数独（後編）

問題

ミニプロジェクト 5 で作成した Board クラスをもとに、ミニプロジェクト 6 では、第 12 回の講義で紹介した SearchProblem のサブクラス Sudoku として数独を定義する。探索問題においては何を状態としてとらえるかが重要である。グラフ探索ではノードが状態を表し、迷路問題では迷路における（x, y）座標が状態を表していた。Sudoku では 9×9 のマスに書かれた数字の配置、すなわち Board オブジェクトが状態を表す。Sudoku のコンストラクタは数独の問題にある数字の配置に対応した Board オブジェクトを受け取り、これを初期状態とする。従ってプログラムの書き出しは次のようになる。

```
class Sudoku(SearchProblem):
    def __init__(self, board):
        self.board = board

    def get_start_state(self):
        return self.board
```

Sudoku は SearchProblem のサブクラスなので、残る is_goal と next_states の 2 つのメソッドを実装すれば、あとは深さ優先探索 dfs に Sudoku オブジェクトを渡せば、初期状態からすべて数字が埋まった終状態への道順を示してくれる。[10 点]

仕様

実装すべきメソッド:

- is_goal(self, board)　　board オブジェクトで表わされる状態（数字の配置）が数独の解になっているかどうかを True または False で返す
- next_states(self, board)　board オブジェクトで表わされる状態（数字の配置）においてどれか空いているマスを選んで、そこに数独のルール上許される数字を記入して得られる状態（Board オブジェクト）のリストを返す（先読みして、さらに候補を減らしても可）

注意

- 2019 年 1 月 28 日（月）深夜までに sudoku.py を e シラバスにアップロードすること。
- 採点にあたってはプログラムの正確性のみならず効率も評価する。すなわち、なるべく余計な探索を減らすように next_states を工夫すること。また、妥当なタイムアウト値を設け、実行時間が異常に長いプログラムは採点の途中で打ち切ることがある。
- ミニプロジェクト 5 の締め切り後に board.py の解答例を newton にアップロードする。採点にあたってはそのプログラムを使うので、自前の board.py の使用は、あくまでも解答例が公開される前の開発時に限定すること（ミニプロジェクト 5 の問題文に示した仕様にはない使い方をしていると採点環境ではプログラムが動作しない可能性があるため）。