

数据结构与算法报告

|  |  |
| --- | --- |
| 教 师： | 刘云淮 |
| 助 教： | 吴 晗 |
| 组 长： | 汤远哲 |
| 组 员： | 辛春晓 戴文晗 方佳娱 张日栋 |

二〇二三年六月

# 思想阐述

星际消战围绕传统“开心消消乐”游戏展开，在继承其“相邻棋子可交换”、“三个及以上同色棋子在水平或垂直方向相连可触发相消”、“消除后空位由上方棋子掉落填充”三个基本规则的基础上，对规则进行了更改：

## 游戏为竞赛模式，双方玩家采用回合制轮流在主棋盘进行操作；

## 将三个及以上同色棋子在水平或垂直方向相连可相消，变更为在三个及以上同色棋子在水平或垂直方向相连可触发相消的基础上，与之形成同色连通块的棋子也可消除；

## 双方已知掉落棋子的颜色、数量即位置；

## 若当前主棋盘上无一次操作后可导致消除时，比赛结束。

在熟知规则的基础上，我们小组采用了贪心算法的逻辑对代码进行设计。主要思想为：

1、扫描所有交换操作；

2、计算所有交换操作的得分情况；

3、执行最优得分的操作。

在形成此算法逻辑后，我们小组对算法细节进行了优化。从仅考虑本次操作的最高得分变更为考虑回合操作的最优得分。考虑到贪心算法可能会导致操作超时，于是对算法复杂度进行了压缩。

# 程序代码说明

在多次对算法进行改进优化后，形成的最终程序代码逻辑如下：

## 操作选择：

已知可操作的主棋盘规模为6\*6，所以在同一行可进行的水平方向移动操作为5种，在主棋盘上可进行的水平方向移动操作为5\*6种，同理在垂直方向的移动操作共有5\*6种，所以在一次操作中共有60种操作选择。

我们将“在本次操作中，执行最高得分操作”的算法称为一度贪心算法；将考虑回合操作的最优得分的算法称为二度贪心算法，即计算本次己方操作得分，以及对方下一步操作的最高得分，将两次得分相减计算分差，执行对我方最优的操作。

根据二度贪心的算法逻辑，程序先遍历己方60种可行操作，执行相消或不相消的操作结果，计算己方得分并形成60个小棋盘。在60个小棋盘上再遍历对方60种可行操作，计算对方最高得分。将己方得分减去对方最高得分得到分差。理想情况下，存在最大正分差，算法将执行此最大正分差的对应操作。若分差均为负，则会执行防守操作，即破坏当前可操作格局。此算法的前提假设为，对方会执行一度贪心算法，即执行当前最高得分操作。

在二度贪心算法中，选择遍历对方60中可行的操作而非仅查看对方的可行操作，目的是在考虑算法的后续升级需要，即将二度贪心算法拓展到三度贪心算法或多度贪心。

## 得分计算

在模拟执行操作后，算法会计算当前棋盘得分。程序采用了广度优先算法计算连通块个数。

程序依据从上至下，从左到右的顺序扫描棋盘，查看棋盘在水平方向及垂直方向上是否存在三个连通块。若有，则检查该连通块是否已存在标记，若无标记，则对连通块进行标记。标记的主要目的是为了避免重复计算，减少下一步计算得分操作的运算时间。

若棋盘存在标记，则程序会扫描所有连通块，计算其棋子个数及得分。在计算连通块棋子个数时，程序使用了以队列为基础的广度优先算法。假设当前连通块棋子个数为5，以标记中的其中一个棋子A为起点，将棋子A加入队列，先遍历A的相邻棋子，将A的相邻同色棋子加入队列，再依次遍历队列中棋子的相邻同色棋子，将新的相邻同色棋子加入队列，以此类推直到队列中无棋子。计算队列中出现过的棋子个数，即为连通块的棋子个数。依据公式 “得分=(棋子个数-2)2”计算分数。

联通块消除后，棋子会掉落填充空位。程序会接着扫描棋盘，查看是否可进行消除，如此循环直至棋盘中无可消除棋子。

## 棋子掉落

棋盘共有1200行，在进行消除操作后，空位上方棋子要对空位进行填充。若在每次棋子消除后，以复制列表的方式来移动棋子填充空位，其所需要的运算时间是极大的。为减少运算时间，程序不选择对储备棋子进行复制移动操作，而是采取截取标记的形式实现棋子掉落操作。

若棋盘中棋子消除，则先移动主棋盘中消除块上方棋子下移，同时计算每列所消除棋子个数。从储备棋盘中存有标记的位置开始，由下至上取要对应数量棋子，并在结束取要棋子的位置做标记。在下次取要棋子时，从最后一次标记开始取要。如此可以极大地减少运算时间，为二度贪心算法留足时间。

# 测试过程

测试主要是依靠对战平台的对战结果。大概测试过程如下。

起初的代码便是简单的贪心算法，通过在对战平台测试，发现胜率不高，可行性不强。

然后想到对手也许会采用贪心算法，便在假设对手采用贪心算法的情况下，开发了二重贪心算法，即交换的两个方块不再是仅仅取决于此次操作的得分高低，而是也考虑了下一步对手得分的高低，通过作差，取可以产生最优分差的交换。通过在对战平台的测试发现，虽然胜率有所提高，但二重贪心算法经常会导致超时，由此产生了本代码最大的改进，便是通过标记列的下落高度，来避免对整个列进行重新复制，这种方法大大降低了时间复杂度。通过在测试平台的对战发现，这种改进，耗时极小，便萌生了采用广度更大的二重算法。

对于六乘六的棋盘，可产生的交换操作共有60种，考虑对手的下一步操作，便有60\*60种情况，但由于标记列的下落位置的做法，大大降低了时间复杂度，是允许我们进行这种广度的计算的，通过在对战平台的测试我们发现，这种算法，并未超时。

# 总结

## 代码优点（获胜点）

通过标记棋盘的下落高度，而不是对整个列进行复制，大大降低了时间复杂度，允许我们进行60\*60的遍历操作。

在得分落后时，有时会出现和对手反复重复彼此操作的情况，我们在观察对局回放时发现了这种情况，并选择采用一重普通贪心算法，来主动改变棋盘布局，防止困毙。

如果整个棋盘没有交换操作可以导致消除，便会直接结束比赛，我们主动利用了这一规则。在得分优势的情况下，如果有的交换操作可以导致对手下一步无法进行消除，我们会直接采取这一交换操作，终结比赛，而在得分劣势时，我们则不会采取这一交换操作。

## 代码缺点（失败的部分原因，或可改进之处）

也许对手也会主动利用“如果整个棋盘没有交换操作可以导致消除，便会直接结束比赛”这一规则，但我们没有阻止对手利用这一规则，实际上这种预知需要“三重”的算法。

得分落后时，对于前面提到的困毙的处理，我们过于草率，其实可以尝试使分差“比较优”的同样可以打破棋盘布局的二重算法，来提高获胜的概率。

实际上我们的代码运行时间还是比较短的，也许在得分落后较多时我们可以部分采取三重算法，在保证不超时的情况下来提高获胜的概率，但我们没有这么做。

# 小组分工

汤远哲 编写代码 实验报告配图

辛春晓 编写代码 实验报告配图

戴文晗 编写代码 技术顾问

方佳娱 实验报告文字

张日栋 实验报告文字