Pythonによる数独の自動生成プログラムの制作

東京都立南多摩中等教育学校５年 須山 凌

数独とは

9×9のマス目にルールに則って数値を埋めるパズルである．列・行・また3×3ごとに区切った9マス分の各ブロックで重複しないように1~9の数字を記述する．その数独の問題を自動生成するプログラムを制作した．

基本となる構想

数独の完成形を自動生成した後，一部の数字を空欄にすることで数独の問題を生成する．

実装上の課題

ランダムソートを繰り返すアルゴリズムを検討したが，答えの生成に10分以上要した．これは作問に対し，著しく時間がかかっているため，高速化が喫緊の課題であると考えた．

解決策

初めに解決策として各列・各行で数字の重複を順に解消するアルゴリズムを検討したが，やがてルールに則りながら重複を解消することが不可能な状態になる場合が多いとわかった．

ここで新たに次のような考察を行った．実現不可能なパターンを複数回生成し却下するため処理に時間がかかる．ただしランダムソートを用い，実現不可能なパターンを却下する方法で高速化できる．この考察に従い，新たなプログラムを試作した．

１マスずつランダムに数字を入れ，ルールに適さないマスがあれば，ランダム生成過程で複数候補があったマスまで戻り，別のパターンを繰り返す．この過程のフローチャートを図1に示す．これにより１秒以内にルールに適合した数独の完成パターンの生成を実現した．

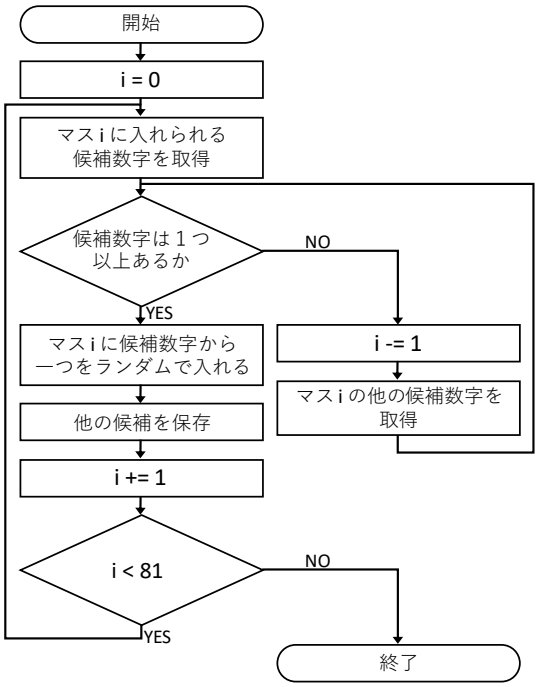

図１ 解決策のフローチャート

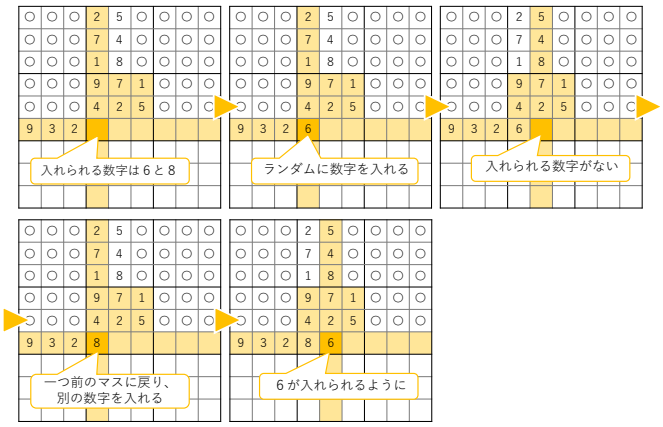

図２ 解決策のイメージ

改良

さらなる処理の高速化を図るため，アルゴリズムの改良のため，プログラム全体を通して細かい処理の改善を行った．

例えば，二次元配列の利用により，各マスを列番号と行番号で取得できるようになった．これにより，単純なfor文で行やブロックの全ての数字を取得することが可能になり，改良前の一次元配列を用いていたプログラムに比べ，計算量の削減に成功した．また，配列同士の比較に，これまで用いていたfor文による配列同士の比較ではなく，集合の演算を利用することで可読性と処理速度が向上した．

さらに，問題を生成する際に，マス目を一定数分まとめて消去することによる処理の高速化を行った．
これらにより，0.1秒未満で問題を作成することができるようになった．（図３）

```
試行回数：177
完成した答え
[[8 2 7 6 5 4 1 9 3]
 [6 9 4 3 1 2 5 8 7]
 [3 1 5 8 7 9 6 2 4]
 [9 6 3 4 2 5 7 1 8]
 [4 7 1 9 3 8 2 5 6]
 [2 5 8 1 6 7 4 3 9]
 [5 3 6 7 8 1 9 4 2]
 [7 4 2 5 9 3 8 6 1]
 [1 8 9 2 4 6 3 7 5]]
空白の数：44
完成した問題
[[0 2 0 6 0 0 1 0 0]
 [6 9 0 0 0 0 5 0 7]
 [0 0 0 8 7 9 0 2 4]
 [0 0 3 0 2 5 0 0 0]
 [4 0 1 0 0 0 0 0 6]
 [0 0 8 1 0 7 4 3 0]
 [5 3 6 0 0 0 9 0 2]
 [7 0 2 0 9 3 0 0 1]
 [1 0 0 0 4 0 0 7 5]]
CPU times: user 10.4 ms, sys: 0 ns, total: 10.4 ms
Wall time: 10.4 ms
```

図３ 改良版プログラムの出力
（Google Colaboratoryを利用）

結果と今後の展望

情報の科学で問題解決を学んだが，今回のプログラム実装までの試行錯誤は，まさにそれを体験していると感じた．

今後は複数の生成アルゴリズムを実装し，その組み合わせ方による処理速度の変化や，256マスや625マスなどの変則的な数独を生成すると処理速度はどのようになるのかなどを調べていきたい．