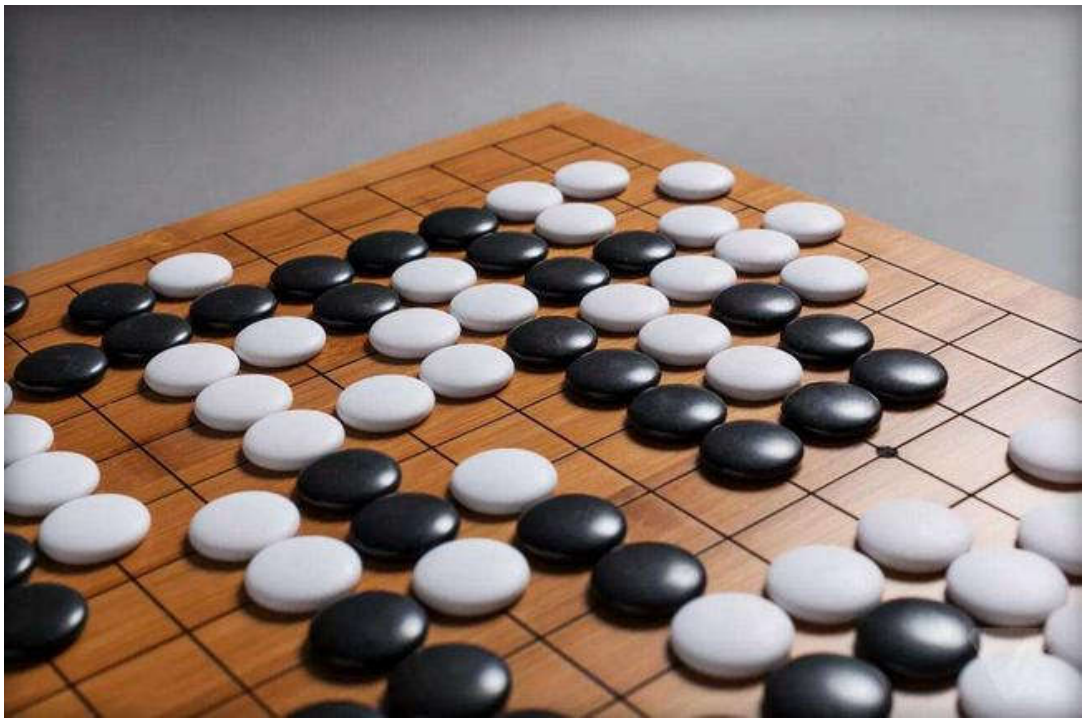
人工智能围棋



1. 简介

在本编程作业中，你将基于一些已学习过或独立研发的 AI 算法开发自己的 AI 智能围棋程序，通过使用多种最优搜索和强化学习的算法在一个缩小版的围棋游戏中进行对战。相比于正常围棋，在本次对战中使用的围棋棋盘尺寸缩小为 5x5（正常围棋尺寸为 19x19）。你的目标是使用你的智能围棋程序和你的同学进行竞赛，通过使用不同的搜索算法，实现最优的围棋策略，打败你的竞争对手们，本次竞赛将根据你的算法表现以及竞赛成绩进行打分，你打败的对手越多，程序表现越好，则将取得更好的成绩，加油！

2. 游戏规则描述

围棋是一种非常抽象的棋盘策略游戏，每一局有两个对战选手，分别执黑子和白子。对战双方的目标是围住更多的对方棋手的棋子，在棋盘上占领更多的领土获胜。围棋的基本概念非常简单：

- 对战玩家：围棋由两个玩家对战，分别为黑棋棋手和白棋棋手
- 棋盘：围棋游戏的主战场，由横纵穿插的方格网线组成。标准的棋盘是 19x19 大小，但在本项目中，棋盘的大小调整为 5x5，以便适应个人电脑的计算能力。
- 点：围棋棋盘上各个交叉点称之为**点，**棋盘上除了网格线的交叉点，还有棋盘边缘和

四角顶点，这些点是棋手落子的地方。

- 黑白棋子：黑棋棋手执黑子，白棋棋手执白子。

本项目中的游戏规则相比于真实围棋游戏稍有变动，规则如下，也很简单：

- 游戏开始于一个空棋盘，棋盘大小为 5x5
- 两个棋手轮流落子，直到一方胜利
- 棋手可以选择任意未落子的空点进行落子（除了那些被”打劫”规则或者“气”规则限制的位置）
- 在游戏开始后，除非被对方棋手捕获的棋子（“提子”）可以被清理出棋盘，其余已经落下的黑白棋子不可以再变动。

本围棋游戏建立在两个简单规则之上，即围棋的“气”规则和“打劫”规则。

1) 规则 1：围棋的“气”

下棋时，对弈双方各执一种颜色的棋子，轮流将一枚棋子放置于交叉点上。与棋子直线相连的空白交叉点叫做气。当这些气都被对方棋子占据后，该棋子就没有了“气”，要被从棋盘上提掉。如果棋子的相邻（仅上下左右）直线交叉点上有了同色的棋子，则这两个棋子被叫做相连的。任意多个棋子可以以此方式联成一体，连成一体的棋子的气的数目是所有组成这块棋的单个棋子气数之和。如果这些气都被异色棋子占领，这块棋子就要被一起提掉，即被移出棋盘。

基于围棋的气规则，对战玩家被禁止投掷“自杀“棋子，即玩家不能够在没有气的点上落子抑或落无气之子，除非这样做时可以立即将对手的棋子捕获移除，空出点位形成气。

例 1:

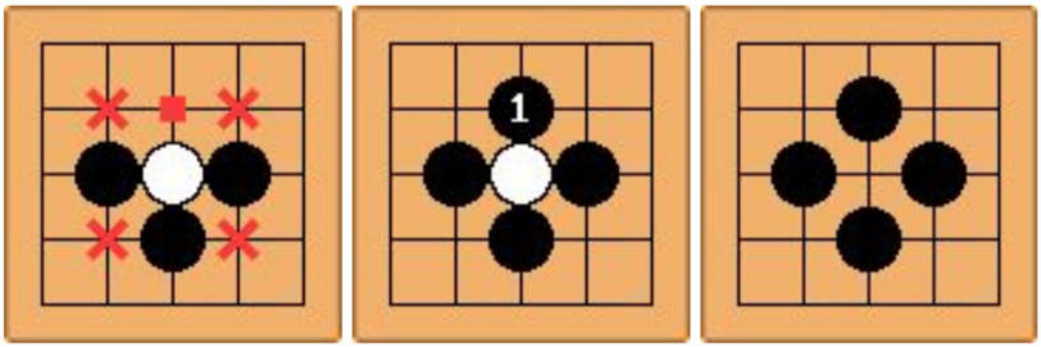


图 1 单棋子捕获

在图 1 中，白棋被**黑棋 1** 所捕获，因为白棋周围所有的正交点都被占据了。

例 2:

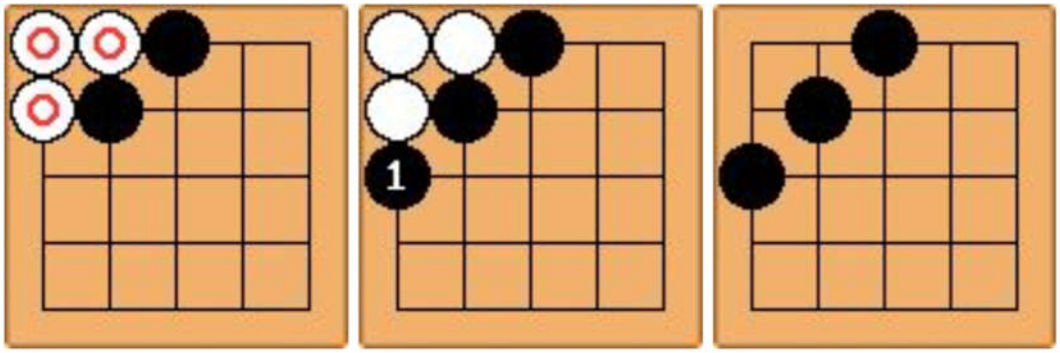


图 2 群体捕获

图 2 中，所有在边缘的白棋被黑棋捕获，因为其连接群没有了气。

例 3:

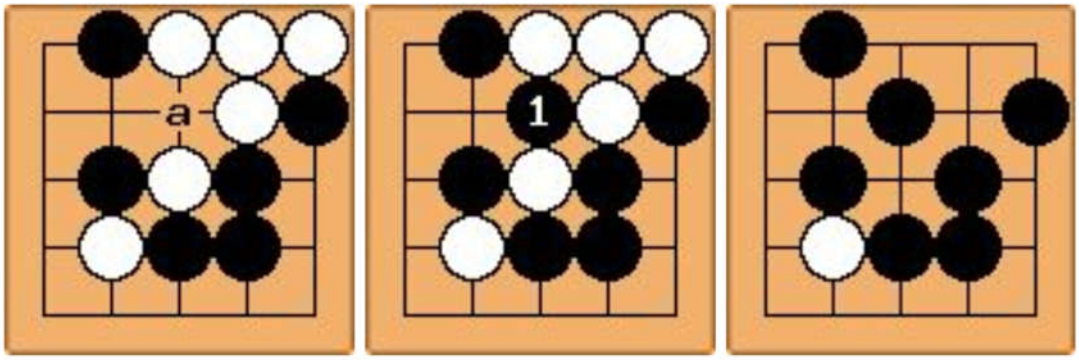


图 3 多个群体同时捕获

在图 3 中，黑棋 1 的落子使得两个部分的白棋被同时捕获并移出棋盘。

例 4:

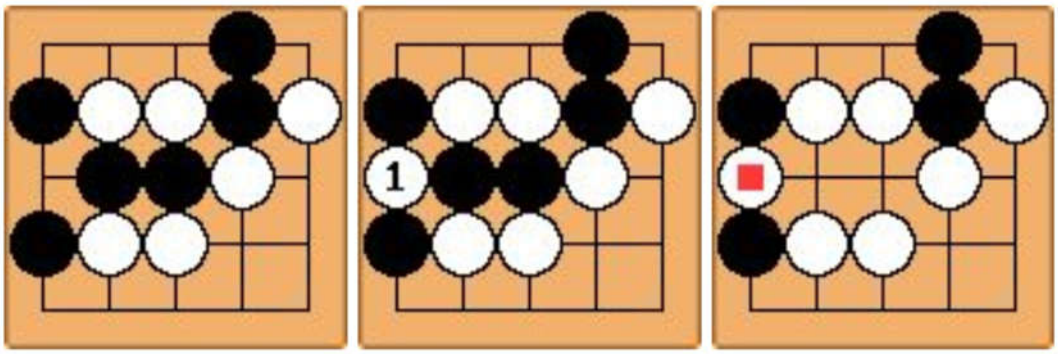


图 4 击杀棋子落点

在图 4 中，白棋 1 的落子两个黑棋棋子被捕获，并由于黑棋棋子的移除，使得落子的白棋拥有的气，这种情况的落子是被允许的。

2) 规则 2：“打劫“规则

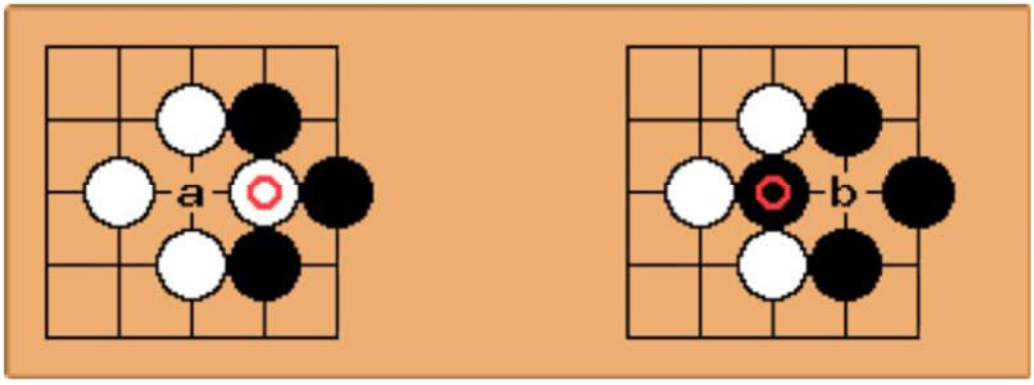


图 5 劫规则

在图 5 所示左侧棋盘中，黑棋可以通过落子到 a 处实现对 b 位置的白棋（带红圈的白棋）的捕获，此时，由于白棋棋子被捕获，a 处的黑棋获得了气；同理，在白棋落子时，可以通过落子到 b 处实现对 a 处黑子的捕获（图 5 右侧棋盘），使得棋盘从右侧局面回到左侧，如此循环往复，使得整个棋盘不能分出胜负，这种情况在围棋中被称之为“劫“。

针对劫这种特殊情况，打劫规则规定，在棋局中一旦出现了“劫“，即一方玩家捕获了劫子后，另一方玩家不能够立刻选择再次捕获劫子。

例子：

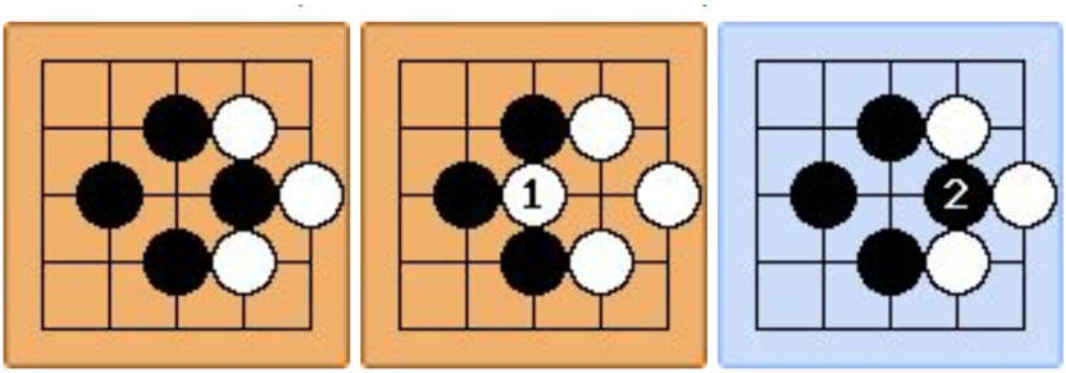


图 6 打劫规则

在图 6 左图形成的劫局面上，黑子被**白子 1** 捕获后，由于“打劫规则“的存在，**黑子 2** 不能够立刻落子再次捕捉白子 1，黑棋棋手在本轮落子时必须选择在其他地方落子，等待下一轮落子时才可以再次选择捕捉白子 1（如果劫依然存在的话）。

劫不仅可以存在于棋盘的中央区域，也可以存在于棋盘的边缘，如图 7 所示。

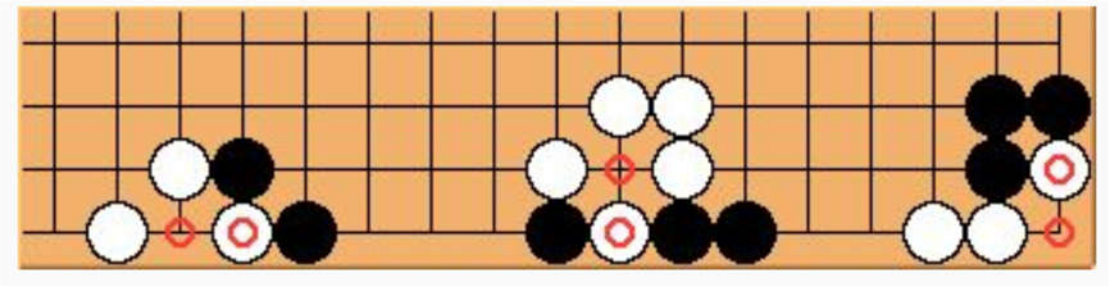


图 7 边缘劫子

3) 贴目：

因为黑棋有第一步的优势，所以判给白棋一定的补偿，即**贴目**；在比赛结束时给了白棋一个补偿分数。 在本作业中棋盘大小为 5x5，白色玩家的贴目设置为 5/2 = 2.5

4) 放弃落子：

棋手可以选择在特殊情况下放弃该轮落子的权利，在本游戏中即放弃掉当前一步落子，玩家可以选择任何一步进行放弃落子，以获取潜在的某些优势。

5) 游戏结束条件：

当一轮棋局满足下列四种情况中任意一种时即视为游戏结束：

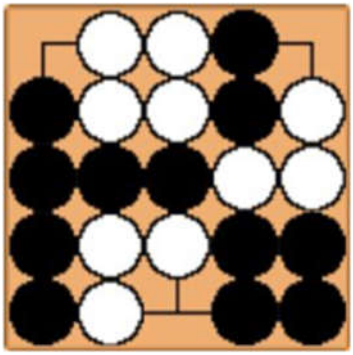
1. 当一方玩家落子时间超时时游戏结束，该轮游戏被判负。
2. 当一方玩家非法落子时游戏结束（非法落子、自杀落子、违反打劫规则、违反气规则等）。
3. 当双方玩家同时放弃落子游戏结束。
4. 当该局游戏的最大步数达到时游戏结束。

6) 获胜条件：

在真实围棋中有很多种情况可以使一方棋手获胜，但是在本项目中，我们基于下列几种情况得分作为最终的获胜条件：

- 占位得分：一方玩家的占位得分是该玩家在棋盘上所占据的点的个数
- 最终得分：黑棋玩家的最终得分即为其占位得分，白棋玩家的最终得分由“占位得分“+”贴目“分数组成。
- 获胜判定：
  - ✓ 如果一方玩家落下非法棋子（投掷“自杀”棋子，违背“打劫”规则等...则被判负）
  - ✓ 如果在一盘棋局中达到了最大步数，或者双方玩家都选择放弃落子，则最终得分

较高一方获胜，如下图所示，黑子占位得分 10 分，白子占位得分 10 分，但白子获得贴目得分 2.5 分，故白子最终得分为 12.5 分，白子获胜。



特殊说明，为适应本次作业目的，本游戏规则相比于正式围棋规则有所变化，比如在正式围棋中得分的计算方式为棋子数量加上围绕一方棋子的空点数量，在本游戏中省略该计分方式使得游戏在缩小版本的棋盘上更加公平合理，感兴趣的同学可以在本次作业后研究真实版围棋，并修改程序，使得游戏更具挑战性。

3. 游戏对战

在本次作业中，各位同学的程序会与其他同学的程序进行对战，同时，也会对一个随机下棋程序进行对战，胜率越高则得分越高。

4. 程序架构

本部分内容将阐述本次作业程序的基本架构。本次智能围棋程序由一个主持程序 host.py 进行对两个智能下棋程序的调用，使得两个程序可以轮番落子。主持程序将会跟踪游戏的进程，从每一方下棋程序中轮番获取双方下一步的落子，判断双方下一步的落子是否遵循游戏规则，并且将会从棋盘上去掉被捕获的棋子以及最终判定获胜方。需要注意的是，双方下棋程序必须准确的按照规定的格式输出自己下一步棋子的位置，并保存在一个命名为 output.txt 的文件中交付给 host.py 程序。总体而言，每一方程序需要做的事情很简单，首先从 host.py 处获取当前对方落子后的棋盘状态，这个状态由一个 input.txt 文件保存，然后输出自己下一步落子后的棋盘状态给 host.py，即输出一个 output.txt 文件给 host.py。
程序架构如图 8 所示：

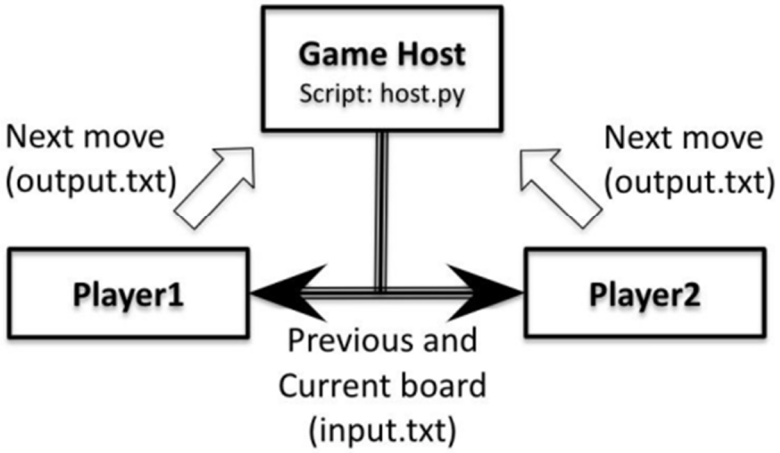


图 8 程序架构

1) 游戏参数

下列参数将在本游戏程序中使用:

- 在棋盘上，0 代表空点，1 代表黑棋，2 代表白棋
- 可视化输出中，X 代表黑棋，O 代表白棋
- 下棋顺序为黑先白后，所有游戏中总是如此
- 棋盘大小为 5x5
- 每局游戏最大步数为 5x5-1=24 步
- 对白棋的贴目分数为 5/2=2.5

2) 游戏棋盘

	j=0	j=1	j=2	j=3	j=4
i=0	0	0	1	1	0
i=i	0	0	2	1	0
i=2	0	0	2	0	0
i=3	0	2	0	0	0
i=4	0	0	0	0	0

		X	X	
		O	X	
		O		
	O			

图 9 游戏棋盘格式

如图 9 所示，host.py 程序将会轮流调用双方下棋程序在图中棋盘上进行对战。棋盘将使用**以 0 为第一个坐标的坐标系**，即左上角第一个位置坐标为(0，0)。(0，4）为右上角的坐标

点，（4，0）为左下角坐标点，（4，4）为右下角坐标点。在棋局程序中，使用 1 代表黑棋，2 代表白棋，0 代表空点。在打印输出中，X 为黑棋，O 为白棋（host.py 程序将会打印输出棋局的每一步）。

3) 机器人对手

针对本次作业，我们将会推出不同的简单机器人对手来和你的机器人进行对战，这些机器人对手的难易程度不同，你打败他们的胜率将决定你的基础成绩。
这些机器人对手包括：

- Random palyer:随机机器人，在满足要求的空点随机落子
- Alpha-beta player：使用 minimax 算法进行下棋的机器人

在你完全打败这些 AI 机器人后，你将和你的同学进行对战，你可以选择任何算法来武装你的程序，将它变得更加强大，你也可以选择使用一些机器学习算法来训练一个智能型的 AI 程序，比如使用 Qlearning 算法通过训练一个 Q-agent 来进行对战，但前提是输入输出必须符合游戏规则，更加重要的是这些程序代码必须由你自己编写，不能够从其他地方获取，抑或是调用第三方已经编写好的训练模型和算法（辅助程序除外），一旦发现代码有抄袭行为，将会被视为作弊。

4) 代码规范

在本次作业中，你将命名你自己的程序为 my_palyer.py，使用 python3 作为你的编写语言。
在游戏过程中，host.py 将会完成下列步骤：
循环直到游戏结束：

- 交替更改当前下棋的玩家
- 清理掉当前目录中的 input.txt 文件和 output.txt 文件
- 创造一个新的 input.txt 文件，其中记录了上一步棋盘状态和当前棋盘状态
- 调用本步棋手的程序，本步棋手程序应该读取 input.txt 并产生一个 output.txt 文件
- 验证新的 output.txt，检测是否有格式错误或者其所记录的新一步落子点是否违反游戏规则，如果有格式错误或者所落棋子违反了游戏规则，则将本局该棋手判负
- 检查是否所有游戏棋局结束，如果游戏结束则获胜者将会被公布。

为了帮助你更好的开始编程，本次作业将会把 host.py 程序和一个 random.py 程序给你，还有一个脚本程序 build.sh 程序也会一并发给你，你可以复制两个 random.py 并将其中一个重新命名为 my_player.py，然后使用 **git bash**(windows)命令行在当前目录下运行·**sh build.sh** 进行测试。对于使用 MacOS 的同学，可以直接使用其自带的 terminal 命令行进行测试。此外，通过调整 build.sh 中第 96 行，可以调整测试的局数。

5) 输入输出格式：

每一步棋子落子位置的输入输出格式对于完成本次作业至关重要，你必须严格按照本节要求的输入输出格式进行落子位置的输出，否则你的程序将不可能赢得任何一场比赛。

- **输入 input.txt：你的程序必须从当前目录下读取一个 input.txt 文件来获取棋盘状态，其包括以下几条格式。**

  1. 第 1 行：一个数值“1”或者数值“2”来表明本局游戏你的棋子的颜色（黑棋=1，白棋=2）
  2. 第 2-6 行：对于前一个状态的棋盘的描述，由 5 行数字组成，每一行由 5 个数字组成，这个状态表明的是**你的上一步**下过后的棋盘状态，其中黑棋=1，白棋=2，空点=0
  3. 第 7-11 行：对于当前棋盘状态的描述，由 5 行数字组成，每一行由 5 个数字组成，这个状态表明的是**你的对手**刚刚一步下过后的状态，其中黑棋=1，白棋=2，空点=0
  4. 在最开始的时候，第 2-11 行 input.txt 文件的默认数值为 0.

- Input.txt 文件举例（**最上面的===input.txt====和最下面的=====不包含在内**）：

```
========input.txt========
2
00110
00210
00200
02000
00000
00110
00210
00200
02010
00000
=======================
```

图 10 输入 input.txt 文件例子

- **输出 output.txt 文件：为了使 host.py 知道你这一步的落子位置，你需要在你的程序中输出一个 output.txt 文件到当前目录下，其格式包含以下几条：**

  1. 输出位置由两个数字及一个逗号组成，比如

```
========output.txt=======
2,3
========================
```

如果你要放弃当前步落子，只需输出大写的 PASS:

```
========output.txt=======
PASS
========================
```

注意，文件中不包含例子中的====output.txt====和例子中的最后一行====，输出文件中只有一行。

5. 提示和帮助

- 在本次作业中，请使用 **python3** 作为编程语言，你的脚本程序应命名为 **my_player.py。**
- 在本次作业中，你只需提交你的程序，即 my_player.py，无需提交其他任何程序。
- 在本次作业中，为了更好地帮助你完成作业，将会下发 build.sh、host.py、以及一个 random.py 程序，你可以修改这些程序以帮助你理解，但最终输入输出需要满足规则要求，且最终评分会在助教版本地 host.py 中进行。
- 程序下发时会以压缩包的形式下发，其中包括上述程序脚本文件以及一个 init 文件夹，init 文件夹中包含 input.txt，请不要修改这个文件嵌套结构，直接解压压缩包到你本地电脑并进行编程。
- 为了更好的组织最终的竞赛，请控制你的程序每一步落子的时间在 10 秒以内，超出 10 秒的限制会被助教版本的 host.py 直接判负。

祝你在本次作业中取得好成绩！