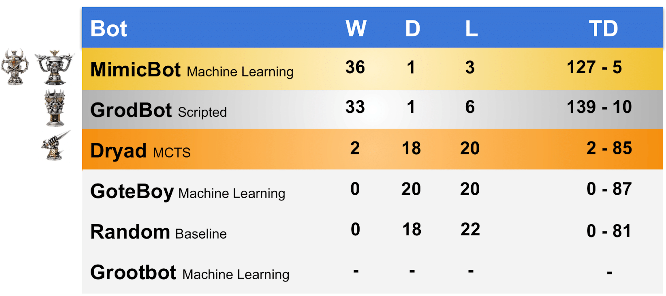
# 测试结果



* 1. 使用MCTS的一局游戏，大概需要至少3个小时以上才能分出胜负 （非常久！）。
  2. 只测试了2次，都是和’random’ 对抗。结果：第一局平局，第二局胜利。

# 游戏逻辑（防重复踩坑）

* 1. 游戏开始，客队需要进行start game这个action。如果客队是robot而不是human，则游戏自动为客队game.step 这个action。
  2. 游戏开始后，有几步action很固定，如下：

1. Start game （客队）

2. Heads / Tails （客队选）

3. Kicking / Receive (如果客队选的heads/tails和系统随机给的一模一样，则客队有攻防位置的选择权，如果客队猜错了，攻防位置就让主队选)

4. Place player / Setup formations / end setup （一局游戏要进行多次，游戏开始一次，turn的轮换也要重新选，\*别踩坑）

5 Place Ball (Kicking 方，也就是防御方选择在对方场地上放球)

6.1 Select a player (receive side，也就是进攻方有几率在对方place ball后选择这个：此时可以选择一个人原地隔空拿起进攻方放置的球)

6.2 Actions (receive side。防御方place ball后，进攻方的另外一种可能就是选择这个)，包括 start\_move, start\_blitz… start\_foul什么的。

# 游戏设置（以及可以改进的点）

* 1. MCTSNode类代表树的节点。该节点的action、state和team属性代表这个节点的父节点，game.step了这个action然后到达了这个state（等于game.Step类，用于forward()函数）。并且基于这个state下，team将执行下一步action’（这个action’属于该节点的child节点）
  2. 实现的MCTS没有考虑随机性，比如选择一个在tackle zone的player进行move，此时要掷骰子。 但MCTS代码里，只考虑了一种情况（事实上有两种，好像是被knock down ？或者成功移动）。
  3. 接上。Expansion阶段，基于当前state，一次性扩展了所有的actions。例如，在place ball后，加入现在的state的available actions 是start\_move, start\_blitz… start\_foul end turn这些，那么就一次性把这些actions全部扩展了。这种做法其实也导致了，难以考虑随机性的问题。也许可以尝试，1次expansion，从available\_actions里随机抽个action，然后仅仅扩展这1个节点（action）？（这会有重复action的情况，可能导致扩展了相同的action和state；但也有可能是相同的action和不同的state——这就考虑了随机性了我觉得）
  4. Simulation阶段，我这里是一旦一次性扩展完所有节点，就遍历这些被扩展节点——对于每一个被扩展节点都进行一个随机游戏（rollout）的做法，然后backup。
  5. Backup的reward值（q值），事实上就是胜率。从下往上递归（但我不确定这里有没有bug…，毕竟经常弄混hhh，如果觉得MCTS很弱智，可以稍微检查一下这里的逻辑）
  6. Select阶段，要根据当前节点的team是谁进行选择（如果是敌对，就得反过来：select胜率小的那个节点）
  7. 如果想升级/改进Simulation阶段，可以参考AlphaGo的做法——用一个ResNet 评估当前state的 value，而不是靠双方随机选action的结果（胜率）。
  8. 游戏时间长的问题：
     1. 可以结合一下 scriptedbot.py里的考虑。比如：当前state下的available\_actions有move和stand up 和end player turn，似乎是选move更好。那么后两个actions就不值得去expand了。或者，可以让
     2. 我的努力：（1）假如当前state下的available\_actions有use\_reroll和don’t\_use\_reroll。我默认忽略don’t\_use\_reroll（感觉use\_reroll比较好）；（2）或者选了start\_move后，available\_actions有move，end\_player\_turn, undo，我默认expansion阶段只expand “move”这个动作下的所有可能（比如move有120个位置可以选，就相当于要expand 120个不同位置选择的move） …其他努力见代码
  9. 如果要忽略一些没有必要的动作，或者增加一些启发式的方法。mctsbot的act()函数，mcts里的expand()和simulate()都最好能同步进行。
  10. 设置阵型：Setup formation。 一局游戏经常要进行好几次（N.B. 我默认忽略place player），但是它的设置很奇葩 （要从 Place player / Setup formations / end setup这里选）：需要先选择某个formations，下一步再选择end setup。如果不这样做，很可能出现这种情况————（Setup formations，Setup formations，Setup formations，Setup formations…），或者（end setup ，Setup formations ，end setup ，end setup…）要花一点时间和运气才能出这个“无限循环”。另外，设置config里的阵型是不起作用的。以及，Github的Formation那章里说可以继承Procbot进行阵型的自动设置（但就没法用MCTS了好像）
  11. 如果要考虑随机性的问题，且还是跟我一样用闭环的方法解决，似乎要考虑chance node的问题（我之前看了几个带伪代码的paper，但一下子看不懂就没实现了。若有需要可以跟我说）。或者用开环的方法解决（我Google搜不太到具体的做法）。但上述俩方法，网上（谷歌/百度/GitHub）都差不多太多有用的示例。。。