AI 2048

付登攀， 张均备， 聂丰

**问题分析**

2048是一款数字类游戏，由每次电脑给出的2, 4的位置进行上下左右的操作，相邻相同数字的方块可以合并，从而达到2048以及更高的方块。

那么关键在于如何拼凑出2048，较为直观的解法是压缩所有每行或每列的空格数，进行列或行的合并，选出一个最可能获得最大值或者说使得空格数最多的方法。

但是游戏的关键在于每次走完，新的2， 4的位置是随机产生的，这就造成了该问题实际上是一种博弈，需要对每一次电脑给出的值进行一个评估。因此很容易想到搜索的方法，利用人给出的是最优解法而电脑给出的可能是最差的局面，建立min-max博弈树，然后进行alpha-beta剪枝，找到当前最优解。

而如何评测最优，这个是需要衡量的。较为直观的是空格数要少，然后最好有一行或一列一直保持着最左或最右有最大值并且在这一行或列的其他空格依次递减，而且希望相邻的数据之间差值能尽可能的小等等。

有了如上的思路，加上自己玩游戏的时候的一些策略，我们总结出了问题的求解方式。

**问题求解**

**最优规则分析， 具体如下**：

1. 最大的数尽量要放在一个角落，并且在角落中后就不要再去改变其位置。
2. 保证空格数尽量的多。
3. 选择最大数所在的行（列）作为主行（列），即在这一行（列）上的数从最大数开始尽可能的保持递减，这个行（列）选定后就尽量不变，除非在极短的步骤内列（行）上的单调性更好，或行（列）上的数恰好合并完了，而列（行）上又有数 时可以考虑主行（列）转成主列（行）。
4. 数据尽量保持一种相近性，相邻的数之间的绝对值之差要小。
5. 在保持主行（列）的条件下，主行（列）相邻行的单调性如果能保持衔接对局势会很有利。比如如下的排列方式（主列）：

|  |  |  |  |
| --- | --- | --- | --- |
| 256 | 128 | 2 |  |
| 512 | 64 | 2 |  |
| 1024 | 32 | 4 |  |
| 2048 | 16 | 8 |  |

1. 发现如果排列是按zig\_zag的顺序递减，其也很容易收拢合并。比如如下方式：

|  |  |  |  |
| --- | --- | --- | --- |
| 32 |  |  |  |
| 64 | 16 |  |  |
| 1024 | 128 | 8 | 4 |
| 2048 | 512 | 256 | 4 |

其按上、左、左、左、下、左、左、上、左、下、下 的方式可以不管出现的数字是什么而最终完成2048的收拢。所以该种形势也是有利的。

**搜索方法（控制树深的MIN-MAX算法）：**

1. 给定一个状态作为博弈树的根节点，分别有上下左右四个分支，其中先减掉不会使状态改变的分支（死循环）。
2. 之后，模拟由机器走的一步，即加2或4，有多种情况，只选取评分最低的四种情况进行深入。
3. 对上述每种情况再模拟人走，即选取四个分支中评分最大是一个作为该节点的值。
4. 之后，回溯到机器节点，机器选择四个分支中值最小的。

之后，回溯到当前状态节点，选择四个分支中最大的方向作为当先的选择，并返回。

**评估函数设计：**

由规则分析中，我们总结了4个评估方法来取加权计算最终的评估值，分别是：空格数，单调性， 平滑性，和最大位置

1. 空格数 = 16-当前状态的空格数。因为空格数越多，可移动的时间越久的可能性越大。
2. 单调性=2\*zigzag+8\*蛇形单调。将当前状态拉成1\*16是一维数组，在按照这两种形态排列，前面的值减后面的值。（之前先取log2）这样做的原因是这两种形态比较能累积起来。
3. 平滑性即每两个相邻的值之间相减的绝对值的相反数。
4. 最大值的位置若在角落，则有评分，否则没有。这样做的理由是最大值在角落，便于给其他需要合并的值腾出空间。

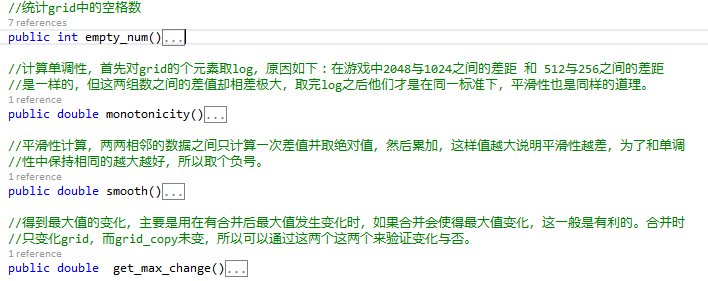
**Trade-off思考：**

由于实际上2048游戏中，机器走的时候是随机在剩余生成2或4的，所以上面1中假设机器每次都将局面带入最劣是有一些问题的，个人觉得在机器走的时候对机器所有的可能走法算期望值或许更好。相对于上一种方法在这里只需将Min步求取评估最小的走法改为计算所有可能走法的均值。

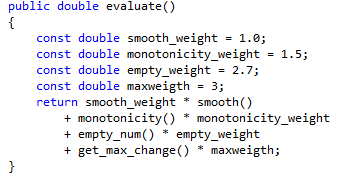
算法设计与实现：

首先封装了MOVE类, 定义上下左右的走的方法，压缩空格，并且进行相邻的方块数字合并。

定义计算空格树，判断单调性，平滑性，以及最大值位置的评估函数



对于以上四个函数，进行加权，因此评估函数如下

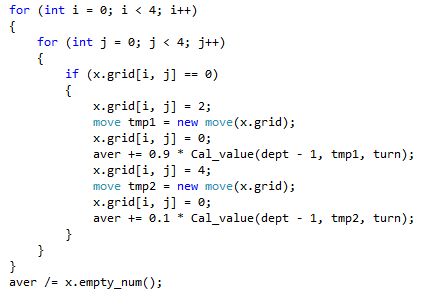


方法一：MIN-MAX博弈树， 为求平均期望值

1. NextStep:输入一个状态及方向，输出朝这个方向走一步之后的下一个状态
2. SimpleTree:输入一个状态，输出朝上下左右走的四个分支
3. Evaluation：输入一个状态，输出一个评分
4. SimpleAdd:输入一个状态，输出所有添加了2或4的情况中评分最低的四种情况
5. MinMax：最小最大算法，输入一个状态，输出一个方向

方法二：对当前的所有可能的局面求平均值，然后选最优解法，递归函数控制树深为4，函数核心算法如下：





**实现结果与比较**

方法一：通过用博弈树来计算当前的最优解，如果为平局则随机取解，我们得到达到2048的概率大概为百分之四五十，达到2048之后平均分在27000左右，没达到的估计在10000左右，所以总平均分估计在17000左右。

方法二：通过进行每步算计算机的所有解法的平均值，可以得到较好的结果。基本上能以80%~90%的概率实现2048，偶尔能实现4096。得分平均在30000分左右。

**总结**

通过此次的实验，有助于我们自己去模拟人机操作，通过制定一些合理的规则，来取得较好的解法。通过实验了不同的规则，以及树深的控制，我们发现取平均的算法要更好一些，而且调整评估函数的权值来得到一个较为合理的评估函数，对于判断整个游戏格局至关重要。这次的实验很好的锻炼了我们的模拟规则的编程技巧，当然我们的程序还有待提高，比如说如何做到更快的搜索，由于限制树的深度牺牲了整个游戏的最优策略，因此这些都是值得我们继续改进的地方。