程序设计基础
数独大作业实验报告

邓朝一　2019010320
吴成彰　2019010333
孔瑞阳　2019010175

一、完成人与分工说明

算法设计：三人共同讨论

解题部分代码完成：吴成彰

出题部分代码完成：邓朝一

代码规范化、实验报告撰写：孔瑞阳

二、程序功能介绍

1、输入说明：

第一行一个整数 c，表示当前需要执行的操作。

若 c=1，则表示要求出题。
要求：输出的题目只有能唯一解，至少一半以上是空格。

若 c=2，则表示要求解题。
若 c=2，则会继续输入 9 行，每行由 9 个数字和‘-’符号组成，表示当前的题面。‘-’表示当前位置待填。

2、输出说明：

c=1 时：
输出包含 9 行，每行由 9 个数字和‘-’符号组成。‘-’表示当前位置待填。

c=2 时：

【唯一解】：
输出一行“OK”，接下来 9 行输出唯一解。

【无解/题目不合法】：
输出一行 “No_solution”

【多解】：
输出一行“Multiple_solutions”，随后输出 9 行，表示第一个可行解。
然后空一行，
再输出 9 行，表示第二个可行解。

三、代码设计思路

一、主程序包含两个部分：出题、解题

1.1 出题部分调用子程序 int solve_rect(int a[][N]) 来求解

1.1.1 先调用子程序 bool check_valid_rect(int a[][N])
判断这个题目是否合法

1.1.2 进行搜索算法的初始化

1.1.3 调用 int search(int a[][N], int x, int y)
进行 dfs 搜索求解

1.2 出题

1.2.1 调用 int solve_rect(int a[][N])
随机生成一个完整的数独

1.2.2 随机生成一个挖数的顺序

1.2.3 用二分算法并调用 int solve_rect(int a[][N])
计算出符合条件的情况下最多能挖掉多少个数

1.2.4 判断是否能挖掉至少一半格子
不能的话重复 1.2.2 ~ 1.2.4

二、其他一些子程序

2.1 int block_id(const int x, const int y) (x,y)所在的块的编号

2.2 int bit_cnt(unsigned int x) 计算 x 二进制表示中 1 的个数

2.3 int valid_point(const int &x, const int &y)
计算(x,y)这个位置能填哪些数

四、解题算法设计思路

一、总体思路

1、先判断题目是否合法

2、采用 深度优先搜索(dfs)算法 逐步填还没有填的格子

3、在搜索的过程中运用一些剪枝提高程序运行效率

二、剪枝思路

1、采用 columnRes、rowRes、blockRes 三个 int 数组保存 10 位二进制分别表示每一列、每一行、每一块有哪些数可以填
如果 i 可以填，那么第 i+1 位是 1，否则是 0

这样可以快速找到当前填的点有哪些数可以填

2、如果在某个位置填了一个数后，
与它同一列/行/大格中的某个位置没有能填的数，那么填这个数就得不到解，这时候直接继续枚举下一个数，不再进行搜索

即可行性剪枝

3、在可以填的数的数量最少的这些格子中
随机选一个格子进行下一步 dfs

这样可以尽可能快地使得当前 dfs 的数独接近答案，从而提高程序运行效率

五、出题算法设计思路

一、总体思路

1、先随机生成一个完整的数独

2、随机出一个删数的序列，规定按照这个序列的顺序删去生成的完整的数独的一些格子

3、按照序列的顺序尽量删，直到出现多解为止

二、生成数独思路

1、把一个空数独作为题目解题，得到一个完整的数独

2、注意到解题的过程本身就具有随机性，所以生成的数独是随机的

三、生成删数序列思路

1、把所有格子进行 random_shuffle 随机生成序列

2、注意到按照生成序列的顺序删数可能没有删完一半数就出现多解
这种情况下就重新生成一个序列

四、最多删多少数优化思路

1、我们可以注意到不可能出现以下情况：
删的数少有多解，删的数多了反而没有多解

即答案是有单调性的

2、所以采用二分算法优化计算答案

六、大作业过程的体会

邓朝一、吴成彰：
写代码真有趣啊！

孔瑞阳：
看别人写的代码顺便规范一下真痛苦啊。