

2018 春季地空数算期末作业

—— paper.io.sessedata（纸带圈地）说明文档

陈斌、陈天翔、张赖和

目录

1	期末作业简要介绍	3
2	本游戏介绍	4
2.1	战斗	4
2.2	棋盘	4
2.3	回合	4
2.4	判定：战斗的胜负	4
2.5	判定：标记和圈地	5
3	战斗的运行	6
3.1	系统设施	6
3.2	参战方	6
3.3	对战平台运行方式	6
4	各小组工作	7
4.1	函数的编写	7
4.2	play 函数	7
4.3	load 函数（可选）	8
4.4	summary 函数（可选）	8
4.5	stat	8
4.6	游戏数据（玩家信息部分）	9
4.7	storage	9
4.8	坐标系统	9

目录	2
4.9 小组算法开发指南	10
5 期末作业安排	11
5.1 算法竞赛规则	11
5.2 组队安排	11
5.3 作业进度安排	11
5.4 评分标准	11
5.5 竞赛分组	12
5.6 赛制安排	12
6 类 player 的功能介绍	13
7 结语和附注	15
7.1 附注	15

1 期末作业简要介绍

本部分将简要介绍本学期（2018 春季学期）本课程（北京大学地球与空间科学学院数据结构与算法课程）的期末作业（以下称本作业）。

本作业要求以小组为单位，综合数据结构与算法课程内容，采用 Python 语言编写算法，使得提交的算法能进行 `paper.io.sessed` 游戏（以下称本游戏），并能在本游戏中自主行动并控制纸卷。

本作业将提供对战平台进行对抗性竞赛，算法应能顺利游玩游戏，并根据与其他算法的对局结果取得竞赛排名。

本游戏简要说明

本游戏的原型为 Xonic PC Game 1984；与其他版本可能稍有不同，本游戏没有视野限制，每个玩家可以在自己当前的回合获取盘面上的所有信息，为完全信息决策游戏。

本游戏为一个回合制游戏，游戏给定回合数和棋盘大小限制。每局游戏由两个玩家或 AI 参与。两个玩家分别控制自己的纸卷。纸卷一直在移动并抽出纸带。

玩家可以透过控制纸卷向左转、向右转或直行等三个方向，让纸带与自己的已有区域形成新的闭合区域并填充，不断扩大领地。游戏过程中，如果玩家的纸卷碰到边界，该玩家的纸带会断裂；如果玩家的纸带被任何纸卷（包含自己或别人的）碰到，该玩家的纸带也会断裂。

游戏目的乃透过合法操作扩大自己的领地或让对方的纸带断裂。

以下进行游戏术语和规则的详细介绍。

2 本游戏介绍

2.1 战斗

每场战斗有两方参与，代号分别为 1 和 2。战斗初始阶段，系统会给定游戏总回合数、双方允许的总耗时以及以下初始属性：

1. 在地图西半部的正中间附近生成 1 方纸卷，给定 1 方纸卷的初始移动方向（东、南、西、北之一），并以 1 方纸卷为中心生成一个 3×3 的 1 方初始已有区域。
2. 在地图东半部的正中间附近生成 2 方纸卷，给定 2 方纸卷的初始移动方向（东、南、西、北之一），并以 2 方纸卷为中心生成一个 3×3 的 2 方初始已有区域。

具体数值将在之后说明；初始阶段结束后，战斗正式开始。

2.2 棋盘

1. 棋盘：本游戏在具有边界的离散格点网上进行，其中东西向格点数为 $2k$ ，南北向格点数为 h ，且 k 和 h 均为奇数。此两个值由系统给定。
2. 格点属性：每个格点有两个属性：
 - 领地属性：1 方领地，2 方领地，无领地属性。
 - 纸带属性：1 方纸带，2 方纸带，无纸带属性。

2.3 回合

1. 回合总数：游戏的回合总数由系统给定。
2. 单个回合：单个回合内，按顺序进行以下操作：
 - (a) 1 方读入当前盘面和相关数据，返回左转、右转或直行的其中一个。
 - (b) 系统按 1 方的要求移动纸卷，并进行相关判定。
 - (c) 2 方读入当前盘面和相关数据，返回左转、右转或直行的其中一个。
 - (d) 系统按 2 方的要求移动纸卷，并进行相关判定。

以上步骤进行完毕后，一回合结束。注意到每回合由两个盘面更改的要求，因此每回合将有两个更新数据；但每回合之中，每个玩家仅能移动一次。

2.4 判定：战斗的胜负

- 下列情况发生时，游戏立即结束，并对操作方判负，另一方判胜：
 - 己方纸卷超出棋盘

- 己方纸卷碰到己方纸带
- 己方纸卷在对方领地碰到对方纸卷
- 己方算法给出不合法操作（包含超时未给出操作）
- 下列情况发生时，游戏立即结束，并对操作方判胜，另一方判负：
 - 侧碰：己方纸卷在无领地属性格点碰触对方纸卷，且双方纸卷运动方向垂直
 - 己方纸卷碰到对方纸带
 - 己方纸卷在己方领地碰到对方纸卷
- 下列情况发生时，游戏立即结束，并统计双方的领地（纸卷或纸带不计）大小。若双方领地大小不同，判领地大者为胜方；若双方领地大小相同，判双方平手：
 - 对碰：己方纸卷在无领地属性格点碰触对方纸卷，且双方纸卷运动方向相反
 - 回合数耗尽

2.5 判定：标记和圈地

纸卷运动到每个格点时，若没有达成战斗胜负的条件，则进行以下判定：

1. 如果该格点的属性不是己方领地，则对该格点附加标记为己方纸带。
2. 如果该格点的属性是己方领地，则按下列顺序进行圈地操作：
 - (a) 首先将标记为己方纸带的格点的领地属性更改为自己的领地属性。
 - (b) 考察己方纸带与己方领地外缘所围成的区域（以闭包形式记录），将这些区域的所有格点的领地属性更改为自己的领地属性（但纸带属性暂时不动）。
 - (c) 将标记为己方纸带的格点的纸带属性移除，完成圈地操作。

3 战斗的运行

3.1 系统设施

系统设施有以下三个部分构成：

- 'match_core.py': 游戏执行逻辑；有 md 文件进行配套说明；战斗运行时调用的档案；主管战斗的进行和判定规则。
- 'roundRobin.py': 进行循环赛时使用的档案；引用'match_core.py' 和各玩家的 play 函数使之进行游戏，并输出对局过程和结果。
- 'visualize_console.py': 控制台复盘代码；有 md 文件进行配套说明；将对局过程和结果进行可视化。

3.2 参战方

各参战方提供一个参战函数。该函数应有如下格式，具体要求见【各小组工作】：

```
def play(stat , storage):  
    # 编写你的函数  
    pass  
  
def load(storage):  
    # 编写你的函数  
    pass
```

3.3 对战平台运行方式

1. 确认'match_core.py', 'roundRobin.py', 'visualize_console.py' 等三个文件与文件夹'AI' 在同一目录且同级。
2. 将要参战的函数按要求存成.py 文件，并这些文件放入文件夹'AI' 。
3. 按顺序运行'roundRobin.py' 和'visualize_console.py'。
4. 根目录 > log > 'A-VS-B.txt' 即为 A 和 B 分执 1 方和 2 方的战斗过程。

4 各小组工作

4.1 函数的编写

- 编写的代码档案应以 python 3.5 或更高版本形式储存为.py 文件；函数对外的接口应命名为 play；代码文件中应只包含 play 函数、load 函数与 summary 函数，其中 load 与 summary 函数非必需；其余一切代码（不含注释部分）均应封装在这些函数内，包含导入库的语句。比赛程序将按语句 `action = play(stat, storage)` 调用 play 函数，其中 action 为战斗中系统识别的方向变更要求，应等于算法的给出的返回值；具体形态将于本章其它部分说明。
- 为维护赛场整洁，禁止使用全局变量语句与 os, sys, threading, multiprocessing 等标准库。
- 不建议使用第三方库编写算法代码；此举可能被比赛环境视为 ImportError 而判负；传递的参数均为标准 python 对象，算法代码无需 `import match_core` 即可正常工作。
- 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 play 函数

编写的 AI 函数对外的接口应命名为 play，函数的接收参数包含游戏信息 stat 与存储对象 storage，比赛系统将按 `action = play(stat, storage)` 形式调用，在每一回合开始时调用执行 play 函数，决定函数控制的玩家方向的改变（左转、右转、不改变方向），随后朝改变后的方向前进一单位的距离。

比赛核心逻辑中使用以下代码执行函数返回值操作：

```

action = play(stat, storage)
...
if isinstance(action, str) and len(action) > 0:
    op = action[0].upper()
    if op == 'L':
        plr.turn_left()
    elif op == 'R':
        plr.turn_right()

```

若函数返回值为非空字符串且首字母在大写操作后为 L 或 R，则分别视为左转或右转操作；否则视为直行。

- 识别为“左转”：'l', 'L', 'Left', 'LEFT', 'Legendary' ...
- 识别为“右转”：'r', 'R', 'right', 'RIGHT', 'Robust' ...
- 识别为“前进”：None, 'iwannawin', [1, 2, 3], '0', 0 ...

4.3 load 函数（可选）

可选的 load 函数在比赛开始前接收游戏信息 stat 与存储对象 storage，可在此进行 AI 函数所需变量的声明与函数的引用等；若该函数未声明将使用 lambda storage:None 替代。

本函数的加载数据用时将计入玩家总时间。

4.4 summary 函数（可选）

对局总结函数，每场比赛结束后运行一次，可将总结内容记录于 storage['memory'] 关键字的字典中，内容将会在不同比赛间保留，比赛运行方保证对双方 storage 字典合乎逻辑的维护；

接收参数包含三部分：对局结果 match_result，游戏数据 stat 与函数存储 storage。

4.5 stat

游戏数据 stat 呈现为字典形式，包含当前游戏状态信息关键字内容；以下所提及的任何关键字 kw，对应的数据访问方式为 stat['kw']，部分内容详见【各小组工作 > 游戏数据（玩家信息部分）】：

- turnleft: 剩余回合数，按先后手排序。
类型: list[int]
内容: [先手玩家剩余回合数, 后手玩家剩余回合数]
- timeleft: 双方剩余总思考时间（秒），按先后手排序
类型: list[float]
内容: [先手玩家剩余时间, 后手玩家剩余时间]
- fields: 纸片场地二维列表
类型: list[list[int]]
内容: fields[x][y] 返回坐标 (x, y) 点纸片领地归属，1 代表先手玩家，2 代表后手玩家，None 代表无纸片覆盖。坐标含义详见【各小组工作 > 坐标系统】部分
- bands: 纸带场地二维列表
类型: list[list[int]]
内容: bands[x][y] 返回坐标 (x, y) 点纸带领地归属，1 代表先手玩家，2 代表后手玩家，None 代表无纸带覆盖。坐标含义详见“附：坐标系统”部分
- me: 自己控制的玩家信息
- enemy: 对手玩家信息
- players: 玩家信息列表，包含双方玩家信息，按先后手排序
类型: list[dict]
内容: [me, enemy]

4.6 游戏数据（玩家信息部分）

play 函数的游戏数据（玩家信息部分）包含当前游戏状态中一个玩家的状态，其中有：

- id: 玩家标记
类型: int
内容: 1: 先手玩家; 2: 后手玩家
- x, y: 横、纵坐标; 详见【各小组工作 > 坐标系统】部分
- direction: 数字标记的当前方向
类型: int
内容: 0: 向东; 1: 向南; 2: 向西; 3: 向北; 详见“附: 坐标系统”部分

4.7 storage

存储对象 storage 呈现为字典形式, 可供各函数自由使用与存储数据; 同时记录以下默认数据:

- size: 列表, 包含游戏场景宽高
类型: list[int]
内容: [场地宽, 场地高]
- log: 运行记录, 包含对局开始以来每一回合的游戏数据
类型: list[dict]
内容: 按操作顺序, 记录初始状态, 以及每次游戏状态变动后的数据

写入 storage 字典中的内容会一直保存至比赛结束, 包括默认数据内容与自定义内容。

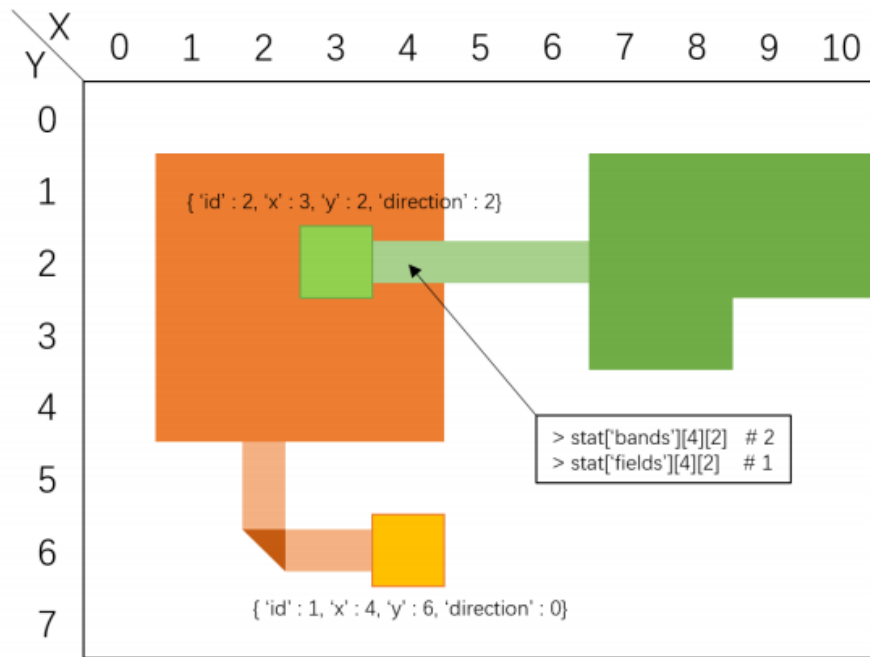
不建议占用 size 与 log 关键字, 可能造成数据丢失。

4.8 坐标系统

游戏数据中返回的横纵坐标均为整数, 表示游戏地图的网格位置: 其中 x 坐标范围为 [0, 场地宽 -1], y 坐标范围为 [0, 场地高 -1]。

fields 与 bands 二维列表中内容按列存储; 调用时第一个下标为横坐标, 第二个下标为纵坐标。

本文档中以东南西北作为绝对方向的指代, 其中 x 坐标增加的方向为东, y 坐标增加的方向为南。



4.9 小组算法开发指南

- 具体范例格式与要求也可参考档案'AI_Template.py'。
- 系统将提供数个（并不聪明的）AI，可供参考和测试。

5 期末作业安排

5.1 算法竞赛规则

1. 竞赛目标：采用算法指挥己方纸卷，使算法能自主改变纸卷在棋盘上的运动方向，利用棋盘态势信息计算纸卷走向，使其能透过圈到更多领地、碰断对方纸带等方式以取胜。
2. 战斗开始时，根据【本游戏介绍 > 战斗】所述赋予双方各自的初始状态。
3. 随着回合的更迭，双方按次序轮流改变运动方向，以每回合 1 格的速度移动纸卷。
4. 系统的回合数和棋盘大小给定。
5. 根据本文所述规则决定胜负。

5.2 组队安排

本作业的进行以组为单位，原则上每组 1 名组长，总人数 4 或 5 人（含组长），组队过程如下：

1. 首先确定组长名单。所有人能自愿报名入选组长名单，报名表稍后公布于课程网站。历次作业优秀数量较多的同学能优先入选组长名单。
2. 组长名单确定后，各组长可开始招募组员，是为组队过程。组队过程中遵循自愿原则，提倡均衡原则。
3. 组长负责召集本作业的过程讨论、代码汇总、报告等内容，并代表小组参加竞赛。

5.3 作业进度安排

本作业内容包含（但不仅限于）开发算法、变成测试、热身挑战、报告撰写等项目。

1. 即日起开始作业的进行，注意组员分工明确，协同合作。
2. 6 月 12 日周二课上进行算法竞赛。
3. 6 月 19 日周二前提交完整作业，包含代码和实验报告。

5.4 评分标准

- 本作业占总评的 25 分，其中算法编程占 9 分，报告占 8 分，竞赛排名占 8 分；此评分以组为单位。
- 竞赛排名分：参赛无异常 3 分；第 1 轮出线 5 分；季军 6 分；亚军 7 分；冠军 8 分。
- 每组共有至多额外 3 分加分，可由组长组织本组民主评议，奖励 1 至 2 名表现突出的组员（含组长）。组长另有权对实习过程中表现差的同学提出批评及降分建议。

5.5 竞赛分组

- 本学期选课人数共 249 人，其中 17 级 127 人，非 17 级 122 人。
- 为了均衡实力，缩小小组规模和数量，将全体同学根据是否为 17 级分为两个联盟。
- 联盟内部按照世界杯赛制进行四轮比赛，决定联盟内名次，结束联盟内正式比赛。
- 联盟内部的正式比赛完毕后，可进行联盟之间的各类友谊赛、挑战赛、乃至人机对战等非正式赛项目。

5.6 赛制安排

- 赛前进行热身挑战赛。愿意参加热身赛的小组，可将代码发给老师，并可获得与其他小组的对战结果和复盘数据（但不可透过热身赛获得其他小组的代码）。
- 正式赛开始时，先将同联盟内的各个小组随机分为东 (E)、南 (S)、西 (W)、北 (N) 四个分区。
- 第一轮为区内赛，采用循环赛制，每区取区内积分前两名出线，是为八强；第一轮每场按对战结果给区内竞赛积分，其中胜者 3 分，负者 0 分，平局各 1 分。若遇积分相同情况，则根据以下规则顺序决定名次先后：
 1. 净胜局多者位列高名次。
 2. 净胜局相同者，比较彼此之间的胜负关系并单序排列，决定名次。
 3. 净胜局相同且胜负关系无法单序排列者，则抽签决定名次。
- 第二轮为四分之一决赛，根据以下赛制安排和比赛结果决定四强，其中不接受和局：
 - 赛局 A：东区第 1 名 vs 西区第 2 名
 - 赛局 B：东区第 2 名 vs 西区第 1 名
 - 赛局 C：南区第 1 名 vs 北区第 2 名
 - 赛局 D：南区第 2 名 vs 北区第 1 名
- 第三轮为半决赛，根据以下赛制安排和比赛结果决定冠军赛和季军赛名单，其中不接受和局：
 - 赛局 E：赛局 A 胜者 vs 赛局 B 胜者
 - 赛局 F：赛局 C 胜者 vs 赛局 D 胜者
- 第四轮为决赛和季军赛，根据以下赛制安排和比赛结果决定名次，其中不接受和局：
 - 冠军赛：赛局 E 胜者 vs 赛局 F 胜者
 - 季军赛：赛局 E 负者 vs 赛局 F 负者

6 类 *player* 的功能介绍

以下为类 *player* 内各项功能的简要介绍；详细内容和实现代码请看 *match_core.py*。

```
class player:
    '''玩家逻辑

    params:
        id - 向数组中添加位置 players
        x, y - 初始坐标
        expand - 初始纸片边长为  $(2 * expand + 1)$ 
        init_direction - 初始方向，默认为随机方向
    '''

    directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]
    # 方向数字对应前进方向

    def __init__(self, id, x, y, expand=1, init_direction=None):
        self.id, self.x, self.y = id, x, y
        # 记录纸带轨迹，便于状态转换时回溯

        # 设置初始方向

        # 生成初始占有区域 (l, r, u, d)

    def turn_left(self): # 左转

    def turn_right(self): # 右转

    def forward(self):
        '''向前移动一步，并相应更新场地
        returns: 若未到达终盘条件
            - None 若到达终盘
            - 胜利玩家（或代表对应玩家直接胜利；
              ID12代表需要进行结算的战斗终止情况）
              None 终局原因编号（详见函数注释）
              parse_match 额外内容
        '''
        # 移动
```

```
# 边缘检测

# 更新占有区域边界

# 更新玩家碰撞

# 更新纸带碰撞

# 更新场地

def update_field(self):
    '''更新玩家在场地引起的效应'''
    # 场地内检查填充情况

    # 场地外延伸纸带

def extend_band(self):
    '''（在自己领地外）延伸纸带'''

    # 纸带转换为领地

    # floodfill 填充空腔

def get_info(self):
    '''提取玩家信息，用于传递给函数AI
    returns:字典，包含，位置，方向
            id'''

    return {
        'id': self.id,
        'x': self.x,
        'y': self.y,
        'direction': self.direction
    }
```

7 结语和附注

本作业提出时间较短，可能存在不足之处，敬请各位见谅，并欢迎各位协助修改或提出指正。

7.1 附注

本文档更新于 2018/05/22 15:03.