****

**《人工智能原理实验》**

**实验报告**

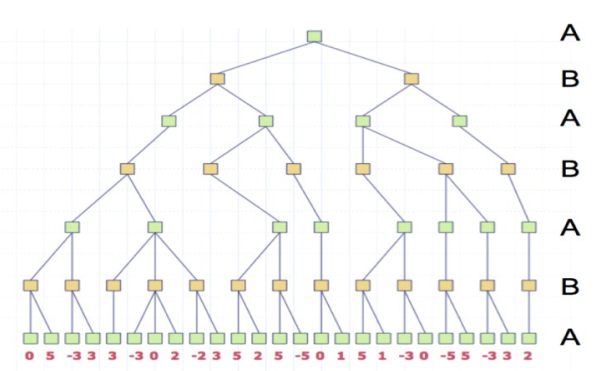
**（Assignment2）**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院名称** | **：** | 数据科学与计算机学院 | | | | | |
| **专业（班级）** | **：** | 17软件工程2班 | | | | | |
| **学生姓名** | **：** | 张淇 | | | | | |
| **学号** | **：** | 17343153 | | | | | |
| **时间** | **：** | 2019 | 年 | 9 | 月 | 28 | 日 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | **成绩** | **:** |  | |
| **Assignment2** | **：** | | **α-β pruning、Sudoku** | | | | |

* + - 1. **实验目的**
  1. 理解并掌握课堂内容：α-β pruning、CSP及MRV
  2. 实践出真知：使用编程运用上述方法解决Sudoku
     + 1. **实验内容**

1. Draw the diagram after alpha beta pruning



1. 实现Sudoku Solver，详情请点击[此处](http://www.cs.toronto.edu/~lyan/csc384/a2/A2Sudoku.pdf)。
   * + 1. **实验原理**
2. α-β剪枝

[**Minimax with Alpha Beta Pruning**](http://web.cs.ucla.edu/~rosen/161/notes/alphabeta.html?tdsourcetag=s_pctim_aiomsg)

Alpha-beta pruning gets its name from two bounds that are passed along during the calculation, which restrict the set of possible solutions based on the portion of the search tree that has already been seen. Specifically,

β：Beta is the minimum upper bound of possible solutions

α：Alpha is the maximum lower bound of possible solutions

Thus, when any new node is being considered as a possible path to the solution, it can only work if:

α≤ N ≤β

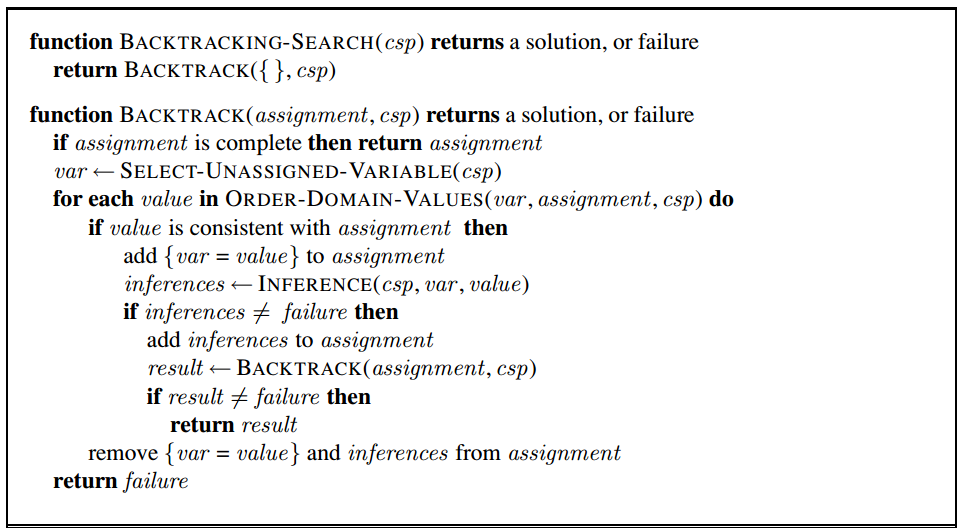
where N is the current estimate of the value of the node.

[**α-β搜索过程**](https://wenku.baidu.com/view/b1af9d2b3169a4517723a3f4.html)

在进行α-β剪枝时，应注意以下几个问题：

1. 比较都是在极小节点和极大节点间进行的，极大节点和极大节点的比较，或者极小节点和极小节点间的比较是无意义的。
2. 在比较时注意是与"先辈层"节点比较，不只是与父辈节点比较。当然，这里的"先辈层"节点，指的是那些已经有了值的节点。
3. 当只有一个节点的"固定"以后，其值才能够向其父节点传递。
4. α-β剪枝方法搜索得到的最佳走步与极小极大方法得到的结果是一致的，α-β剪枝并没有因为提高效率，而降低得到最佳走步的可能性。
5. 在实际搜索时，并不是先生成指定深度的搜索图，再在搜索图上进行剪枝。  如果这样，就失去了α-β剪枝方法的意义。在实际程序实现时，首先规定一个搜索深度，然后按照类似于深度优先搜索的方式，生成节点。在节点的生成过程中，如果在某一个节点处发生了剪枝，则该节点其余未生成的节点就不再生成了。
6. CSP

约束满足问题的简单的回溯算法：



1. MRV

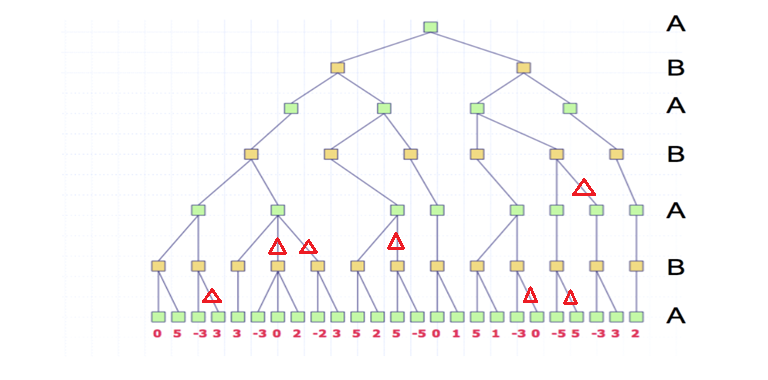
最少剩余价值（MRV）启发式：选择合法取值最少的变量。----强有力的指引，也就是选择最少剩余值的变量，通过早期有效剪枝，降低搜索树结点数。

* + - 1. **实验器材**

Windows10, VS Code, Python 3.7.2-64bit

* + - 1. **实验过程与结果**

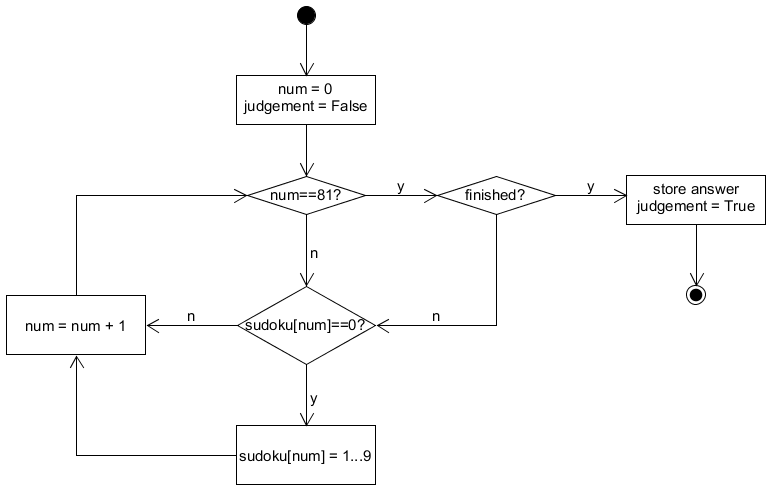
1. 根据实验原理中的α-β剪枝部分的内容，可求得答案为：



（注：红色三角形表示对应边及其子节点均删掉）

1. Sudoku Solver
2. Brute Force

既然是暴力搜索，那么对于数独中每一个空白格均采用0~9进行尝试，直到找出一组数字能够让数独完成。为了编程方便，采用了递归的方式实现，作业中的算法流程图如下：

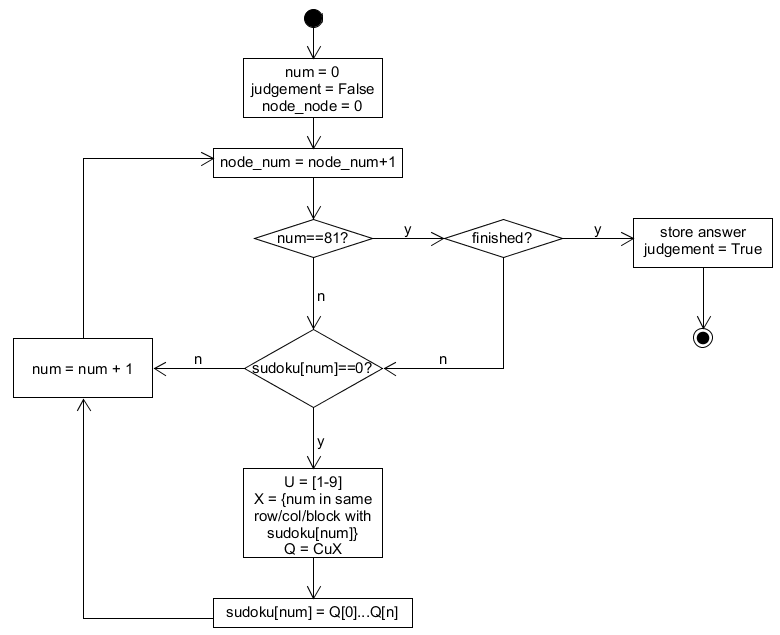


时间复杂度为：，其中n代表空格数量

因为这种方式过于直接，所以即使让程序跑一晚上也没有办法得出一个测试样例的答案。故本步骤没有相关结果的截图，但我们可以从下一个方法中得知此方法理论上是可以得出结果的，只是时间要求比较长一点。

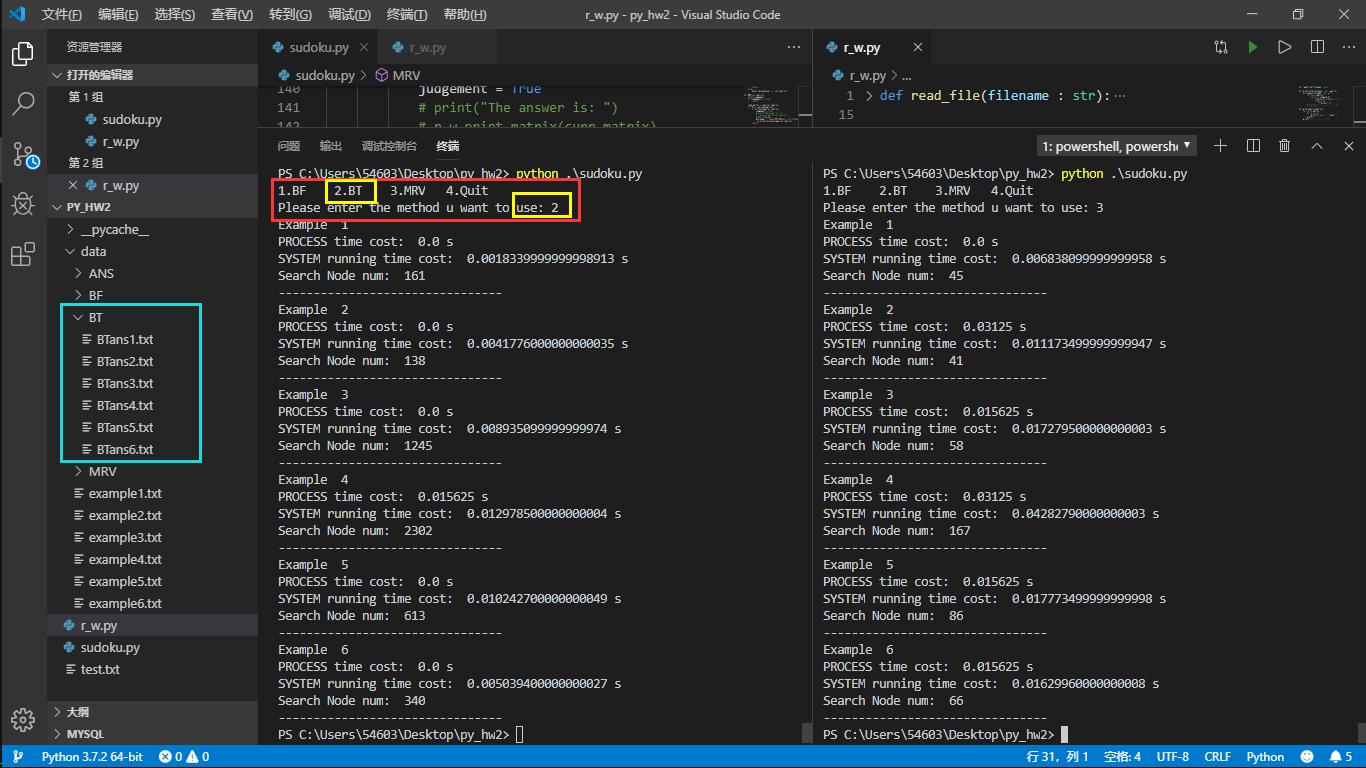
1. Back Tracking（CSP）

这个方法指明使用回溯法，但是需要使用条件满足约束的思想，可以参照上文中实验思想中的算法，对上述暴力搜索加一点条件，让其搜索空间缩小，从而能够在“可接受”的时间内求得答案。作业中的算法流程如下：



从某种意义而言，回溯法就是深度优先算法，而深度优先的复杂度为，其中n代表空格数量。

实验结果如下：



如上图，我们在程序运行（左半部分的terminal中）开始时选择的是方法2（BT：BackTracking），分别列出了求解各个test-case所消耗的CPU时间（time.process\_time()：精确到微秒,**返回当前进程执行 CPU 的时间总和，不包含睡眠时间。由于返回值的基准点是未定义的，所以，只有连续调用的结果之间的差才是有效的。**）、进程所占用操作系统的时间（time.perf\_counter()：精确到纳秒，**返回计时器的精准时间（系统的运行时间），包含整个系统的睡眠时间。由于返回值的基准点是未定义的，所以，只有连续调用的结果之间的差才是有效的。**）

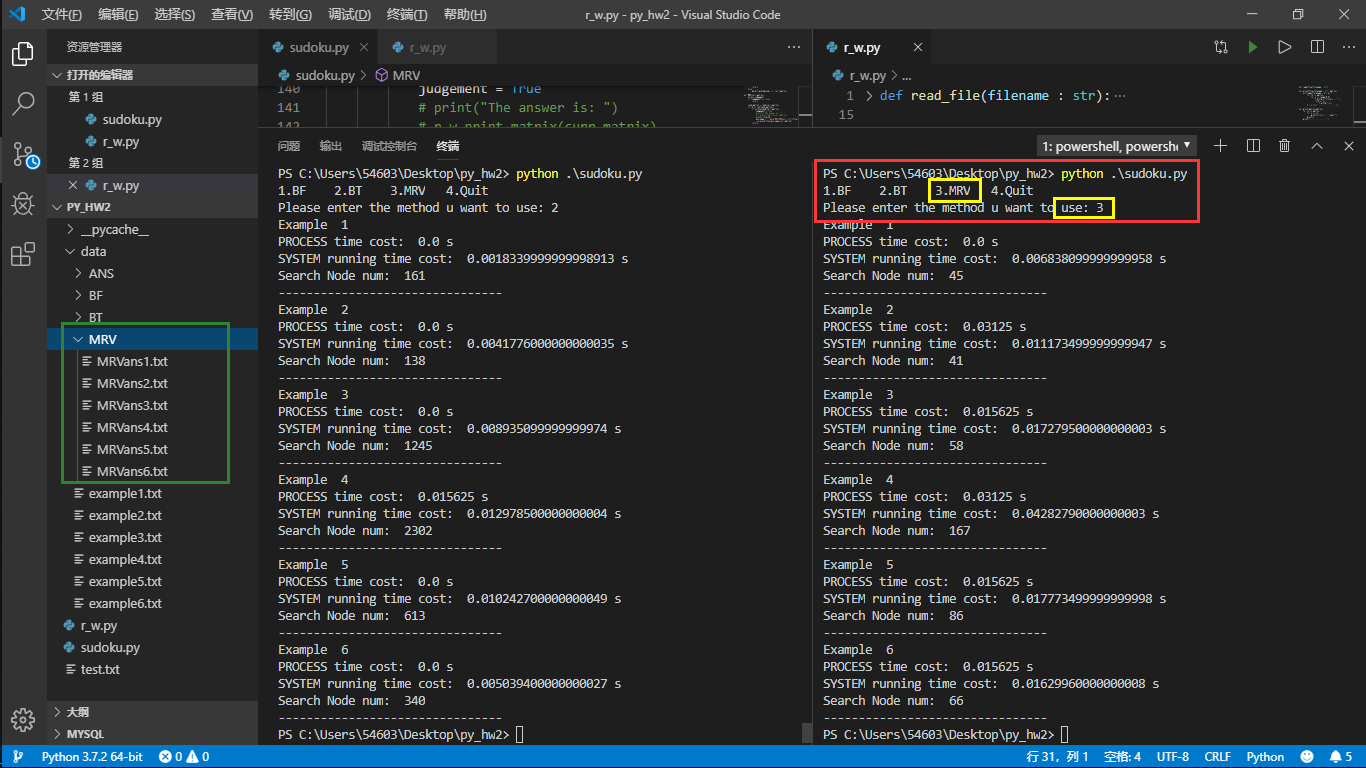
因为只需要求出一组解即可，所以我们的搜索的节点数（node\_num）与数独中的空白格子的数量没有明显的关系。对于每个数独的答案，程序运行完后会将其存储到路径为”./data/BT/”的文件夹中。

1. Minimal Remaining Value（MRV）

MRV是CSP的一种解决思想，因此可以本方法可以建立在方法二的基础上。根据题意，最小剩余价值需要通过forward tracking使当前步骤的选择空间缩小来实现。

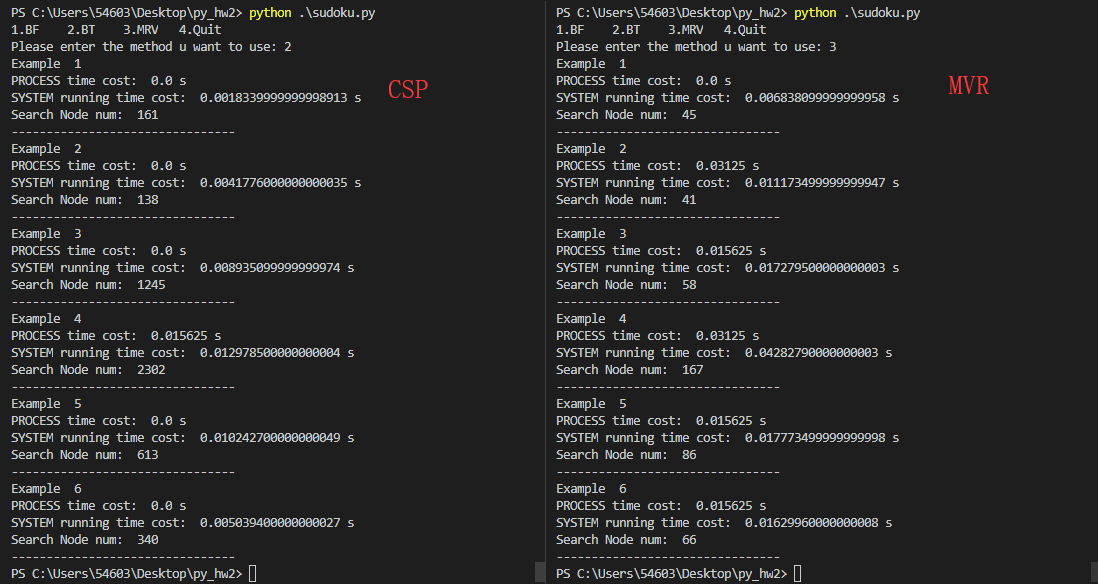
考虑到方法二中对于空白格子的填充是按照索引顺序进行的，具有较大的不确定性。因此我们可以先**将当前数独中的空白格子按照“可能填入的数字”的个数，从小到大进行排序，然后首先让选择空间最小（可能填入的数字最少）的格子填**，从而能够使整体问题的选择空间有效的缩小（如：假设有津剩下两个空白格子a,b的数独，经过行、列、九宫格筛查后，格子a只有两个备选数字，格子b有5个候选数字。如果先填a，那么整体问题的搜索空间就是原来的1/2；反之，整体的搜索空间就是原来的4/5）。与此同时，这种操作还能够让回溯的层数更少一些，在递归调用这一部分中消耗的时间减少。

实验结果为：



同样，我们在terminal的右半部分使用本方法对各个test-case进行求解，分别打印了求解过程的CPU使用时间、占用操作系统的时间以及搜索的节点数，求得的答案存放在路径”./data/MRV/”的文件夹中。

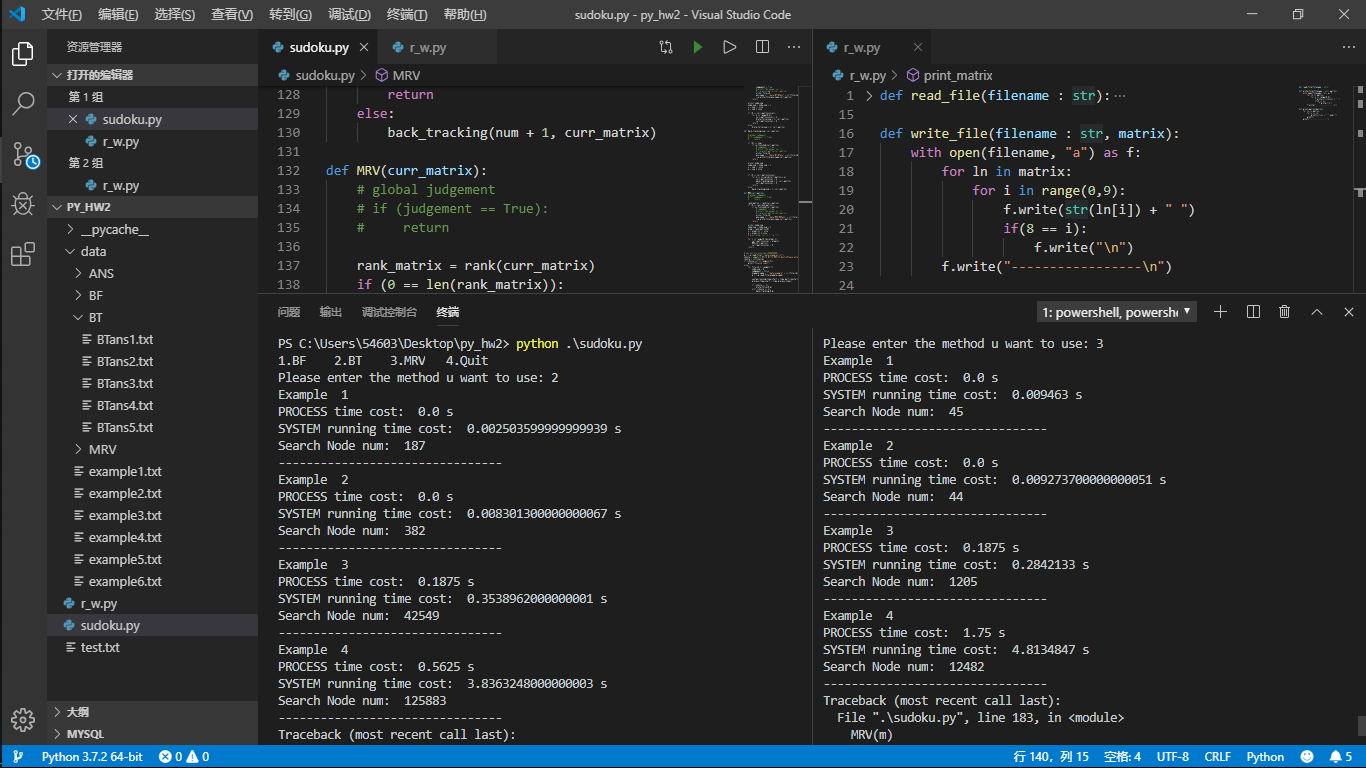
1. 对比



因为time.process\_time()的精度仅仅到微秒，所以在CSP中以及部分MVR中得到的CPU\_time结果是0.0s；相比之下，time.perf\_counter()返回的是使用计时器得到的精确时间，比较直观。

我们可以发现，尽管在MVR的search node明显少于CSP的情况下，MVR的运行速度却是CSP的两倍以上。这是因为对于每一层递归，我们都需要计算出当前数独中的空白格子的可能填入的数字的数量，并对其进行排序，这一步骤是导致时间消耗较多的主要原因。

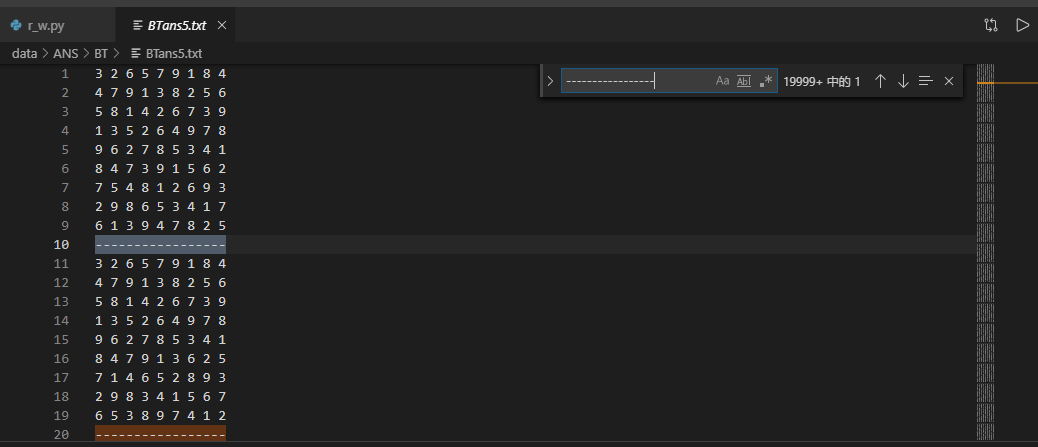
为了更明显的对比，我们在实验过程中将变量judgement（见方法一、二的流程图）注释掉，也即程序会跑完所有情况，求出数独的所有的解。得到的结果如下：



同一种方法的不同测试样例的数据可以看出，数独中空白格子越多，对应的搜索空间（可用search\_node的大小表示）越大，需要递归的次数越多。

同一个测试样例不同的方法的数据…似乎运行时间与空白格子的数量之间的关系不是很明显，猜测这个时间是与当前状态下的电脑的其他进程有关的。

因为可能性太多（test-case4的解答共有400多种，而test-case5得解答：



所以只测试了部分前四个test-case的所有答案。存放在路径为”./data/ANS/”对应的文件夹下。