豆人游戏的完整状态表示，包括食物、能量豆、智能体（吃豆人和幽灵，吃豆人 agentIndex=0）的配置信息，以及得分变化。游戏引擎通过 GameState 来追踪游戏的进行状态，而智能体（agents）通过它来进行推理和决策。可以调用的部分主要方法如下：

- `getLegalActions(agentIndex=0)` 返回指定智能体（默认为吃豆人）的合法动作列表。若游戏已结束（胜利或失败），则返回空列表。
- `generateSuccessor(agentIndex, action)` 根据指定智能体执行某个动作后，生成并返回后继状态（一个 *GameState*）。若当前状态为终止状态（胜/负），则抛出异常。会更新得分、移动记录和幽灵计时器。
- `getLegalPacmanActions()` 返回吃豆人的所有合法动作列表。
- `generatePacmanSuccessor(action)` 生成吃豆人执行某动作后的后继状态（一个 *GameState*）。
- `getPacmanState()` / `getPacmanPosition()` 返回吃豆人的状态（一个 *AgentState*）/当前位置 (x,y)。
- `getGhostStates()` / `getGhostPositions()` 返回所有幽灵的状态（一个 *AgentState*）/位置 (x,y)。
- `getNumAgents()` 返回智能体总数（吃豆人 + 幽灵）。

`game.Agent` ：智能体的抽象类，游戏中的编写所有智能体必须定义一个 `getAction` 方法来确定智能体的行动逻辑，接受一个 `GameState` 输入，输出一个动作（`Directions.{North, South, East, West, Stop}`）

`game.Directions` ：游戏中智能体动作表示，包括 North, South, East, West, Stop 四个属性。

`game.AgentState` ：表示游戏中某个智能体（吃豆人或幽灵）的完整状态信息。它包含角色的当前位置、方向、身份（吃豆人或幽灵）、是否处于恐惧状态等。能够通过 `getPosition` 方法返回（x,y）位置，`getDirection` 方法返回返回当前方向。

`game.Grid` 类是一个用于表示吃豆人地图状态的二维布尔数组，其底层由列表（list of lists）实现，支持对每个位置的访问、拷贝与统计操作。坐标系采用左下角为原点的笛卡尔坐标，x 轴水平向右，y 轴垂直向上。其主要属性与方法如下：

- **主要属性:**

  - `width`：网格的宽度（x 方向格数）。
  - `height`：网格的高度（y 方向格数）。
  - `data`：一个二维列表，保存布尔值（`True` 或 `False`）。

- **主要方法:**
  - `copy()`：生成当前网格的副本。
  - `shallowCopy()`：浅拷贝，仅复制外层引用。
  - `count(item=True)`：统计指定值（默认 `True`）的数量。
  - `asList(key=True)`：返回所有满足 `grid[x][y] == key` 的坐标列表。

`search.SearchProblem` ：一个搜索问题的抽象类，定义了搜索问题的基本结构，为所有搜索算法提供统一接口。其核心方法包括：

- `getStartState()`：返回起始状态；
- `isGoalState(SearchState)`：判断给定状态是否为目标状态；
- `getSuccessors(SearchState)`：返回当前状态的所有后继状态、对应动作及代价，形式为三元组 `(successor, action, stepCost)`；
- `getCostOfActions(actions)`：输入动作序列，计算该序列的总路径代价。

其他 在 `game.py` 和 `pacman.py` 中有这些类全部的详细定义和用法，还有其他可能相关类的定义，推荐结合注释指引阅读。

常用数据结构

在 `utils.py` 中实现了搜索算法中常用的数据结构,包括 `Stack`,`Queue`,`PriorityQueue`，可以根据需求阅读和调用。

注意

1. 请仔细阅读注释，根据注释的引导补全代码。
2. 对于代码中不确定的输入输出值类型，多用 print 输出或 pdb 调试来进行确认。