Artificial Intelligence Sudoku Write Up

Total time to run through all 128 puzzles (seconds): 190.030389

Total number of guesses: 1060115

**Quick History of Sudoku**

The word Sudoku comes from the Japense words with “Su” meaning number and “Doku” meaning single. Ironically enough, the Sudoku puzzles originated from Switzerland and founds its way to Japan later. Euler, a Swiss native, is credited with the inception of Sudoku by combining Greek and Latin characters. In 1979, the modern Sudoku was formulated by Howard Garnes, an employee at American Dell magazine. Garnes mainstreamed the puzzle calling it Number Place. Upon arrival to Japan, Number Place was dubbed the term Sudoku.

**What is Sudoku?**

Sudoku is an analytical puzzle, resting in logic and unique number placement. The objective is to fill a grid with the correct positioning of symbols without any repeats in the distinguished rows, columns, or blocks. This rule is crucial to the solving of a Sudoku; every symbol must only appear once in every group. Prior to solving a Sudoku puzzle, each symbol but one must appear at least once or else it is impossible to solve.

FUN FACT: The 2016 World Sudoku Championship will be held in Slovakia!

**My Sudoku Solver**

Good definitions to keep in mind:

group – either a row, column or block; a series of positions, where symbols can only appear once

cell – position in a Sudoku grid

unmarked – a cell in the puzzle that is unfilled, denoted by “.”

**Principles**

*Principle 1: If a position is unsolved (represented by a “.”) and all of the symbols except one exist in that cell’s neighbors, that position must be that one symbol.*

*Principle 2: If a symbol only appears once as a possibility in all of a group’s unmarked cells, at the position the symbol is a possibility, the symbol must go in cell of that position in the puzzle.*

**Methods**

1. positions

At the start of my code, I call positions which has one argument of the side length of the Sudoku. In positions, I create the global data structure called “aGroups” which is a list of lists. Each of the lists are positions of the rows, columns or blocks. The data structure of aGroups is used in my cellNeigh, validateSudoku, and makeDeductions.

1. symbols

The method, symbols, is another method that created an important global data structure

called “allSyms” which is a list of all the possible symbols in the puzzle. allSyms is used in findPossible and makeDeductions.

1. cellNeigh

cellNeigh creates another important global data structure called “cellNeighbors”. It has one argument (puzzle). cellNeighbors is a list of sets. The indexes of the list are all of the positions in the puzzle and the sets contain all of the neighbors of that position based on its appearance in its unique three groups in aGroups. cellNeigh is used in findPossible and makeDeductions.

1. findPossible

findPossible implements *Principle 1*. findPossible creates dictP, which is a dictionary

with the key as an unmarked position and the value is a set of all possible symbols at that cell.

*Principle 1: If a position is unsolved (represented by a “.”) and all of the symbols except one exist in that cell’s neighbors, that position must be that one symbol.*

The method goes through all of the positions in the puzzle and when it comes across and empty position, it loops through all of the positions of the cellNeighbors creating a set of found symbols. Next, I create the set notfound which includes all of the symbols not in found. If the length of notfound is one, I immediately set the cell I am currently out to the symbol that was not found in its neighbors. If not, I set the the value of the position in dictP to be notfound.

Pseudocode:

def findPossible(puzzle):

dictP = {}

for each position in puzzle that is unmarked:

for each neighbor in cellNeighbors[pos]:

if neighbor is not blank:

add it to the found set

for each symbol in allSyms

if symbol is not in found:

add it to notfound

if the length of notfound is 1:

set the puzzle[position] to the symbol in notfound

else:

dictP[position] = notfound

1. validateSudoku

The method validateSudoku is used to check if the guess taken is correct and follows the

primary rule of Sudoku: no repeats in any of the groups. To do this, I loop through all of the groups in aGroups and create a set called visited for each group. Then, I go through all of the filled positions in the group, adding the symbols to the set if it is not already in visited. However, if the symbol already resides in visited, the method returns False, because there should be no repeats within a given group. Otherwise, it will return True, indicating the current guess works.

1. makeDeductions

makeDeductions implements *Principle 2.*

*Principle 2: If a symbol only appears once as a possibility in all of a group’s unmarked cells, at the position the symbol is a possibility, the symbol must go there in the puzzle.*

makeDeductions performs its task by first passing the puzzle to findPossible to implement Principle 1. After receiving a new puzzle and calculated dictP, makeDeductions loops through all of the groups in aGroups. For each group, it creates a dictionary with the keys as all of the symbols in allSyms and the values are empty sets. While going through each unmarked position in the group if the position has a symbol that appears as a possibility, it is added to the set for that symbol.

Then, upon going through each symbol in allSyms, if there is only one position that it appears at as a possibility, the symbol is added to the puzzle at that position. Right after, the symbol is removed as a possibility for all of the position’s neighbors. The position is also removed from dictP. This is then returned to bruteForce with a deduced puzzle and dictP, to make the next guess.

1. bruteForce

bruteForce is a recursive method that checks if a solution is correct and accordingly guesses if there are no deductions left to be made. If the puzzle is validated as False, then the method returns an empty string (“”). If not, it finds the position at which the least amount of guesses there are to be made by looking through dictP. Then, for all the possibilities at dictP of that position, guesses are made with that symbol in that position.

FUN FACT: Sudoku has been used as an ailment for depression, blood pressure, and Alzheimer’s.