深蓝学院
shenlanxueyuan.com

递归实现数独求解

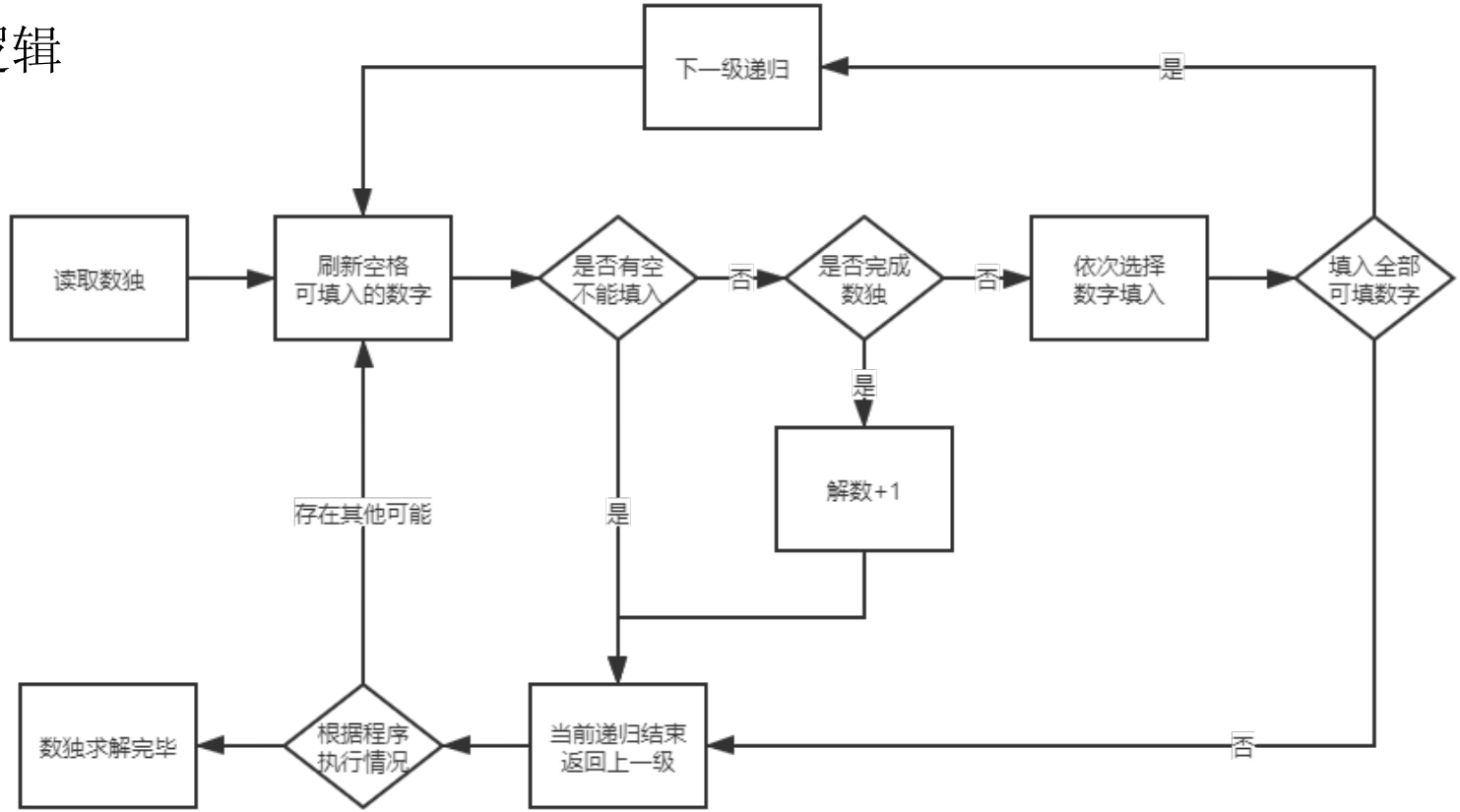
主讲人 腹黑大狸子

纲要

- 第一部分：算法思路
- 第二部分：算法优化

算法思路 — —主逻辑

- 程序递归逻辑

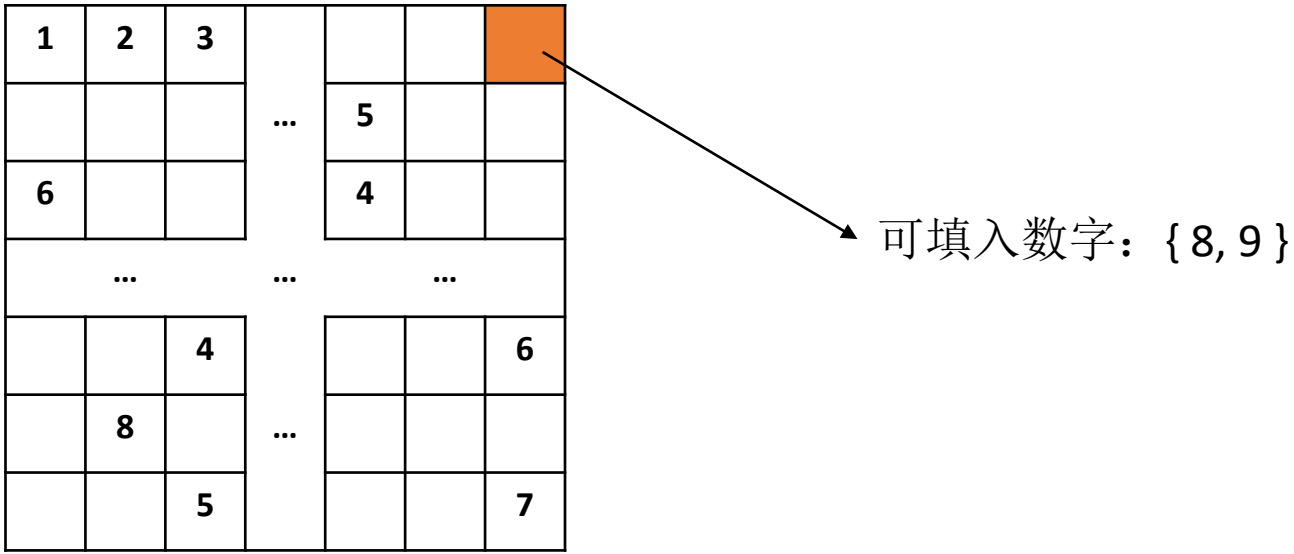


算法思路 — —查询数字

- 存储数独状态：int mSudoku[81]，用index表示对应位置
- 确认可填入的数字
- 对在index位置的数字，其行与列为：
  - int x = index / 9;
  - int y = index % 9;
- 进一步根据行、列和块内的数字，依次判断即可

算法思路 — —查询数字

- 查找可填写数字时，遍历mSudoku数组，对空白的位置求得全部可以填入的数字



算法思路 — —迭代状态

- 定义状态变量
  - 定义bool bHasUpdated，在遍历时查询到可以填入数字的格子时设为true
  - 定义bool bMeetDeadEnd，在出现空白格无法填入数字时设为true
- 迭代遍历81个格子，求得每个空白格可填入的数字
  - bMeetDeadEnd == true，无法继续迭代，返回上一级
  - bHasUpdated == false，说明没有空格，求得一个解，++solveCount
  - 记录当前层级可填的数字availableNumbers和对应空格在数独的位置index，每次选择一个availableNumbers中的数字填入后往下递归
  - 递归返回时需要恢复现场，mSudoku[nextIndex] = 0;

纲要

- 第一部分：算法思路
- **第二部分：算法优化**

算法优化 — 状态压缩

- 算上代表空格的0，总共只有10种数字可以被填入一个格子，我们可以使用int类型变量的不同位来表示该位置是否可以填入对应数字
  - 如某位置可以填入4，则用0x000001000表示
- 定义int mNextAvailable表示可填入的数字
- 当可以填入数字k时，我们将mNextAvailable从右往左第k位设为1
  - 即：mNextAvailable |= (1 << k)
  - 异或也是可以的，mNextAvailable ^= (1 << k)

算法优化 — —状态压缩

- 找到对应空格，需要取出每一个可能填入的数字时，遍历mNextAvailable的每一位，判断是否为1，这可以使用与操作实现
- 使用数字状态压缩可以将空间消耗降为4个字节

```
int available = mNextAvailable;
for (int i = 0; i < 9; ++i)
{
    if (available & 1)
    {
        mSudoku[nextIndex] = i + 1;
        Solve();
        mSudoku[nextIndex] = 0;
    }
    available >>= 1;
}
```

算法优化 — —选择空格

- 由于有很多个格子可以填入数字，为了减少分支，可以选择当前状态下可以填入数字最少的空格进行递归
- 定义int minAccessibleLength = INT_MAX，用于记录在遍历数独空格时每个格子可以填入的数字个数的最小值
- 用countOnes记录当前空格可以填入的数字的数量，并用变量记录最小值对应的空格位置和可填入数字

```
if (countOnes < minAccessibleLength)
{
    mNextIndex = i;
    mNextAvailable = tempNext;

    minAccessibleLength = countOnes;
}
```

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening