2024春季学期《数据结构与算法》期末大作业

Pac-Man游戏简介与说明

中山大学电子与信息工程学院

教学班号: 202325995

课程：数据结构与算法

2024年05月10日

1. 期末大作业简要介绍

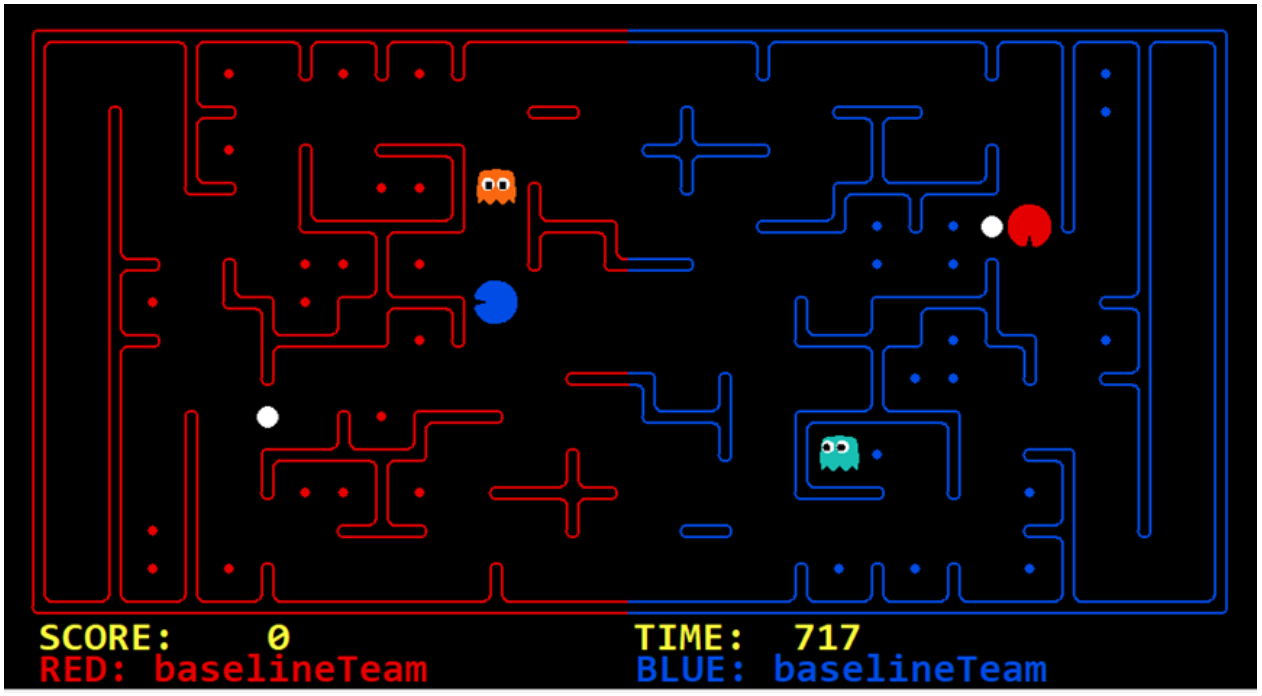
本章将简要介绍2024春季学期中山大学电子与信息工程学院数据结构与算法 (教学班号: 202325995) 课程的期末大作业内容。

- v1.0：将本项目从Py2修改为Py3，并做少量修改修改相关描述，更新对应文档介绍
- 注意，本项目的介绍文档与代码可能会更新，请以最新版为准

1.1 本次期末大作业简介

本作业要求以**小组**为单位，结合课程学习的内容，使用Python3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。

> - 为避免语法糖的不同，版本指定为 `Python3.8` 或 `Python3.9`



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。

- 本项目基于Python标准库修改而来，理论上无需安装任何依赖与库，若运行出问题，可找助教咨询或协助修改

2. 游戏规则

2.1 计分规则

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 对局解析与胜负评判标准

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。

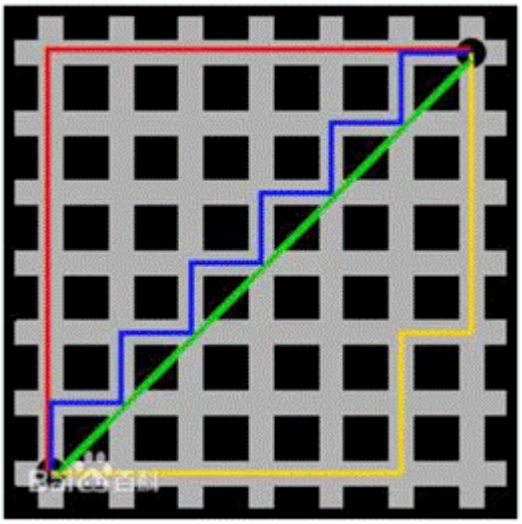
如果在达到移动限制之前，某支队伍已经率先运送回了绝大多数（`K` 个）食物点，则游戏直接结束该队取胜。在本游戏中 `K = N-2`，其中 `N` 表示食物点总数。

2.3 特殊道具 -- 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 计算时间限制

每个智能体有最多 `0.5` 秒的时间返回其采取的动作，超出 `0.5` 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，测试机子的性能不一定比你的机器性能要好。**

具体时间限制准则请见 `capture.py` 的 `CaptureRules` 类定义，不匹配的内容后续会修改为上述限制数值。

- 时间限制测试需要在运行时加上 `-c` 标志，如 `python capture.py -c`

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。
- `algo/myTeam.py`： 本文件的 `DummyAgent` 为定义的智能体样例。请借鉴本文件与 `captureAgents.py` 编写您的智能体，并将编写好的文件放置在 `./algo` 文件夹（相对路径）下，以参与比赛。具体而言，假设提交的文件名以小组编号命名，那么第5小组的算法主文件请命名为 `./algo/5.py` ，若有额外的文件（如辅助函数、神经网络权重），请建立并放置到 `./algo/5/` 文件夹，如 `model.pth` 文件位置为 `./algo/5/model.pth` 。

3.2 游戏运行支持文件

- 推荐阅读这些文件，算法的实现可调用这些文件中的函数，但请自行解决import路径问题；除debug需求外，请勿修改这些文件，最终验证时候不会同步这些修改
- `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 游戏运行

在进入pacman文件夹后，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 algo/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：

```
# 注意，由于算法运行时间可能较快，建议人工试玩的时候在 game.py的run函数约712行的 agentIndex = ( agentIndex + 1 ) % numAgents语句的下一段加入 time.sleep(0.05) 以确保游戏运行人类可反应
python capture.py --keys0 # 将控制红色的吃豆人
```

默认为随机生成地图游玩（100个种子），若要指定游戏地图/固定游戏地图

1. 第一种，指定lay文件所在文件夹，如 `layout/eval`

```
python capture.py --layout "layout/eval"
```

2. 第二种，自行修改 `capture.py` 文件约882行判别分支，锁定lay文件或seed

你还可以用以下命令设置游戏局数

```
python capture.py -n 3
# or
python capture.py --numGames 3
```

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
# or
python capture.py --time N
```

其中$N = K \cdot 4$，其中K为最大回合数，例如本游戏默认$N = 1200$，$K = 300$。这里介绍为什么回合数是K而不是显示的时间N，因为游戏中，每个事件步只有一个pacman运行（可以思考以下同时运行会有什么问题呢？），故4个时间步为一轮，4个pacman故$K = N/4$。

更多参数设置参考 `capture.py` 文件或使用以下命令：

```
python capture.py -help

Options:
  -h, --help            帮助
  -r RED, --red=RED     红队算法
  -b BLUE, --blue=BLUE  蓝队算法
  --red-name=RED_NAME   不要修改这个
  --blue-name=BLUE_NAME 不要修改这个
  --keys0               让agent 0 (first red player)人为操控
  --keys1               让agent 1 (second red player)人为操控
  --keys2               让agent 2 (first blue player)人为操控
  --keys3               让agent 3 (second blue player)人为操控
  -l LAYOUT, --layout=LAYOUT 默认随机地图
  -t, --textgraphics    仅在命令行显示地图
  -q, --quiet           只运行（可用于采样等快速操作）
```

```
  -Q, --super-quiet       与上面一样，加强版
  -z ZOOM, --zoom=ZOOM    放缩地图
  -i TIME, --time=TIME    时间步限制（/4就是回合数）
  -n NUMGAMES, --numGames=NUMGAMES 游戏局数
  -f, --fixRandomSeed     随机种子（估计用处不大）
  --record                将游戏交互数据保存
  --replay=REPLAY         将保存的游戏交互数据回放
  -x NUMTRAINING, --numTraining=NUMTRAINING 训练-自行弄清楚
  -c, --catchExceptions exception捕获
```

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python3为标准，函数对外接口为 `algo/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，**4.2.2**子小节将详细介绍该内容；

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，该部分实现较为简单且代码量较多故不详细介绍。

4.2.2 Agent编写相关参考文件介绍

- 基类 `Agent` -- 位于 `game.py` 文件约32行

```
class Agent:
    """
    An agent must define a getAction method, but may also define the
    following methods which will be called if they exist:

    def registerInitialState(self, state): # inspects the starting state
    """
    def __init__(self, index=0):
        self.index = index

    def getAction(self, state):
        """
        The Agent will receive a GameState (from either {pacman, capture, sonar}.py) and
        must return an action from Directions.{North, South, East, West, Stop}
        """
        raiseNotDefined()
```

- 基类告诉我们，一个Agent应该有哪些内容，你的代码可以不继承它，但必须含有其包含的内容
- `self.index`：指示当前Agent的编号 (1, 2, 3, 4)
- `getAction`：调用该函数以获取当前Agent的动作，即Agent根据当前 `state` 做出相对应决策，返回 `action`

- 派生类 `RandomAgent` 与派生类 `TimeoutAgent` -- 位于 `captureAgents.py` 文件约36行与约293行

```
class RandomAgent( Agent ):
    """
    A random agent that abides by the rules.
    """
    def getAction( self, state ):
        return random.choice( state.getLegalActions( self.index ) )

class TimeoutAgent( Agent ):
    """
    A random agent that takes too much time. Taking
    too much time results in penalties and random moves.
    """
    def getAction( self, state ):
        import random, time
        time.sleep(2.0)
        return random.choice( state.getLegalActions( self.index ) )
```

- 派生类 `RandomAgent` 的 `random.choice( state.getLegalActions( self.index ) )` 表示根据当前 `state` 任意挑选一个合法 `action`
- 派生类 `TimeoutAgent` 用于演示超时惩罚
- 若想使用 `RandomAgent` 或 `TimeoutAgent` 进行功能演示/测试/对战，请加上 `import random, time`，所给予的代码中没有import这些个模块

- **算法原型类** `CaptureAgent` -- 位于 `captureAgents.py` 文件约46行，下面将粗略介绍改类下面的函数

```
def registerInitialState(self, gameState):
    # 仅在Agent/游戏初始化时调用一次，初始化地图信息、初始化distanceCalculator.Distancer最短距离测算子，以及计算Agent与地图相关元素的初始距离
    pass
```

```
4
5  def getAction(self, gameState):
6      # 最关键的实现，action返回的接口，下面的chooseAction函数告诉我们，我们可以通过控制条件与多个函数/策略选择性调用获取action
7      pass
8
9  def getFood(self, gameState):
10     # 返回Agent可/要吃的食物（红色方吃蓝色区域的食物，反之亦然）。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。
11     pass
12
13 def getFoodYouAreDefending(self, gameState):
14     # 返还Agent要保护的食物（红色方保护红色区域的实物，反之亦然）。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。
15     pass
16
17 def getOpponents(self, gameState):
18     # 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]）。
19     pass
20
21 def getTeam(self, gameState):
22     # 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]）。
23     pass
24
25 def getScore(self, gameState):
26     # 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)
27     pass
28
29 def getMazeDistance(self, pos1, pos2):
30     # 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。
31     pass
32
33 def getPreviousObservation(self):
34     # 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）。
35     pass
36
37 def getCurrentObservation(self):
38     # 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）。
39     pass
```

- 请仔细阅读该文件提供的函数，即便不继承该文件也可以从中获取自身算法设计的灵感，降低编写代码难度
- 请注意 `getScore` 函数中对显示分数与Agent获取的分数的区别
- 建议编写算法时继承 `CaptureAgent` 类，省掉自身造轮子的时间

- 算法编写可参考的 `baselineTeam.py` 中的 `ReflexCaptureAgent` 类与 `ReflexCaptureAgent` 派生的攻击型Agent `OffensiveReflexAgent` 与 `DefensiveReflexAgent` 。为更好地介绍，本文以 `baselineTeam.py` 中出现的函数与类做详细讲解

```
##################
# Team creation #
##################

def createTeam(firstIndex, secondIndex, isRed,
               first = 'OffensiveReflexAgent', second =
'DefensiveReflexAgent'):
    """
    This function should return a list of two agents that will form the
    team, initialized using firstIndex and secondIndex as their agent
    index numbers.  isRed is True if the red team is being created, and
    will be False if the blue team is being created.

    As a potentially helpful development aid, this function can take
    additional string-valued keyword arguments ("first" and "second" are
    such arguments in the case of this function), which will come from
    the --redOpts and --blueOpts command-line arguments to capture.py.
    For the nightly contest, however, your team will be created without
    any extra arguments, so you should make sure that the default
    behavior is what you want for the nightly contest.
    """
    return [eval(first)(firstIndex), eval(second)(secondIndex)]
```

- 以上函数为队伍初始化接口
  - firstIndex/secondIndex：队伍中第一/二个Agent在游戏中的序号
  - isRed: 红方蓝方标识符，你可以使用这个flag针对红蓝方地图差异（镜像+翻转）做处理，可选
  - first: 指示第一个Agent需要调用哪个类进行初始化，这里第一个Agent为OffensiveReflexAgent
  - second: 指示第二个Agent需要调用哪个类进行初始化，这里第二个Agent为DefensiveReflexAgent

- 介绍 `eval(first)(firstIndex)` 函数，动态实例化，以上面的默认参数为例，`eval(first)(firstIndex)` 等价于 `OffensiveReflexAgent(firstIndex)`，实例化一个Agent，不建议初学者在代码的其他地方出现这种写法

```
##########
# Agents #
##########

class ReflexCaptureAgent(CaptureAgent):
    """
    这是一个OffensiveReflexAgent类与DefensiveReflexAgent类的"基类"，用于定义两者通用的函数
    """
    def registerInitialState(self, gameState):
        # 初始化，同上
        pass

    def chooseAction(self, gameState):
        # 选择动作的流程一致，准则不同
        pass

    def getSuccessor(self, gameState, action):
        # 查找下一个后续目标，即网格位置（位置元组）。具体作用与后续实现类有关，自行探索
        pass

    def evaluate(self, gameState, action):
        # 根据特征与权重评估行为价值

    def getFeatures(self, gameState, action):
        # 返回状态的特征计数器，与策略相关，自行探索

    def getWeights(self, gameState, action):
        # 不同行为的分数权重，指引行为倾向

class OffensiveReflexAgent(ReflexCaptureAgent):
    """
    这是一个OffensiveReflexAgent类，用于吃豆
    """
    def getFeatures(self, gameState, action):
        # 针对OffensiveReflexAgent攻击性特征计算
        pass
    def getWeights(self, gameState, action):
        # OffensiveReflexAgent行为权重
        return {'successorScore': 100, 'distanceToFood': -1}

class DefensiveReflexAgent(ReflexCaptureAgent):
    """
```

```
    这是一个DefensiveReflexAgent类，用于防守和吃吃豆人
    """

    def getFeatures(self, gameState, action):
        # 针对DefensiveReflexAgent防守性特征计算
        pass

    def getWeights(self, gameState, action):
        # DefensiveReflexAgent
        return {'numInvaders': -1000, 'onDefense': 100, 'invaderDistance': -10, 'stop': -100, 'reverse': -2}

```

- 请借鉴 `algo/myTeam.py` 或 `baselineTeam.py` 实现你的算法，**并将其命名为** `组号.py` **或** `组号+组名.py` **放置在** `algo` **文件夹下**
- 不需要两个Agent有明确分工，你可以选择两个进攻性Agent
- 上述解释不一定完全准确，请参考但不要盲从

4.3 一些问题的提前解答

- 怎么验证红蓝两方都能运行 -- 算法可以左右互搏，如

```
python capture.py -r algo/myTeam -b algo/myTeam
```

- 组名太长在图中无法显示 -- 修改 `captureGraphicsDisplay.py` 中 `InfoPane` 类的两个函数即可

```
class InfoPane:
    def _redScoreString(self):
        return f"RED: {self.redTeam}"

    def _blueScoreString(self):
        return f"BLUE: {self.blueTeam}"
```

- 不想看到游戏中算法的 `algo/` 字样，修改 `captureGraphicsDisplay.py` 的 `InfoPane` 类的初始化变量即可

```python
import os
class InfoPane:
    def __init__(self, layout, gridSize, redTeam, blueTeam):
      ...
      self.redTeam = os.path.split(redTeam)[-1]
      self.blueTeam = os.path.split(blueTeam)[-1]
```

- 修改为 `组名.py` 后无法运行成功
  - 原因1：文件名中包含以下元素
    - 非UTF8编码字符
    - & " ? < > # { } % ~ / \ : 与转义字符 (全角符号一般是可以的)
  - 原因2：相关 `import` 没修改

- 为什么我的代码在助教/评测中无法运行
  - Python版本问题：如在 `Python3.10` 中使用 `match` 关键词，而评测机器 `python` 版本不支持 → 使用conda/virtualenv创建对应虚拟环境，创建 `Python3.8` 或 `Python3.9` 环境（虚拟环境创建步骤可参考 `tensorflow` 教程**使用 pip 安装 TensorFlow**）
  - 设备问题：使用了 `cuda` 训练模型，但在保持模型参数时候没转为 `cpu` 类型（参照文档选择默认在cpu运行**Saving and loading models across devices in PyTorch — PyTorch Tutorials 2.3.0+cu121 documentation**）
  - 库版本问题： `pytorch` 请使用1.8.2 LTS版本， `numpy` 请使用最新版本， `tensorflow` 请使用最新版本

4.4 组队

- 人数限制：1~3人，需要确定一名队长（独立组队则只有队长），提交等操作皆由队长负责
- 组队信息填写： **【企微文档】2024春季学期大作业分组**（需要登录后才能填写）
  - 请根据第一行的格式填写，组号以1开始
  - 终止填写时间: **2024春季学期 -- 2024年05月17号星期五21:56:59**。终止时间到后将会锁定文档中组队成员相关列数据，改为仅可观看。与此同时，会将表格信息导出并发至群里。如有异议，请在**2024年05月19号星期日 21:56:59前**联系助教修改，过时不候。最终版分组名单也会在修改时间过后发至群里。**后续将不允许修改组队信息（没组队的可能会随机分配）。**

a. 代码提交

- 在最终比赛评估前提交，具体时间**待定**，通过收集链接收集，需要提交算法可运行的最小组件，以确保算法能正常运行
- 将下列文件压缩为 code.zip
  - `./algo` 下的 `.py` 文件，请注意命名这些文件的时候请根据组号或组号+组名命名，如 `0+test_name.py` 或 `0.py` ， `.pth` 、 `h5py` 等文件请放置在 `algo/0/` 文件夹下
  - 删除 `__pycache__` 之类的缓存文件、测试无关的文件

b. 报告与相关文件提交

- 需要提交纸质报告（归档需要）与电子版报告，纸质报告提交时间**待定**（后续通知）。电子报告可在纸质报告提交后，上传至收集链接，提交时间**待定**（后续通知）
    - 以组为单位提交，报告每组写一份。其中算法设计部分合写一份，感想部分每人一份
    - 以 算法设计-感想 为顺序合并到一个文件，并导出为**pdf**文件，以防遗漏和排版错乱，并将其命名为 组号+组名+报告.pdf ， 如00+test_team.pdf
    - 算法设计部分需详细介绍算法设计流程，并解释所编写的核心代码模块的作用/逻辑。需在该文档最前面注明以下必要信息，尽量使用表格形式填写下述内容 小组序号+小组名字 与 各成员姓名+学号 以及 分工合作情况 ，下面提供一个例子，可以借鉴该例子在报告文档中赋予

组号+组名	0+test_name
各成员姓名+学号	王五+22222222 张三+22222220 李四+22222221
分工合作情况	王五：算法总体设计 张三：进攻智能体设计调优 李四：防守智能体设计调优

    - 电子版报告提交时需同时附上最终版代码，以组为单位提交，将这些文件压缩到一个压缩包（zip格式），并根据命名规则将压缩包命名 ，命名规则为： 组号+组名.zip
        - 代码所需的提交内容与要求，以下为一些提交要求
            - 代码需要必要的注释，需要交代每个函数的作用，以及介绍输入输出的含义，引导阅读者。对于一些magic number、复杂写法与不常见语法糖需要提供容易理解的解释。
            - 将下列文件压缩为 code.zip
                - ./algo 下的 .py 、 .pth 、 h5py 等文件
                - 删除 __pycache__ 之类的缓存文件、测试无关的文件
            - 对于学习类算法 -- 还需下列文件压缩
                - 需要提供整个项目，并需要写一个README文档，介绍如何使用你的算法进行训练与测试（运行指令、测试指令）、状态输入/奖励内容与修改原因
                - 删除不使用的模型文件与训练测试log以及 __pycache__ 之类的缓存文件
- 评分会考虑以下因素，具体包含哪些因素以及组成成分请根据老师后面制定的规则为主：
    - 比赛成绩（客观）
    - 报告质量（主观）：代码可读性、工作量以及idea等