SDK使用文档

SDK使用文档
一、如何开始编写AI
二、AI可以调用的参数或接口
三、可以参考的AI
四、如何进行本地测试
五、如何进行本地调试
六、录像文件在哪里

一、如何开始编写AI

在主目录下找到player.hpp文件，在里面补充完整以下函数

```
void getOperations(Parameters* parameters,
       State* state,
       Operations* opt){
}
```

然后用这个文件替换AI/src/player.hpp，你就完成了AI的编写了

二、AI可以调用的参数或接口

在**getOperations**函数下面有详细的注释，可以参考注释来编写您的代码，下面是结合该注释做的一些补充说明

```
const int WIDTH = 10;
const int HEIGHT = 10;
const int MAXRANGENUM = 3;
const int PLAYER = 2;
  以上是可以调用的全局常量，前三个与parameters里的mapWidth，mapHeight，maxRangeNum一一对应，最后一个是玩家数量，玩家编号从0开始

@parameters
   这是跟游戏相关的常量，不会随着回合的变化而变化
  member variables(public):
   AI编号：int num;
   地图宽度：int mapWidth;
   地图长度：int mapHeight;
   初始地价：int landPrice;
   病毒数目：int pollutionComponentNum;
   最大回合数：int maxRoundNum;
   覆盖范围种类数：int maxRangeNum;
```

```
    情报价格: int tipsterCost;
    建筑物: std::vector<std::pair<int,int>>buildings;
    治理设备各种范围价格: std::vector<int>processorRangeCost;
    治理设备各种类型价格: std::vector<int>processorTypeCost;
    检测设备各种范围价格: std::vector<int>detectorRangeCost;
    病毒治理收入: std::vector<int>pollutionProfit;

 @state
  member variables(public):
     行动AI编号: int num;
      双方钱数: int money[PLAYER];
      双方分数: int score[PLAYER];
      疫区: int pollution[WIDTH][HEIGHT];
      地皮情况: Land lands[WIDTH][HEIGHT];
      放置检测设备情况: std::vector<Detector>detectors;
      放置治理设备情况: std::vector<Processor>processors;

      以下变量是关于上两回合己方和敌方AI的操作回馈，如果上两回合无该操作，则对应的容器大小为0或者变量为-1，注意游戏不是全知信息，因此在操作回馈中只回馈己方AI可以得知的信息
      我方情报贩子的中心点:int tipsterX, tipsterY;
      我方情报贩子侦测情报位置和疫情:int tipsterCheckX, tipsterCheckY, tipsterCheckPollution;
      我方检测设备放置位置和范围类型:int myDetectorX, myDetectorY, myDetectorRange;
      我方治理设备放置位置和范围类型、治理病毒类型:
       int myProcessorX, myProcessorY, myProcessorRange, myProcessorType;
      我方标价位置和标价:int myBidX, myBidY, myBidPrice;
      对方标价位置和标价:int otherBidX, otherBidY, otherBidPrice;
      对方检测设备放置位置和范围类型:int otherDetectorX, otherDetectorY, otherDetectorRange;
      对方治理设备放置位置,范围类型,治理病毒类型:int otherProcessorX, otherProcessorY, otherProcessorRange, otherProcessorType;
      我方检测设备侦测到的疫区位置:std::vector<std::pair<int,int>>myDetectorCheckPos;
      我方检测设备侦测到的疫区病毒组成:std::vector<int>myDetectorCheckPollution;
      我方获得收益的点:std::vector<std::pair<int,int>>profitPos;

 @opt
    这是策略函数，您应当在确定AI的策略后调用它，当您结束getOperaions函数的时候，您的操作会自动发送给游戏逻辑。注意在未结束getOperations函数前，您调用以下函数会覆盖上一次调用该函数时使用的策略；四个函数每个回合均可调用。

  member functions(public):
   @x 获取情报的中心位置x坐标,对应mapWidth那一维,下标从0开始,以下同理
   @y 获取情报的中心位置y坐标,对应mapHeight那一维,下标从0开始,以下同理
   void setTipster(int x, int y):设置本回合使用情报的中心位置，如果本回合不使用，请勿调用
```

```
    @x 放置检测设备的位置x坐标
    @y 放置检测设备的位置y坐标
    @rangeType 放置检测设备的检测范围类型,对应maxRangeNum,下标从0开始,以下同理
        void setDetector(int x, int y, int rangeType):设置本回合放置的检测设备,如果本回合不使用，请勿调用
        @x 放置治理设备的位置x坐标
        @y 放置治理设备的位置y坐标
        @rangeType 放置治理设备的治理范围类型
        @processingType 放置治理设备的治理病毒类型,对应pollutionComponentNum,下标从0开始
        void setProcessor(int x, int y, int rangeType, int processingType):设置本回合放置的治理设备,
        如果本回合不使用，请勿调用
    @x 地皮竞价的位置x坐标
    @y 地皮竞价的位置y坐标
    @bidPrice 本回合对该地皮的报价，要求bidPrice大于上一次报价且为landPrice*0.1的整数倍
        void setBid(int x, int y, int bidPrice):设置本回合的地皮报价信息，如果本回合不使用，请勿调用
     @覆盖类型:DeltaPos = [
        [(0,0),(0,1),(0,-1),(1,0),(-1,0),(0,2),(0,-2),(2,0),(-2,0)],      十字
        [(0,0),(0,1),(0,-1),(1,0),(-1,0),(1,1),(1,-1),(-1,1),(-1,-1)],    区域
        [(0,0),(1,1),(1,-1),(-1,1),(-1,-1),(2,2),(2,-2),(-2,2),(-2,-2)], 斜十字
    ]
```

以下是使用过程中您可能需要了解的一些类的参数的定义

```cpp
class Processor{
public:
 std::pair<int, int>pos; //位置
 int rangeType; //范围类型
 int processingType; //治理类型
 int owner; //持有AI编号
};

class Detector{
public:
 std::pair<int,int>pos; //位置
 int rangeType; //范围类型
 int owner; //持有AI编号
};

class Land{
public:
 int owner;//持有人，无持有人时为-1
 int occupied;//土地上是否有治理设备，没有为0
 int bid;//土地的最高竞价，初始值为土地价格-1
 int bidder;//当前最高价出价人，如果没有人出过价，为-1
```

```
 int round;//最高价出价持续的回合数
 int bidOnly;//是否发生流拍，-1表示没有，否则为流拍方的对手编号
 int filled;//土地上是否有检测设备，没有为0
};
```

三、可以参考的AI

在您没有替换掉AI/src/player.hpp之前，该路径下的文件是一个比较简单的智能水平较弱的样例AI，对于四种操作分成四个部分编写，您可以参考该AI的实现来弄清楚上面的各个参数和接口的使用方法。此外，该AI还可以用来帮助您测试您AI的强度。

四、如何进行本地测试

在您用player.hpp替换掉AI/src/player.hpp后，请确保您本地cmake的版本高于3.15.4，如果您没有安装cmake，您可以根据自己的本地环境安装cmake，或者手动编写makefile，或者新建vscode工程，将代码copy进去进行编译等等等等。此外，在编译之前，您需要确保您的本地环境中有C++语言的编译器，除AI部分外的代码中未使用C++11及以上的新特性。

使用cmake编译的指令的如下，首先您需要保证您在AI目录下:

```
cmake CMakeLists.txt
```

这时候在AI目录下理应出现makefile文件，然后执行：

```
make
```

这时候在AI目录下理应出现sample_ai，这就是编译出来的可执行文件

然后切换到游戏逻辑所在的目录，即main.py所在的目录，在原目录结构下是AI的上级目录，执行指令格式如下

```
python main.py <AI0的路径> <AI1的路径>
```

这里请确保您的python指令调用的python解释器的版本为python3.6及以上，python3.6以下未经过测试，python2绝对不可用。AI0和AI1的路径即其可执行文件的路径，可以是相对路径也可以是绝对路径，比如以下示例：

```
python main.py ./AI/sample_ai ./AI/sample_ai
```

之后会执行本地评测，您只需要稍做等待，即可在控制台上看到最终结果，即两方的分数，如果有一方的AI出现了崩溃异常或者超时，控制台上会输出比分0 1或者1 0，0代表逻辑最先检测到出现问题的AI编号

五、如何进行本地调试

以上方法可以简单的评测AI对战，但是并不利于调试，由于在评测的过程中，AI的stdin和stdout即标准输入输出都被重定向到了管道里，因此您不能通过在stdout里输出来进行调试，相反，这会造成您的AI出现错误，因为游戏逻辑会接受到您的输出，并对您的输出做格式检查。

因此，您有以下几种输出调试方案：

- 将AI需要输出的消息输出到文件里，在AI/src/main.cpp中有一个很简单的示例，您只需要把这一行

```
#define DEBUG 0
```

改成

```
#define DEBUG 1
```

即可在log0.txt或者log1.txt(取决于本局比赛您的AI的编号)看到一些输出出来的信息，您可以查看AI/include/client/client.cpp获取更多输出信息。此外，您还可以通过修改client.cpp和main.cpp来自定义您的输出格式来方便您的调试

- 在逻辑中输出中间变量进行调试

由于逻辑是用python进行编写的，因此可能需要您对python基本语法有一定的了解，在简要的阅读完逻辑的代码之后，您可以在合适的位置输出log信息来进行调试，这主要用来解决一些通信上的问题

此外，如果您不喜欢使用输出调试方案，您还可以使用现有流行的调试工具如gdb等进行调试，这需要您对这些调试工具的使用有一定的了解。

六、录像文件在哪里

在您评测完一局对局后，录像文件默认为replay.json，生成在和main.py同级的目录下，如果您不希望它生成在这里，您可以自行修改评测脚本

最后，感谢您的阅读！如果您有任何问题，欢迎在选手群里进行提问！