CS 325 HW 8 - Sudoku

Naomi Grant

December 2020

1 Overview

My Sudoku board is a dictionary where the keys are square locations (A3, B6, H4, etc.) and the values are the numbers in those squares. The rows are labeled by the letters A-I and the columns are numbered 1-9. Here is what the user sees:

```
~~~~~ SUDOKU ~~~~~
Game Rules: The 9x9 board is divided into 9 larger squares (each with 9 squares inside).
The goal is to get the numbers 1-9 in each big square.
Each number can only appear once in a row, column, or big square.

Playing Instructions:
Use the letters on the left and numbers on the top of the board to see the names of the squares (Ex: A6, G2, etc.).
To enter a number into a square, you enter the square name and the number you wish to put there, separated by a space.
For example, 'A4 3' would put a '3' in spot 'A4'.
To make the spot empty again, just put an underscore in place of a number: 'A4 _'

Choose a difficulty level (easy, medium, hard): easy
Here is your easy Sudoku Board:
  1 2 3   4 5 6   7 8 9
A _ 8 _ | 9 _ 3 | _ _ _
B _ 9 7 | 6 _ _ | 5 _ _
C 4 6 _ | 5 _ _ | 8 9 _
  ------|-------|------
D _ 3 2 | _ 7 _ | 9 _ _
E 7 _ 8 | _ _ _ | 2 _ 1
F _ _ 4 | _ 5 _ | 3 6 _
  ------|-------|------
G _ 7 6 | _ _ 2 | _ 1 9
H _ _ 3 | _ _ 6 | 7 5 _
I _ _ _ | 7 _ 5 | _ 3 _

Reminder: Enter 'q' to quit, 'submit' when done, 'solution' to give up and
 see the fully solved board, or 'help' to see the instructions again.
Enter square name and number:
```

2 Verification

To be clear, the user is only allowed to enter values 1-9 into the squares so the verification does not need to check for invalid entries. The board also has to be completely filled out - if there are empty spaces on the board, it will return False and will print to the user, "Sorry, your solution is incorrect." Because a half filled board is obviously not the correct answer to the puzzle.

The verification process is as follows:

It scans through each row, column, and 3x3 square to make sure each number 1-9 only appears once per row/column/3x3 square. To check one row or column, it starts with an empty string called nums_seen and if the number in the current square is NOT in that string, it adds the current number to nums_seen. If it is in nums_seen already, then False is returned to indicate that there is a duplicate and that the user's solution is incorrect. nums_seen is reset to " after checking a particular row or column.

For example, this is how it would check each column:

```
for col in '123456789':
    nums_seen = ''
    for row in 'ABCDEFGHI':
        if board[row + col] not in nums_seen:
            nums_seen += board[row + col]
        else:
            return False
```

2.1 Time Complexity of Verification

For a standard Sudoku board, it is a constant input size - a 9x9 grid. This means that the verification of it would have O(1) constant time complexity. If this were checking a much larger $n^2 \times n^2$ grid, I would obviously need to change some things about the way my algorithm runs, since it's designed specifically for a 9x9 grid. But if we just look at the basic structure of it, it still completes in polynomial time - $O(n^2)$.

3 NP-Completeness

The decision question, "Given a Sudoku Instance, does it have any solutions?" can be proven to be NP-Complete as follows: By reducing a known NP-Complete problem, such as the Latin Squares problem, to Sudoku, it is proved to be NP-Hard. We have shown above that a solution to Sudoku can be verified in polynomial time, which means it is in NP. By proving that it is both in NP and NP-Hard, Sudoku is proven to be NP-Complete.

README:

To play the Sudoku game, run sudoku.py and follow the printed instructions in the terminal.

1. It will print the game rules and instructions as follows:

Game Rules: The 9x9 board is divided into 9 larger squares (each with 9 squares inside). The goal is to get the numbers 1-9 in each big square. Each number can only appear once in a row, column, or big square.

Playing Instructions: Use the letters on the left and numbers on the top of the board to see the names of the squares (Ex: A6, G2, etc.). To enter a number into a square, you enter the square name and the number you wish to put there, separated by a space. For example, 'A4 3' would put a '3' in spot 'A4'. To make the spot empty again, just put an underscore in place of a number: 'A4 _'

2. Then it will ask you to select a difficulty level (easy, medium, hard). You can also enter 'test' to print a correctly filled out full board.

3. After entering the difficulty level, your puzzle will be printed, along with: Reminder: Enter 'q' to quit, 'submit' when done, 'solution' to give up and see the fully solved board, or 'help' to see the instructions again. Enter the square name and number:

4. After entering a square name and number, it will update and print the board again, print the reminder message, and ask for another square name and number. This process will continue until you enter 'q', 'submit', or 'solution' (as seen in reminder message).