《数据结构与算法》

期末大作业

Pac-Man 游戏说明

中山大学

2021 年 11 月

目录

1 期末大作业简要介绍 3

2 游戏规则 4

2.1 计分 4

2.2 获胜 4

2.3 能量胶囊 4

2.4 环境观测 5

2.5 计算时间 5

3 游戏文件基本介绍 6

3.1 关键文件 6

3.2 支持文件（推荐阅读，请勿修改） 6

3.3 运行游戏 6

4 小组的工作 8

4.1 函数入口 8

4.2 智能体的设计 8

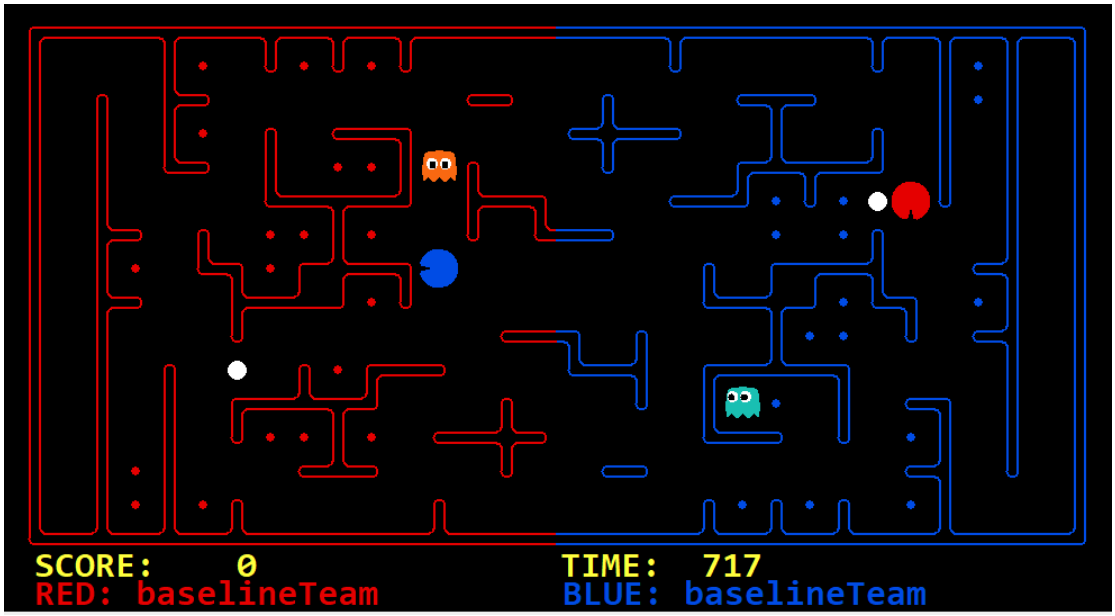
4.2.1 距离计算 8

4.2.2 CaptureAgent 工具 8

1 期末大作业简要介绍

本章将简要介绍本学期数据结构与算法课程的期末大作业内容。

本作业要求以小组为单位，结合课程学习的内容，使用 Python 语言（需要注意的是，**本项目的 Python 版本为 2.7**）编写算法，使得提交的算法可以顺利进行 Pacman 游戏。本作业已经内置了对战模块，不同的算法可以通过相互对战取得对应的竞赛排名。



上图是 PacMan 的游戏界面。在 PacMan 游戏中，像迷宫一样的地图被左右分为对称的红蓝两半，其上随机分布着红色和蓝色的小点作为 PacMan 的食物。红色方和蓝色方各控制两个智能体（总计四个智能体）。

以红色方为例，游戏的基本准则为：联合控制己方的两个智能体尝试吃到蓝色的食物并保护红色食物。当智能体越过地图中线进入蓝半部地图时会变为红色 PacMan（如图中最右侧的智能体），可以吃蓝色食物。当智能体留在红半部的地图时角色为 Ghost（如图中左上方的智能体），可以攻击蓝色方的 PacMan 以保护己方红色食物。更详细的游戏规则请参看第二章。

2 游戏规则

2.1 计分

当PacMan吃到对方的食物时，这些食物点会从地图上移除并储存在PacMan体内。PacMan 需要将体内的食物带回自己的领地（相同颜色的半区）每带回一粒食物即可获得一分。

红色方得分为正，蓝色方得分为负，两队的得分之和会在游戏界面的 score 处显示。例如 score = 1 表示此时红色方领先蓝色方 1 分。

需要注意，如果 PacMan 在返回领地的过程中被 Ghost 吃掉，它会爆炸并将体内的食物点返回到死亡地点附近。被吃掉的 PacMan 会在起始位置作为 Ghost 重生。吃掉对手的 PacMan 不会得分。

2.2 获胜

整场游戏被限制为 300（可修改为更大或更小的值）个回合，在一个回合中各方均可控制一次自己的每个智能体（移动或呆在原地都算是一次控制）。当达到此移动限制时，运送更多食物到己方半区的队伍获得胜利。如果游戏界面中的 score = 0，则记录为平局

如果在达到移动限制之前，某支队伍已经率先运送回了绝大多数（K 个）食物点，则游戏直接结束该队取胜。在本游戏中 K = N-2，其中 N 表示食物点总数。

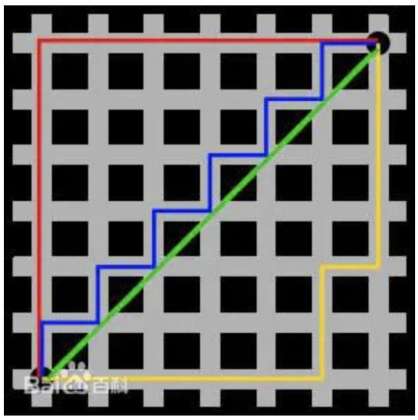
2.3 能量胶囊

地图中左右半区分别存在相同数量的能量胶囊（以较大的白色点显示）。如果 PacMan 吃到了能量胶囊，则对手 Ghost 状态的智能体会在接下来的 40 个回合中变得“害怕”。害怕的 Ghost 如果遭遇了对方的 PacMan 会被吃掉并在起始点重生，重生的 Ghost 将不再害怕。

2.4 环境观测

当对方的智能体位于己方智能体 5 个**曼哈顿距离（两点在南北方向上的距离加上在东西方向上的距离）**内时，己方智能体才能获取对方智能体确切的配置（如当前的位置坐标和方向）。此外，对于尚不能观测到的对方智能体，己方智

能体可以获得一个带有噪声的距离信息，以大致确定对方智能体的位置。



上图中红线代表曼哈顿距离，绿色代表欧氏距离，也就是直线距离，而蓝色和黄色代表等价的曼哈顿距离。

2.5 计算时间

每个智能体有最多 1 秒的时间返回其采取的动作，超出 1 秒的返回会引发警告。累计 3 次警告或者单次返回时间超出 3 秒会终止游戏，判定该队伍弃权。**所以你需要注意算法的运行时间。**

3　游戏文件基本介绍

3.1 关键文件

- capture.py: 这是本地运行游戏的主文件。文件中的 GameState 类提供了许多获得当前游戏的状态信息（包括食物点、能量胶囊、智能体配置信息等）的函数。该文件还描述了游戏的运行逻辑。
- captureAgents.py: 文件包含了基本的智能体类 CaptureAgent，推荐继承它并根据需要重构某些函数。
- baselineTeam.py: 文件包含了两个基本的攻击型和防御型智能体。尽管这两个智能体远非最优，但这个示例文件可以帮助你快速理解游戏。
- myTeam.py: 这是你唯一需要提交的文件，文件中需要包含你定义的智能体以参与比赛。

3.2 支持文件（推荐阅读，请勿修改）

- game.py: 文件描述了 PacMan 游戏运作的逻辑，比如智能体的动作，配置方式，地图的坐标表示等。
- distanceCalculator.py: 提供了计算地图两点之间的最短路径的方法。
- util.py: 提供了许多可选的数据结构，以帮助实现各种搜索算法。

3.3 运行游戏

默认情况下，可以使用以下命令运行游戏：

```
python capture.py
```

该命令会默认根据 baselineTeam.py 创建红蓝双方的智能体

你也可以通过加入参数指定红蓝两队的智能体形式，如以下的命令：

```
python capture.py -r baselineTeam -b myTeam
python capture.py -r baselineTeam -b baselineTeam
```

如果你想亲自玩 PacMan 游戏，以下命令允许你以键盘方向键操控红色方的一个智能体：

```
python capture.py --key0
```

你还可以设置随机种子，换一幅地图进行游玩，例如以下的命令：

```
python capture.py -l RANDOM23
```

或

python capture.py -l RANDOM5

你还可以用以下命令设置游戏最大回合数的限制：

python capture.py -i N

其中 N = K * 4，其中 K 为最大回合数，例如本游戏默认 N = 1200，最大回合数为 300

更多参数设置参考 capture.py 文件或使用以下命令：

python capture.py –help

注意，请务必确保你提交的代码作为红、蓝两队都能顺利进行比赛。

4 小组的工作

4.1 函数入口

project 以 python 2.7 为标准，函数对外接口为 myTeam 中的 createTeam，传入参数中的 first ='xx', second = 'xx' 即为第一个智能体和第二个智能体所使用的算法；

建议编写代码时有规范的格式和合理的注释，以便于小组的报告撰写;

对规则的最终解释权归本游戏技术组所有。违规代码可能被取消参赛资格;

4.2 智能体的设计

在本 Project 中，同学们需要在进攻与防守之间进行权衡，并在游戏环境中有效地同时扮演 ghost 和 Pacman 的角色。下面介绍同学们在设计智能体时候可能需要用到的在 captureAgent 中返回数据的函数以及距离计算函数。

4.2.1 距离计算

关于游戏中点与点的计算可以通过 distanceCalculator.py 文件来返回两点间的最短路径距离。

4.2.2 CaptureAgent 工具

为了更好地让同学们获取游戏数据以便做出更好的决策选择，在 captureAgent.py 中定义了部分函数及返回的类型。

- def getFood(self, gameState):

#返回玩家想吃的食物。数据以矩阵的形式返回，其中 m[x][y]=True 表示在该位置中有你可以吃的食物，也就是豆子。

- def getFoodYouAreDefending(self, gameState):

#返还你要保护的食物（也就是对方智能体想要吃的豆子）。同样数据以矩阵的形式返回，其中 m[x][y]=True 表示 (x,y) 处有你的对手可以吃的豆子。

- def getOpponents(self, gameState):

#返回对手的 agent 的编号。返回的数据为对手 agent 编号的列表（例如，红色方可能是[1,3]）。

- def getTeam(self, gameState):

#返回自己团队的 agent 编号。同样，返回的数据为自己 agent 编号的列表（例如，蓝色方可能是[1,3]）。

- def getScore(self, gameState):

#以数字形式返回你与对方分数的差距，该数字是您的得分与对手得分之间的差值。如果你输了，这个数字就是负数。(注意，这里要跟 UI 界面的 score 得分区分开，UI 界面中 score 为正数表示红色方领先，否则为蓝色方领先，这里返回的 score 为正是自己领先，否则对方领先)

- def getMazeDistance(self, pos1, pos2):

#返回两点之间的距离；这些是使用提供的 distancer 对象进行计算的。如果 distancer.getMazeDistances()已被调用，则迷宫里的距离可以计算。 否则，这只会返回曼哈顿距离。

- def getPreviousObservation(self):

#返回与 agent 看到的最后一个状态对应的 GameState 对象（即 agent 移动时上一次观察到的游戏状态-其中可能不完全包括对手的所有 agent 位置）。

- def getCurrentObservation(self):

#返回当前 agent 观察的 GameState 对象（当前观察到的游戏状态-其中可能不完全包括对手的所有 agent 位置）。