CS24011实验1：逆转

伊恩普拉特哈特曼
弗朗西斯科洛博

学术课程： 2022-23

Git标签： 24011-lab1-S-Games

介绍

在这个练习中，您将编写一个Java程序来玩反转录游戏，有时它的商标名称是“奥赛罗”。Reversi是在一个8乘8的棋盘上播放的，最初的设置如图1所示，玩家通过在棋盘上放置他们自己的棋子（黑色或白色）来交替移动，一次一个。黑色总是第一位的。

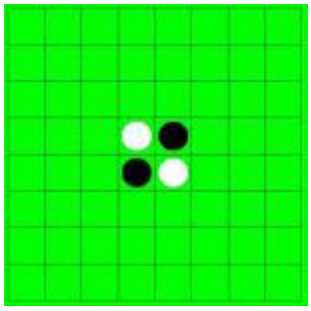

图1：反向游戏中的开局位置；黑色移动。

在下面的描述中，线段是形成一条连续的直线（水平、垂直或对角线）。玩家放置棋子的规则是：

棋子必须放置在一个空的正块上，这样就有一段线穿过演奏的棋子，然后通过一个或多个相反颜色的棋子，并以玩家自己的颜色结束。

当这样的一个线段存在时，我们说对手在该线段上的棋子被用括号括起来。当一段乐曲被演奏时，方括号内的乐曲会根据以下规则改变颜色：

每一段通过一块，然后通过一个或多个相反的颜色，并结束在一块球员的自己的颜色，相反的颜色通过线段通过都改变了块的球员自己的颜色。

在图2中，左边的图片显示了白色可能的移动，它支架起了他的对手的三个棋子，导致了在右边的图片中显示的位置。

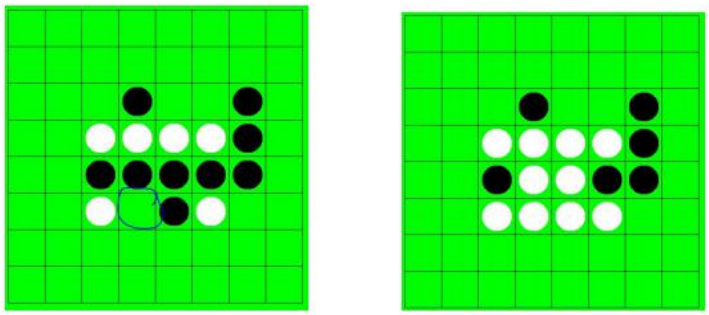

图2：白色在一个反向游戏中的移动，及其结果。

如果，也只有当，一个玩家不能移动，但他的对手可以，他才会错过一个回合。当双方玩家都不能移动时，游戏就结束了。（这通常，但不总是，因为所有的方块都被占据了。）获胜者是在棋盘上有更多他自己颜色的棋子的球员；如果没有这样的玩家，结果就是平局。

分配

本实验室的材料可在您的Gitlab repo COMP24011的实验室1分支中获得_

2022/COMP24011_用户名

它应该存在于你的Gitlab帐户中。按照CS手册的课程作业部分的说明来克隆回购。顶部的。md文件解释了如何组织回购，如何访问实验室1分支，以及如何使用刷新脚本来获取实验室材料。

要完成这个实验室，请解压缩文件的标签代码。zip.解压缩的目录包含一个用来玩反向游戏的Java程序。要编译代码，请使用javac奥赛罗软件。java。随后的调用java Othello将调用一个简单的程序，该程序允许人类玩家（总是黑色）与计算机（总是白色）对抗。提供的图形界面是相当基本的，委婉地说：点击不代表合法动作的方块只会产生恼人的哔哔声；如果是你去，但你没有合法的移动，那么你必须点击棋盘上的任何地方，允许游戏传递到电脑；如果你想玩另一个游戏，你需要关闭窗口，重新运行程序。此外，最好在电脑“思考”时不要点击。最后，作为一种特别的刺激，如果游戏以电脑的移动而结束，那么你必须点击棋盘才能看到结果。试一试，然后玩几个游戏。你会看到电脑玩得很厉害。实际上，它计算当前位置上可能的移动，但只选择第一个！你的任务是修改程序，使它使用极大极小搜索和阿尔法-beta修剪来玩一个更好的游戏。

主要的游戏逻辑是在类的棋盘状态中。这个类的一个实例表示游戏中的一种情况。字段颜色编码（1表示白色；-1表示黑色），而低级方法int内容（int x、int y）和空白内容(int x、int y、int块）允许在双板方块上检索和设置。这里，值1、-1或0分别表示在秩x和文件y的正方形上有一块白色、一块黑色或根本没有。为了让事情更容易，我们包括了布尔检查方法（int x，int y），它检查当前玩家是否有可能在正方形（x，y）上移动，以及无效的合法移动（int x，int y），它实际上执行移动。事实上，为了让事情变得真正简单，我们提供了一种方法来返回所有和只有那些合法行动的列表

对于当前的玩家。在这里，我们依赖于一个类Move来编码实际的移动；它只有两个公共字段，x和y；所以这个类的实例以显而易见的方式表示移动（合法的或非法的）。检索当前玩家的合法移动列表的方法是<<移动>获取合法移动（）。（确保您理解Java中的泛型类型！）。请注意，此列表可能是空的，因为当前的玩家可能无法进行移动。

计算机播放器位于移动选择器的类中。java，其中的主程序创建了一个实例。这个类所做的唯一的事情是实现静态方法移动选择移动（板状态板状态）。这是当轮到计算机移动时称为的方法。在它的当前版本中，这个方法只是获取合法的移动，如果该列表为空，则返回null（记住我说的有时没有合法的移动），否则返回该列表中的第一个移动。控制的其余部分则由程序来处理。你所要做的就是写一个更好的移动选择函数，而不是仅仅选择第一个。实际上，您必须在α β-修剪中使用极大极小值。搜索的深度应该简单地由静态的最终int搜索深度来控制，它目前在奥赛罗类的开始时被设置为6。您可能希望将其设置为一个较低的值，以加速开发。当你让一切都能正常工作时，尝试设置搜索深度8。这应该足以打击大多数人类玩家，而且不会太慢。

您将需要一个静态的计算函数，并且有几个选项可供选择。对于这个任务，你需要做如下的要求。游戏区域中的每个方块都被分配了一个数字：

```
120 -20 20  5  5 20 -20 120
-20 -40 -5 -5 -5 -5 -40 -20
 20  -5 15  3  3 15  -5  20
  5  -5  3  3  3  3  -5   5
  5  -5  3  3  3  3  -5   5
 20  -5 15  3  3 15  -5  20
-20 -40 -5 -5 -5 -5 -40 -20
120 -20 20  5  5 20 -20 120
```

这些数字反映了一个玩家在各自的方块上的价值。请注意，边缘上的正方形具有很高的值（因为这里的块很难取），而角落上的正方形具有更高的值（因为这里不能取块）。相比之下，相邻的方块有负值，因为这里的一块将允许对手移动到一个高值的方块上。然后，板位置的值可以通过将白块占据的所有方块的权重加起来，然后减去黑块占据的所有方块的权重来定义。因此，该值总是从怀特的角度来计算的。（顺便说一句，上述数值组取自PeterNorvig的一本旧教科书）。

你还必须编写一个程序来计算董事会位置的极大极小值。如果你发现α β修剪困难，你可能更喜欢首先尝试普通的极大极小值，然后毕业到α β修剪一旦工作。当你搜索一棵树的棋盘位置时，你需要做什么。请记住，板体状态的实例是Java对象。当你打电话给船州时。makeLegalMove（x，y），这将更新板（和当前的播放器）。因此，如果您想在搜索树中获得一个顶点板状态的子节点，您将需要为每个合法的移动创建一个新的板状态副本，然后在该副本上执行有问题的移动。方法董事会状态。我们在板状态中提供的deepCopy（）是一种克隆方法，因此调用板状态1=板状态。deepCopy（）将董事会状态1设置为董事会状态的新副本；随后修改董事会状态1(e。g.通过执行一个合法的行动)，然后不改变董事会的状态。当计算游戏树中一个顶点的子值时，要注意两个拐角的情况：一种是计算机没有合法移动的情况。在这种情况下，控制

传球给另一个球员。就搜索树而言，这意味着表示有问题的棋盘状态的顶点只有一个子节点，所有的片段都是相同的，但是当前的玩家改变了。

顶级的调用与后续的调用有点不同，因为你必须选择产生最佳子级的移动（从计算机播放器的角度来看），而不是简单地评估子级。重要的是，只有当且仅当没有可用的移动时，才返回移动为空。这很好：我们的代码将理解这意味着什么。

试试奥赛罗节目。java与您的MoveChooser版本一起。爪哇节，和恒定的奥赛罗。搜索深度设置为值6。由此产生的程序应该与一个人类玩一个不错的奥赛罗游戏，并且肯定应该打败一个或多或少选择随机移动的玩家。此外，在普通电脑上以搜索深度运行时，程序的任何移动都不应该超过几秒钟；如果是这样的话，你可能编写了低效的代码。

提交

自述文件。md说明解释如何上传你的文件到你的Gitlab回购的lab1分支。其中一个文件必须是您的MoveChooser .java的更新版本；您可能包括其他。java文件（例如，包含子类）。但是，您可能不包含任何具有其他扩展名的文件。您也不能修改练习中提供的文件（当然，MoveChooser.java除外）。任何此类文件或对现有文件的更改都将被忽略。重要提示：不要压缩解决方案的源文件。记住：您需要用命令标记您的提交

git标签24011-lab1-S-Games

你很可能也会在其他课程中这样做。关于使用Gitlab提交课程作业的一般说明可以在CS手册中找到。

提交的截止日期是10月17日星期一的18：00。实验室将自动离线。标记程序将从Gitlab下载您的代码，并根据随机板配置上的参考实现进行测试。为了避免这类随机板的影响，自动标记将进行3次，你的最终成绩将是中位数。（这需要一段时间，所以预计标记至少需要两周的时间。）

参考实现的代理有三种设置：简单、中和硬。每个测试都在一个随机板上开始，并以固定数量的移动对参考代理播放实现。在每次测试运行之前和之后，标记程序调用“板状态”。结果（），并记录这个板的得分。评分程序将使用这些棋盘分数来衡量你的实现在多大程度上提高了它所控制的玩家的位置。它还将跟踪您的实现所做的移动，以检查它是否正确地遵循具有规定的计算函数的极大极小算法。最后，标记程序测量实现选择其移动的速度。

标记方案

总共有20个分数。标记方案如下：

1标记您的代码完成所有测试运行，没有运行时错误。(18标记）对于
三个参考代理设置中的每种设置：

- 3个分数取决于测试的比例，其中你的实现提高了它所控制的代理的董事会分数。（3分需要75%，2分需要50%，1分需要25%。）
- 3标记取决于您的实现正确遵循极大极小算法和规定的计算函数的移动比例。（3分需要100%，2分需要75%，1分需要50%。）

1标记：在50%的测试中，实现的速度与引用代理一样快。