# 程序设计思路

## 终局生成思路：回溯法

数独游戏的终局生成，本质上是一个空白数独游戏的求解问题。因此，我们可以使用回溯法来解决。按一定顺序或随机选择一个空白格，填入一个数字，然后递归地进行求解，如果求解失败，就回溯到上一步，重新选择数字填入。当所有的空白格都填满时，数独游戏生成成功。

## 数独游戏生成思路：终局生成+挖空

数独游戏生成依赖于回溯法求解。我们首先生成一个数独游戏的终局，然后对终局进行挖空，直到挖空后的数独游戏满足难度要求、唯一性要求和空白格数量要求。

对于难度的设计，参照了综合空白格数量和空格自由度（所有空格所在行、列、块的空格数量）的算法。首先执行预实验，指定随机生成游戏的数量，统计所有空格数对应的空格自由度分布，以空格自由度3:6:11，将难度等级划分为困难、中等、简单。

## 数独求解思路：回溯

用一个数组记录每个数字是否出现。在存储时，我们使用一个长度为9的布尔型的数组，其中第`i`个元素的值为`True`，当且仅当数字`i+1`出现过。例如：我们用`row[2][3]=True`表示数字4在第2行已经出现过，那么当我们在遍历到第 2行的空白格时，就不能填入数字4。

我们首先对整个数独数组进行遍历，当我们遍历到第`i`行第`j`列的位置：如果该位置是一个空白格，那么我们将其加入一个用来存储空白格位置的列表中，方便后续的递归操作；如果该位置是一个数字`x`，那么我们需要将`row[i][x-1]`，`column[j][x-1]`以及`block[i/3][j/3][x-1]`均置为`True`。

当我们结束了遍历过程之后，就可以开始递归枚举。当递归到第`i`行第`j`列的位置时，我们枚举填入的数字`x`。根据题目的要求，数字`x`不能和当前行、列、九宫格中已经填入的数字相同，因此`row[i][x-1]`，`column[j][x-1]`以及`block[i/3][j/3][x-1]`必须均为`False`。

当我们填入了数字`x`之后，我们要将上述的三个值都置为`True`，并且继续对下一个空白格位置进行递归。在回溯到当前递归层时，我们还要将上述的三个值重新置为`False`。