**中山大学计算机学院**

**人工智能**

**本科生实验报告**

课程名称：Artificial Intelligence

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | **22336326** | 姓名 | **朱禹溪** |

# 实验题目



# 实验内容

1. 算法原理

本次实验中主要使用了Alpha-Beta剪枝算法来于优化博弈树搜索。

递归搜索：从当前棋局状态开始，算法会递归地探索可能的走法，考虑双方玩家的行动。在每一层，算法交替考虑最大化和最小化玩家的行动，并评估当前局面的价值。

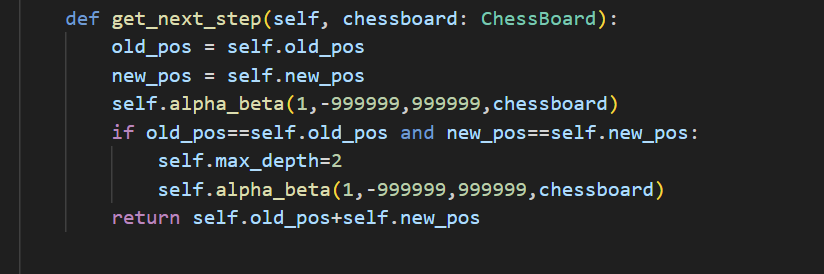
Alpha值和Beta值：Alpha和Beta值是算法用于剪枝的关键。Alpha值代表在Max层面上带来的最大值，Beta值代表在Min层面上带来的最小值。通过比较当前的节点值和Alpha/Beta值，算法可以准确评估哪些分支不值得深入搜索。

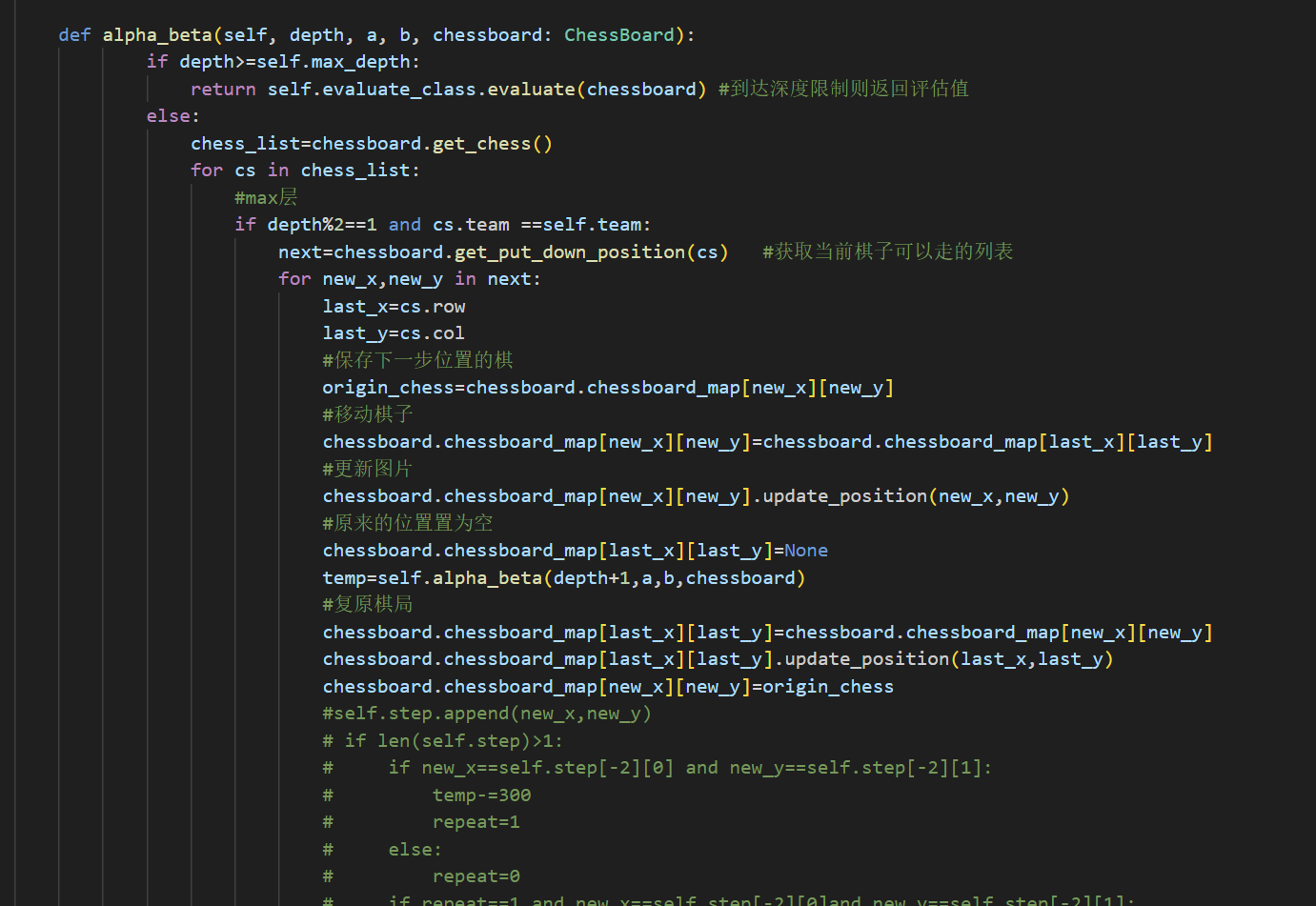
剪枝：在搜索树的过程中，如果算法发现某些分支不够优秀，会剪掉这些分支，这样就不必继续搜索这些分支，从而提高搜索效率。

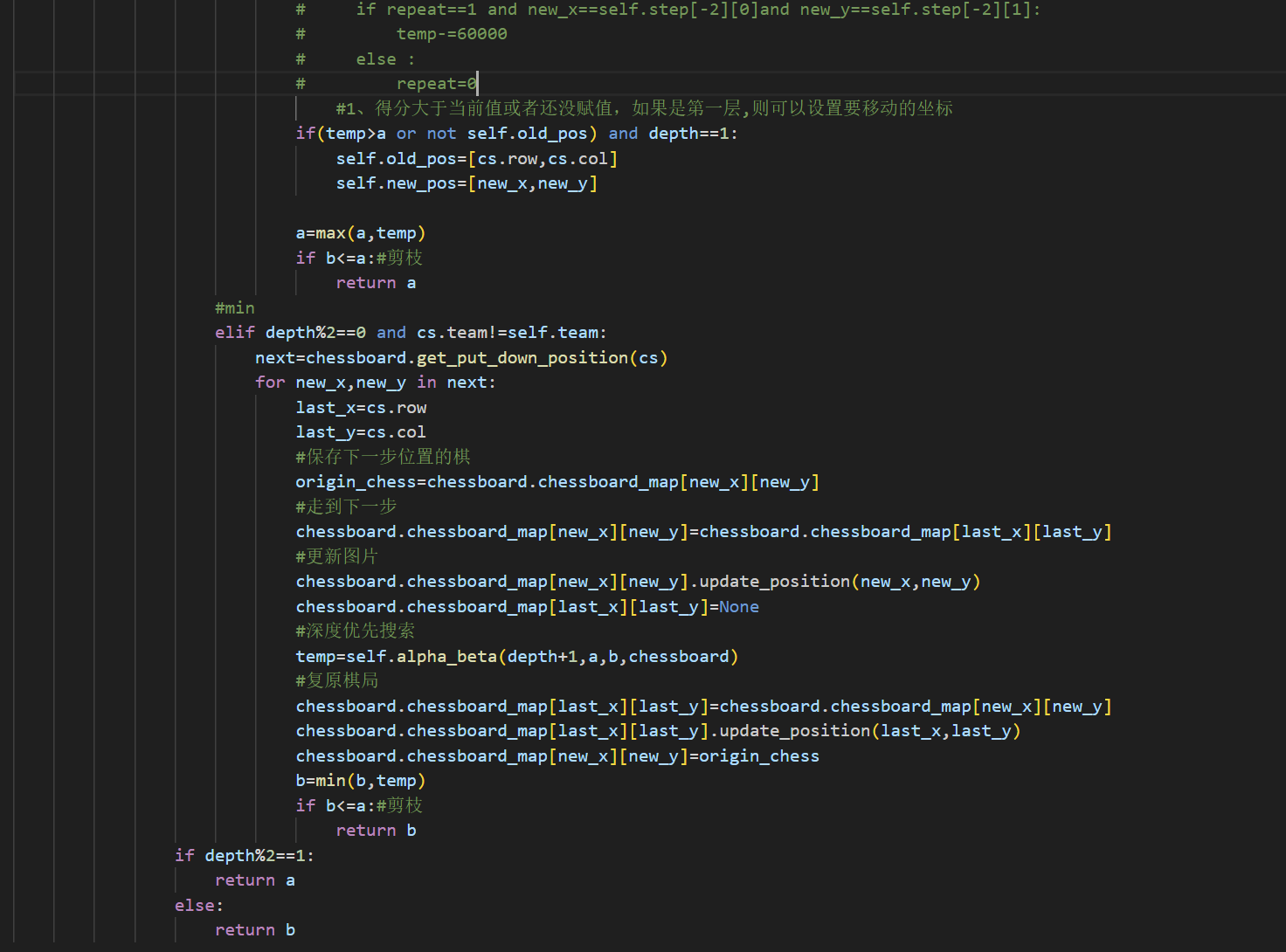
搜索顺序：在每一层，算法会根据当前是最大化还是最小化玩家，有选择地深入搜索。最大化玩家希望获得更高的分数，因此会尝试获得更大的Alpha值；最小化玩家希望获得更低的分数，因此会尝试获得更小的Beta值。

深度限制：为了避免无限的搜索，通常会设置一个搜索深度限制。当达到深度限制后，算法将评估当前局面的得分，然后返回。

1. 关键代码展示（可选）







# 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

人机对战（人执红）：



人机对战（人执黑）



Ai互搏（人执红）：



Ai互搏（人执黑）：



1. 评测指标展示及分析（机器学习实验必须有此项，其它可分析运行时间等）

对于Alpha-Beta剪枝算法，其时间复杂度取决于搜索树的大小和有效的剪枝操作。

搜索树分支因子：在这段代码中，每一步棋的走法都会产生一个分支，因此搜索树的分支因子取决于每一步棋的可选移动数量。

最大搜索深度：在算法中，通过 self.max\_depth 来控制搜索的最大深度。当达到最大深度时，会返回当前局面的评估值。

剪枝：Alpha-Beta剪枝算法的关键在于剪掉那些不会对最终结果产生影响的分支，因此有效的剪枝操作可以显著减少搜索的节点数量。

接下来我们大致计算一下这段代码的复杂度：

对于每一个棋局状态，它的搜索树的分支因子取决于每个棋子可以走的位置数，即平均分支因子为 (b)。最大搜索深度为 self.max\_depth。

在最坏情况下，需要遍历完整的搜索树，因此复杂度为 (O(b^d))，其中 (d) 为最大搜索深度。

整体而言前期每走一步需要时间大概在2.5s左右，后期棋子数量减少搜索 难度降低速度会大幅提高，整体一盘棋局会在3分钟以内。程序还存在一个bug，就是当遇到重复局面的时候无法及时脱离（算法中注释掉的就是原本想用于脱离的），导致ai对战最后会长将或者重复走闲棋，并且可能是由于没有将可走的棋子的列表进行随机的排列，互博时走的较为呆板，每次都是一样的棋局，后续还需要进一步的修改。

# 参考资料

[怎样做一道阿尔法贝塔剪枝的题(图解)-CSDN博客](https://blog.csdn.net/Sacredness/article/details/93124338)

4-搜索算法.pptx（课上ppt）