

搜索大作业报告——2048

姓名： 张嘉玮

学号： 2016011528

班级： 自 61 班

日期： 2018.04.07

一、算法概述

该算法提供一个 2048 的“AI”玩法，采用 minmax 算法，然后借助 alpha-beta 修剪的思路，在不削弱算法功能的基础上，提高搜索效率。

二、设计过程

1. 学习 python

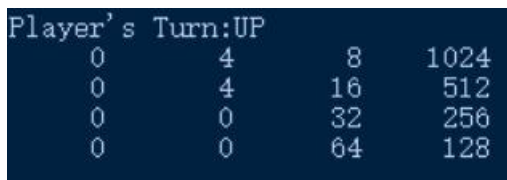
由于之前没有接触过 python，所以在遇到这个作业时，首先第一个过程学习语言。由于之前的 C 学得还算可以，所以虽然是一门新的语言，但就好像“在哪见过”，再加上在作业之前就受到高人指点，所以在学习时，还算顺利。但是目前所学的只是 python 里面很少的一部分，强大的 python 还有许多方面的应用，还需在接下来继续学习。

2. 玩游戏

讲真，之前并没有玩过 2048 这个游戏，所以对其里面的一些基本策略都不是很清楚。所以就玩了两天 2048，并通过网上的一些策略，对 2048 游戏有了初步的了解。

3. 了解游戏策略

本游戏主要的策略是尽量使得空格子多，尽量使得已经累计的数字有一定的规律，而这个规律就是要这些数次的排列具有单调性。单调性的意思是让 4*4 的网格在某个方向上，按从小到大的顺序递减。比如下面的这种状态就比较好：



Player's Turn: UP			
0	4	8	1024
0	4	16	512
0	0	32	256
0	0	64	128

（运行结果截图）

当然，相比于空的各自格子数，“单调性”更加重要，所以应优先考虑“单调性”。

4. 算法设计

由于框架已经给出，所以在设计时，集中精力考虑 PlayerAI_3。刚开始走了不少弯路，

主要还是评价函数不知怎么设定。在评价函数设定以后，权值的设定很难，在进行了一番实验之后，才得到了一个比较好的结果。但是好像还不是很好，但人工“试探”真是很费劲、而且还需要大量的时间。借助博弈树的 AI 算法，采用 minmax 算法，然后进行 alpha-beta 修剪。（具体步骤见三）

5. 算法优化

由于本游戏的特殊性，探索路径的八条路径可以优化，同时 alpha-beta 算法其实可以直接融合在算法当中，而不用单独给出，这样会增加搜索效率，提高运行速度。（具体步骤见四）

三、搜索部分算法

1. 评价函数：

本算法主要考虑一个 4*4 矩阵的“点调性”，所以应该有下面八条可能递减的路径（数字代表递减顺序）

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

路径一：1、2、3、4、8、7、6、5、9、10、11、12、16、15、14、13

路径二：1、5、9、13、14、10、6、2、3、7、11、15、16、12、8、4

路径三：13、14、15、16、12、11、10、9、5、6、7、8、4、3、2、1

路径四：13、9、5、1、2、6、10、14、15、11、7、3、4、8、12、16

路径五：16、15、14、13、9、10、11、12、8、7、6、5、1、2、3、4

路径六：16、12、8、4、3、7、11、15、14、10、6、2、1、5、9、13

路径七：4、3、2、1、5、6、7、8、12、11、10、9、13、14、15、16

所以需要搜索的路径也应该是这八条。

而要递减的原则便是不同的位置在某一条搜索路径下面，其权值不一样。比如路径三，13 位置的数的权值应该最大，下来是 14，下来是 15……一次类推，所以 1 位的权重应该最小；相反，若是搜索一路径，则 1 位置分权重应该最大，

13 位置的权值应该最小，一次类推。

具体程序权值：第 n 个数，其权值是 $0.25^{(n-1)}$

在八条路径中，找出结果最大的一个，作为此矩阵的 `score`。

2. Max 层：

该层的元素个数是一个 4×4 数方可以移动的方向，最多为 4 个（上下左右），最少为 0 个（游戏结束），该层的每一个结点的 `score` 是原矩阵在按照对应方向移动后的评价函数值加上来自 `min` 层的返回值。

3. Min 层：

对来自 `max` 的矩阵进行 `computer_play`，而这里应该是遍历所有为 0 的点，让这些点分别填上 2 或者 4（由于知识估算，程序当中只考虑了 2），在移动（等同与增加了一层搜索深度）后算其评价函数值，在所有这些评价值里面选取最小的，返回到 `max` 结点。

4. alpha-beta 修剪：

在进行下一个 `max` 结点的评价时，把已经搜索过的 `max` 结点中选取最大值（记做 a ）带入，在进行该结点的 `min` 结点搜索时，若某一个位置填上 2 以后其对应的 `max` 结点已经小于 a ，则停止该结点的搜索。返回到上一层。

四、算法优化：

1. 搜索路径优化

为了得到递减的最大值，需要搜索八条路径，但是分析矩阵不难发现，评价价值最大的路径的一定是从“最大的”拐角处出发的路径。八条路径分别从四个拐角触发，每一个拐角两条。所以，只需要选取最大的两条搜索即可。这样可以减少六条遍历路径。

2. alpha-bate 优化

本算法是在所有的为 0 的位置上填充 2，然后选最“糟糕的”一个，分析不同的位置，可以的到，最糟糕的位置其实就是在一条路径上最先出现的为 0 的点，

所以剧可以简化这个寻找“最糟糕”的位置的过程，这样不仅使得后面的 `max` 结点只需搜索一次，第一个 `max` 结点也只需有序哦一次，这样将大大提高搜索效率，得到最好的移动方向。

五、感想收获

这次作业是我第一次使用 `python`，回过头想，学习一门语言最好的方法应该就是搞一个大作业吧。由于这次作业，把代码框架已经给出，所以大大减少了工作量，让我们把主要精力投入到算法设计上。比起之前的 `c++` 大作业，要好得多。

但是设计的算法还是有点简单，按照这个算法，在最大的数字超过 2048 后，权值应该要修改才会走的更远，得到的数更大，但是和遗憾，没有实现。稍稍设想一下，这个权值如果通过机器学习，在不同的阶段进行优化，可能会得到更好地结果。同时也就不同我们自己花大量时间去得到权值了。