

SESSDSA.2048 开发者文档

1. 引言

本文档面向 sessdsa.2048 开发者。

2. 对战平台

对战平台的接口并不直接对外暴露。如果您是对抗算法开发者，可以略过这个部分的说明；如果您是前端开发者或赛制工具开发者，可能需要了解对战平台的接口和数据协议以利于您的开发。

A. 说明对象

[plat.py](#)的Platform类。

B. 类变量

- i. self.states：参赛 ai 的运行状态，为 dict 类型，遵循states的数据协议。
- ii. self.match 比赛名称（保存路径），为有效的str格式路径。
- iii. self.livequeue 用于直播线程通信的队列，为queue.Queue类型。
- iv. self.toSave 是否保存记录文件，为bool类型。
- v. self.maxtime 最大时间限制，为int类型。
- vi. self.rounds 总回合数，为int类型。
- vii. self.winner 胜利者，为bool类型或者None；为True代表先手，为False代表后手，为None代表无。
- viii. self.violator 违规者，为bool类型或者None；为True代表先手，为False代表后手，为None代表无。
- ix. self.timeout 超时者，为bool类型或者None或者字符串'both'；为True代表先手，为False代表后手，为None代表无，为'both'代表先后手。
- x. self.error 报错者，为bool类型或者None或者字符串'both'；为True代表先手，为False代表后手，为None代表无，为'both'代表先后手。

- xi. self.currentRound 当前轮数，为int类型，值为 0 代表第一轮。
- xii. self.change 某次操作中棋盘是否发生改变，为bool类型。
- xiii. self.next 根据随机序列得到的下一个位置，为tuple类型，遵循position的数据协议。在极特殊的情况下，self.next = (None, None)，代表在先后手方初始化的过程中抛出了异常。
- xiv. self.board 棋盘，为Chessboard类的实例。
- xv. self.log 日志，为平台内定义的Log类的实例。Log类继承list类，可以依据list类的接口操作逐条存储的日志记录。单条日志记录遵循log的数据协议。
- xvi. self.name 本局比赛的名称，为repr格式的哈希字符串。

C. 类方法

- i. __init__(self, states, match, livequeue, toSave, MAXTIME, ROUNDS) 初始化平台对象，为参赛 ai 重载方法使其具有计时与捕获异常的功能。

参数表如下

- 1. states 与self.states遵循相同约定，但是参赛ai对象应当只实例化而未经初始化。（即使用object.__new__(cls)实例化对象）
- 2. match 与self.match遵循相同约定。
- 3. livequeue 与self.livequeue遵循相同约定。
- 4. toSave 与self.toSave遵循相同约定。
- 5. MAXTIME 与self.maxtime遵循相同约定。
- 6. ROUNDS 与self.rounds遵循相同约定。
- ii. play(self) 初始化参赛 ai 并进行单局比赛，返回比赛报告。
返回值遵循result的数据协议。
- iii. start(self) 进行比赛，**不支持外部调用**。
- iv. checkState(self, isFirst) 检查超时和报错，**不支持外部调用**。
- v. checkViolate(self, isFirst, mode, value) 检查非法输出，**不支持外部调用**。
- vi. save(self) 保存比赛记录文件，遵循记录保存数据协议，**不支持外部调用**。

D. 数据协议

- i. states
states = {True: {'player': player,
 'path': path,

```

        'time': time,
        'time0': time0,
        'error': error,
        'exception': exception,
        'index': index},
False: {'player': player,
        'path': path,
        'time': time,
        'time0': time0,
        'error': error,
        'exception': exception,
        'index': index}}

```

键True与False分别代表先后手，其对应的值是其运行状态，说明如下

1. player 参赛ai对象，为参赛队伍提供的类的实例。
2. path 参赛队伍提供的类的路径或者类本身（视读取的时候是按路径读取还是按类读取），为有效的str格式路径或类。
3. time 参赛ai的当前时间，为float类型。
4. time0 参赛ai的时间零点，为int或float类型，**通过调节此变量可以实现对特定参赛ai的总时间限制进行修改。**
5. error 参赛ai是否抛出异常，为bool类型。
6. exception 参赛ai运行时抛出的异常信息，为标准traceback字符串或者（当未抛出异常时）None。
7. index 参赛ai的编号，为tuple类型，遵循index的数据协议。

ii. log

单条log记录由&起始。分为三类

1. &d?: ... 某方作出决策

这里的?与...分别为当前轮数与决策说明。

作为例子，在第一轮先手方选择向右合并，将会生成记录

&d0: player 0 set direction right

而在第五轮后手方选择在(1, 4)下棋，将会生成记录

&d4: player 1 set position (1, 4)

2. &p?: ... 当前棋盘

这里的?与...分别为当前轮数与棋盘的repr格式。

作为例子，第 43 轮的某个棋盘记录为

&p42:

```
+00 +00 +00 +03 +02 -00 -00 -00
+00 +00 +01 +02 -03 -01 -00 -00
+00 +05 +03 +01 -05 -03 -01 -00
+00 +00 +01 +05 -02 -02 -04 -01
```

3. &e: ... 游戏事件触发

这里的...为事件说明。游戏事件有：违规、超时、报错、胜利、计分、平局。

作为例子，列举一些记录如下

&e: player 1 violate by illegal output of direction

&e: player 0 win

iii. result

```
result = {True: {'index': index,
                 'win': win,
                 'lose': lose,
                 'violate': violate,
                 'timeout': timeout,
                 'error': error,
                 'time': time,
                 'exception': exception},
          False: {'index': index,
                  'win': win,
                  'lose': lose,
                  'violate': violate,
                  'timeout': timeout,
                  'error': error,
                  'time': time,
                  'exception': exception},
          name: name,
          rounds: rounds}
```

键name与rounds对应的值分别为本局比赛名称与比赛实际进行总轮数，分别遵循self.name和self.rounds的约定。键True与False分别代表先后手，其对应的值是其表现报告，说明如下

1. index 参赛ai的编号，为tuple类型，遵循index的数据协议。
2. win 是否胜利，为bool类型。
3. lose 是否失败，为bool类型。
4. violate 是否违规，为bool类型。
5. timeout 是否超时，为bool类型。
6. error 是否报错，为bool类型。
7. time 参赛ai总运行时间，为int或float类型。
8. exception 参赛ai运行时抛出的异常信息，为标准traceback字符串或者（当未抛出异常时）None。

iv. position

position = (row, column)

row, column为int类型，代表位置的行、列坐标

v. index

index = (count, isFirst)

其中count为int类型，是按顺序编定的参赛模块编号；isFirst为bool类型，代表是否先手。

vi. 比赛记录保存

比赛记录将保存为标准txt文件，包含该局比赛的元信息和全部日志记录。

某次比赛记录如右图所示，从上而下分别记录了模块的路径（或模块本身）、比赛时间、比赛结果和全部比赛日志。

此比赛记录可以被复盘工具解析并实现可视化，利于开发者进行调试。

```
player0: 0 from path player.py
player1: 1 from path player.py
time: 2020-05-05 14:42:10
*****basic record*****
| timeout | violator | error | winner |
| None | player 1 | None | player 0 |
*****
|player| 1| 2| 3| 4| 5| 6| 7| 8| 9| 10| 11| 12| 13|
| 0| 0| 1| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0|
| 1| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0|
*****complete record*****
&d0:player 0 set position (0, 0)
&p0:
+01 +00 +00 +00 -00 -00 -00 -00
+00 +00 +00 +00 -00 -00 -00 -00
+00 +00 +00 +00 -00 -00 -00 -00
+00 +00 +00 +00 -00 -00 -00 -00
&d0:player 1 set position (0, 4)
&p0:
+01 +00 +00 +00 -01 -00 -00 -00
+00 +00 +00 +00 -00 -00 -00 -00
+00 +00 +00 +00 -00 -00 -00 -00
+00 +00 +00 +00 -00 -00 -00 -00
&d0:player 0 set direction right
&p0:
+00 +00 +00 +00 +02 -00 -00 -00
+00 +00 +00 +00 -00 -00 -00 -00
+00 +00 +00 +00 -00 -00 -00 -00
+00 +00 +00 +00 -00 -00 -00 -00
&d0:player 1 set direction None
&e:player 1 violate by illegal output of direction
&e:player 0 win
```

3. 单循环赛工具

此工具可以实现数个参赛队伍之间进行指定重复次数的先后手交替单循环比赛，并在指定文件夹中将比赛记录归档整理，同时具有对全部比赛的统计功能。

A. 说明对象

[round_match.py](#)的main函数。

B. 参数表

```
main(playerList,
      savepath = None,
      livequeue = None,
      toSave = True,
      toReport = True,
      toGet = False,
      debug = False,
      REPEAT = c.REPEAT,
      MAXTIME = c.MAXTIME,
      ROUNDS = c.ROUNDS)
```

i. 必填参数 playerList：参赛队伍的模块列表

```
playerList = [module, module, ...]
```

module 支持绝对路径、相对路径和已读取的类。

以下都是正确调用的例子。

```
>>> # 绝对路径调用两个模块
>>> main(['/Users/gaoyunhao/Documents/0ACGN/sessdsa.2048-master/src/player.py',
         '/Users/gaoyunhao/Documents/0ACGN/sessdsa.2048-master/src/player.py'])
>>>
>>> # 相对路径从父目录调用三个模块
>>> main(['../player.py' for _ in range(3)])
>>>
>>> # 调用已读取的类
>>> from player import Player as p
>>> main(['player.py', p])
>>>
```

如果将某个module改为元组(module,time0)，将指定该模块计时零点为time0（此值默认为0），通过调整计时零点可以延长或缩短对特定模块的时间限制。使用的例子如下。

```
>>> # 将第二个模块的时间限制延长30秒
>>> main(['player.py', ('player.py', -30)])
>>>
```

ii. 可选参数 `savepath` : 比赛文件夹的保存路径

`savepath` 支持绝对路径、相对路径和函数返回值；当参数缺省时，在当前文件夹下生成以当前时间命名的比赛文件夹。

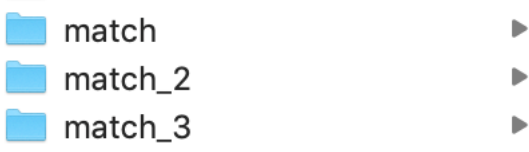
以下都是正确调用的例子。

```
>>> # 缺省调用
>>> main(['player.py', 'player.py'])
>>>
>>> # 路径调用，在父目录生成指定名字match的文件夹
>>> main(['player.py', 'player.py'], savepath = '../match')
>>>
>>> # 函数返回值调用，函数须没有参数，且返回路径字符串
>>> # 例如在当前文件夹下生成随机数文件夹
>>> import random
>>> def x(): return str(random.random())

>>> main(['player.py', 'player.py'], savepath = x)
>>>
```

如果该路径下已经存在同名文件夹，将创建文件夹副本以存放比赛记录。

右图所示是match文件夹名
被两次占用后，产生副本
match_2，match_3的情形。



iii. 可选参数 `livequeue` : 直播中线程通信用的队列

模块`queue`中的`Queue`对象，缺省值为`None`，表示直播模式关闭。此队列需要在直播工具中用到，**对抗算法开发者无需手动设置**；赛制工具开发者可以在设定通信队列之后接入即可。

```
>>> q = __import__('queue').Queue()
>>> main(['player.py', 'player.py'], livequeue = q)
```

为了保证直播观看体验，在比赛中设置了等待时间，所以运行总时间将会很长。

iv. 可选参数 `toSave` : 是否保存单局详细对局记录文件

`bool`类型，缺省值为`True`。

v. 可选参数 `toReport` : 是否生成全部比赛统计报告

`bool`类型，缺省值为`True`。

- vi. 可选参数 toGet：是否返回全部对战平台对象
bool类型，缺省值为False。
- vii. 可选参数 debug：是否生成报错信息报告
bool类型，缺省值为False。
- viii. 可选参数 REPEAT：单循环轮数
int类型，缺省值为默认单循环轮数
- ix. 可选参数 MAXTIME：最大时间限制
int类型，缺省值为默认最大时间限制
- x. 可选参数 ROUNDS：总回合数
int类型，缺省值为默认总回合数

一点提示，如果需要保存某些特定的参数组合，可以使用字典传入参数。

```
>>> myParams = {'savepath': 'myMatch',  
                'debug': True,  
                'MAXTIME': 1}  
>>> main(['player.py', 'player.py'], **myParams)  
>>>
```

C. 返回值

一般情况下，函数没有返回值。

若toGet参数被设置为True，函数将返回全部比赛的平台对象，遵循result的数据协议。

D. 数据协议

i. result

result = {(i,j): [platform1, platform2, ...], ...}

有序数对 (i,j) 表示 playerList 中下标为 i,j 的模块分别作为先后手。

一个数对对应的值为多轮比赛的平台对象列表（长度为REPEAT）。

ii. 全部比赛统计报告

在比赛文件夹中保存为 _txt 文件，
包含全部比赛的统计信息。

作为例子，某次统计报告（部分）
如右图所示。

首先是按每局比赛的统计报告。从
上而下分别记录了比赛总局数，每
局比赛的名称、比赛队伍与实际回
合数。

之后是按参赛模块的表现统计报
告。包括其路径（或类本身），在
先手方的平均耗时、胜率、胜利场
次、失败场次、违规场次、超时场
次、报错场次、对应比赛名称，以
及在后手方的相应信息。

根据此报告，再在比赛文件夹中找
到对应的详细比赛记录，可以分析
参赛模块的表现，有利于开发。

```
=====
total matches: 20
name: 1262178350798472901 -> player0 to player1 -> 117 rounds
name: 1274676031437932229 -> player1 to player0 -> 1 rounds
name: 1541577862427969221 -> player0 to player1 -> 152 rounds
name: 1556360287612831429 -> player1 to player0 -> 1 rounds
name: 1678772701430286021 -> player0 to player1 -> 74 rounds
name: 1687730834651481797 -> player1 to player0 -> 1 rounds
name: 1896198321837967045 -> player0 to player1 -> 124 rounds
name: 1909794623684871877 -> player1 to player0 -> 1 rounds
name: 2111666558231643845 -> player0 to player1 -> 123 rounds
name: 2289035844035020485 -> player1 to player0 -> 103 rounds
name: 115209882593920710 -> player0 to player1 -> 76 rounds
name: 123944510452401862 -> player1 to player0 -> 1 rounds
name: 126181201229645510 -> player0 to player1 -> 1 rounds
name: 128600249601623750 -> player1 to player0 -> 1 rounds
name: 130755131374831302 -> player0 to player1 -> 1 rounds
name: 133428922364724934 -> player1 to player0 -> 1 rounds
name: 135696615201048262 -> player0 to player1 -> 1 rounds
name: 138233595913832134 -> player1 to player0 -> 1 rounds
name: 357446505213724358 -> player0 to player1 -> 133 rounds
name: 370845565249390278 -> player1 to player0 -> 1 rounds
=====
player0 from path player.py

offensive cases:
average time: 0.006
win rate: 80.00%
win: 8 at
1541577862427969221
1678772701430286021
1896198321837967045
2111666558231643845
126181201229645510
130755131374831302
135696615201048262
357446505213724358
lose: 2 at
1262178350798472901
115209882593920710
violate: 2 at
1262178350798472901
115209882593920710
timeout: 0 at

error: 0 at

defensive cases:
average time: 0.001
win rate: 0.00%
win: 0 at

lose: 10 at
1274676031437932229
1556360287612831429
1687730834651481797
1909794623684871877
2289035844035020485
123944510452401862
128600249601623750
133428922364724934
138233595913832134
370845565249390278
violate: 10 at
1274676031437932229
1556360287612831429
1687730834651481797
1909794623684871877
2289035844035020485
```

iii. 报错信息报告

在比赛文件夹中保存
为 _Exceptions.txt 文
件，包含每局比赛的
报错信息。如果没有
报错，将输出pass。
某次报错信息报告
（部分）如图所示。

```
-> 316944209041966082

offensive:
pass

defensive:
Traceback (most recent call last):
  File "/Users/gaoyunhao/Documents/0ACGN/sessdsa.2048-master/src/plat.py", line 52, in wrappedFunc
    result = func(*args, **kwargs)
  File "/Users/gaoyunhao/Documents/0ACGN/sessdsa.2048-master/1.py", line 29, in __init__
    raise Exception
Exception
-> 318375648711970818

offensive:
Traceback (most recent call last):
  File "/Users/gaoyunhao/Documents/0ACGN/sessdsa.2048-master/src/plat.py", line 52, in wrappedFunc
    result = func(*args, **kwargs)
  File "/Users/gaoyunhao/Documents/0ACGN/sessdsa.2048-master/1.py", line 29, in __init__
    raise Exception
Exception

defensive:
pass
```

4. 本地直播工具

此工具可以实现比赛的实时本地直播，使比赛更有观赏性，对于开发和调试用处并不很大。

A. 说明对象

[livetool.py](#)的Live类

B. 简介

此类唯一提供的外部调用接口为`__init__`，所以接下来只介绍`__init__`的参数表；运行时，只需如示实例化即可观看直播。

```
>>> Live()
<__main__.Live object .!live>
>>>
```

C. 参数表

```
__init__(self,
           mode = 0,
           func = __import__('round_match').main,
           playerList = ['player.py' for _ in range(2)])
```

i. 可选参数 mode：可视化版本

为 0 或 1 分别对应 2048 原版可视化和[柚子社版](#)可视化，缺省值为 0。

ii. 可选参数 func：赛制工具

目前开发的只有单循环赛工具可用，缺省值为单循环赛工具。

iii. 可选参数 playerList：参赛队伍的模块列表

遵循与单循环赛工具相同的约定，缺省值为`['player.py' for _ in range(2)]`，即两个样例随机ai比赛。

需要提醒的是，由于tkinter对不同系统的兼容性并不好，而编写者使用的是macOS，所以此UI可能在win界面显示异常。

5. 对抗算法开发者工具

为了方便对抗算法开发，我们编制了面向对抗算法开发者的工具。

A. 逻辑约定

i. 坐标

坐标为tuple变量，具有结构(row, column)，row行坐标，从上到下为 0 到 3 的int变量；column列，从左到右为 0 到 7 的int变量。

ii. 方向

方向为int变量，0、1、2、3 分别代表上、下、左、右。

iii. 归属

归属为bool变量，True为先手方，False为后手方。

B. ai接口规范化

参赛队伍的ai须以以下方式实现。

实现一个Player类，并实现两个方法。

i. `__init__(self, isFirst, array)` 初始化

1. 参数：isFirst是否先手，为bool变量。
2. 参数：array随机序列，为tuple变量。

ii. `output(self, currentRound, board, mode)` 给出己方的决策

1. 参数：currentRound 当前轮数，与对战平台类变量self.currentRound 遵循相同约定。
2. 参数：board 棋盘对象的拷贝，是Chessboard类的实例。
3. 参数：mode 模式。
mode = 'position' 对应位置模式，mode = 'direction' 对应方向模式。
4. 返回：位置模式下返回下棋的坐标，方向模式下返回合并的方向。

C. 棋盘对象Chessboard

棋盘对象提供了对棋盘的基本操作接口，参赛ai可以直接调用。接口如下

i. `__init__(self, array)` 初始化棋盘对象

参数：array随机序列

ii. `add(self, belong, position, value = 1)` 在指定位置下棋

参数：belong操作者，position位置坐标，value 下棋的级别，缺省值为 1

注：棋子的数值取以 2 为底的对数即为其级别，为int变量。

- iii. `move(self, belong, direction)` 向指定方向合并，并且返回棋盘是否变化
参数：belong操作者，direction合并方向
返回：棋盘是否变化，为bool类型
- iv. `getBelong(self, position)` 根据位置得到归属
参数：position位置坐标
返回：该位置上的棋子的归属，若为空位，则返回空位的归属
- v. `getValue(self, position)` 根据位置得到级别
参数：position位置坐标
返回：该位置上的棋子的级别，若为空位，则返回 0
- vi. `getScore(self, belong)` 获取某方的全部棋子的级别列表
参数：belong某方
返回：由该方全部棋子的级别构成的list类型变量
- vii. `getNone(self, belong)` 获取某方的全部空位的位置列表
参数：belong某方
返回：由该方全部空位的位置构成的list类型变量
- viii. `getNext(self, belong, currentRound)` 获取某方在本方领域允许下棋的位置
参数：belong某方，currentRound 当前轮数
返回：该方在本方领域允许下棋的位置，若不可下棋，返回空元组
- ix. `copy(self)` 返回一个对象拷贝
返回：深拷贝的棋盘对象
- x. `__repr__(self)` 和 `__str__(self)` 棋盘的打印字符串
打印结果如图所示。

```
+00 +02 +01 +02 -00 -00 -00 -00
+00 +00 +00 +01 -00 -00 -00 -01
+00 +00 +00 +00 +02 -00 -00 -02
+00 +00 +00 +00 -00 -00 -02 -01
```

D. 复盘可视化analyser

你一定很好奇为什么整个技术组都把这个词拼错了

解析对局记录文件并可视化复盘，为调试提供方便。

采用交互式设计，直接运行即可，根据终端提示依次输入对局文件名和可视化版本，即可进入可视化界面。

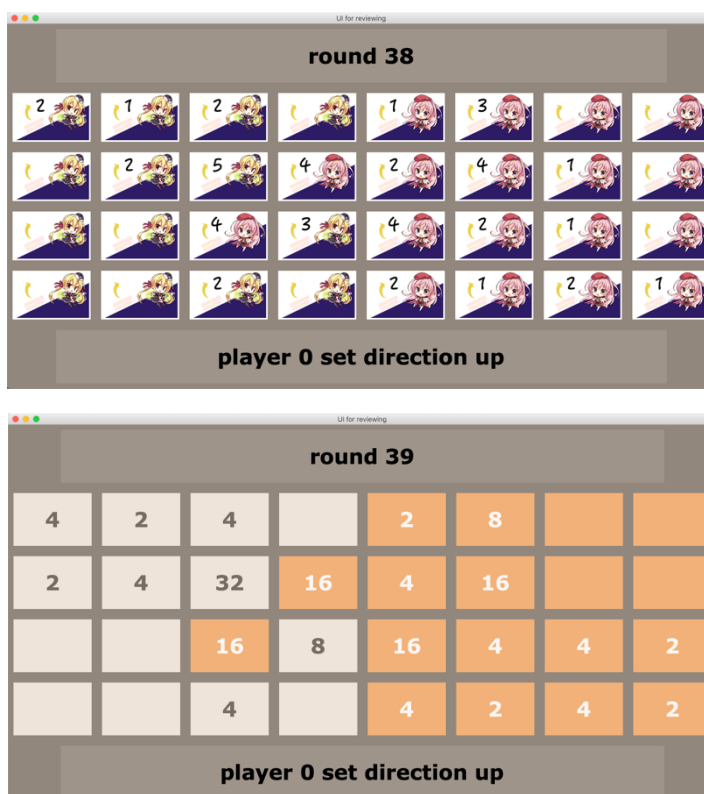
使用示例如下。

```
filename: 1954264233434154401
=====
enter 0 for original mode.
enter 1 for YuzuSoft mode.
mode: 0
=====
player0: 1 from path player.py
player1: 0 from path player.py
time: 2020-05-05 19:11:10
=====
| timeout | violator | error | winner |
-----
|   None  | player 1 |  None | player 0 |
=====
=====
|player| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|
-----
|   0| 5| 4| 1| 1| 0| 0| 1| 1| 0| 0| 0| 0| 0|
-----
|   1| 3| 2| 5| 2| 0| 1| 1| 0| 0| 0| 0| 0| 0|
=====
```

UI界面如示。

按键盘的 '[' 和 ']' 分别
对应前进、后退操作。

需要提醒的是，由于
tkinter对不同系统的兼
容性并不好，而编写者
使用的是macOS，所以
此UI可能在win界面显
示异常。



6. 编辑历史

- 2020.5.5 @SophieARG 创立文档