# 文件说明

## 0.cpp

原始的数独解决程序，按照元素在数独中的位置从上到下，从左到右枚举所有可能性。

缺点：没有任何剪枝与选择操作，因此枚举方案非最优，在一些情况下耗时较多。

因为该程序并非项目重点且我比较懒，不再具体解释其中变量的含义。

效果：https://www.luogu.com.cn/record/92759980

## 1.cpp

改进后的数独解决程序，每次枚举可选数目最少的位置。

具体实现：（在`Search()`中）维护一个数组`total_remain[10]`使其具有性质：

对$i \in \mathbb{N}, i \in [0,9]$，当前数独中有`total_remain[i]`个位置，其可选元素数目为$i$。其中，已填数字位置的可选元素数目为$0$。

这样，遍历`total_remain`可以直接找出最少的可选数目，而不用对$81$个位置分别计算可选数目。（这样做应该比分别计算快...？）

***变量与程序解释：***

**全局变量**

`cell`：记录当前数独局面，元素值为$0$表示位置为空。

`candidate`：`candidate[i][j][k]`的值表示`cell[i][j]`非零时将$k$填入此处是否会产生冲突，其中值为$0$表示不会，值为$i(i \neq 0)$表明会，且产生冲突的因素（有值为$k$的数在同行/列/九宫格）个数为$i$。这样规定是为了方便在递归中回溯。

`cell_remain`：`cell_remain[i][j]`的值代表`cell[i][j]`可选元素的个数，即`cell[i][j]==0`时，`candidate[i][j][k]` $(1 \leq k \leq 9)$ 中零元素的个数。

`total_remain`：对$i \in \mathbb{N}, i \in [0,9]$，当前数独中有`total_remain[i]`个位置，其可选元素数目为$i$。其中，已填数字位置的可选元素数目为$0$。

`solved`：值为$1$表明数独已解决，值为$0$表明数独未解决。

**Input**

输入数独，预处理`candidate`数组。

**InitiateRemain**

预处理`cell_remain`和`total_remain`数组。

`cnt`：记录`cell[i][j]`可选元素的个数。

#### Search

递归主体。当`solved`为1时返回，当数独内所有位置的可选元素数目都为零时判断数独是否已被填好，其余情况按可选元素数目最少的位置枚举。

`minimum`：记录遍历`total_remain`后得到的可选元素个数的非零最小值。当该值不存在时`minimum`为零。

`flag`：用来退出循环的变量。

`current`：记录当前枚举位置的坐标。

#### Output

输出数独。

#### Solve

运用上述函数输入，解出并输出数独。

优点：枚举量少，效率高。

可改进：

1. 存在多个可选数目最少位置时，综合同行/同列/同九宫格上其余格子的可枚举情况决定枚举哪个。
2. 枚举时运用人类解数独的技巧可能可以进一步加快效率。

效果：https://www.luogu.com.cn/record/94620828

## 2.cpp

第九周作业写的数独辅助程序，应该用不上，但因为和数独有关就贴上来了。