# Assignment 7: Solve the Sudoku Problem using a Monte Carlo Solution

## Learning Outcomes

* Solve Sudoku using Monte Carlo

## Motivation

Sudoku is a challenging puzzle, even for a computer. It's known as a constraints solving puzzle. Here are the constraints.

* Each puzzle is a 9x9 grid.
* Each row of 9 values should have the values 1 through 9, each appearing exactly once.
* Each column of 9 values should have the values 1 through 9, each appearing exactly once.
* The grid is divided into nine 3x3 smaller grids (see below). These smaller grids are called a "sector".
* Each sector should also contain the values 1 through 9, each appearing exactly once.
* Some of the values in the puzzle will be given to you ahead of time. The values are not allowed to be changed or moved.
* The blanks will be represented by a zero.

The Sectors.

A A A | B B B | C C C  
 A A A | B B B | C C C  
 A A A | B B B | C C C  
 ---------------------  
 D D D | E E E | F F F  
 D D D | E E E | F F F  
 D D D | E E E | F F F  
 ---------------------  
 G G G | H H H | I I I  
 G G G | H H H | I I I  
 G G G | H H H | I I I

Note: This is a matter of dispute among some sudoku puzzle creators, but it's also believed that each puzzle should have one and exactly one solution. While this is true most of the time, I'm going to give you a sample puzzle in which I know there are potentially many solutions. We are throwing this requirement out for this assignment.

## Inputs

You will be given 81 integers ranging from 0 (the blanks) to 9 (the set values). I'll give you code ahead of time which does the task of reading in values and displaying values. Your task should be focusing on the algorithm.

## Outputs

You should display the original grid and the solved grid.

## How to solve the problem.

Here's the core to solving Sudoku with a Monte Carlo approach. It's slightly different than what was described in class.

// Copy the original into a new vector named `board`.  
// Load all sectors with initial values within the new `board`.  
// Compute the energy of the board. Save this to `energy`.  
  
// while energy is greater than 0  
 // Select a random sector  
 // Get two location from that sector which are swapable  
 // Swap those two locations  
 // Compute the new energy. Save this to `newenergy`  
  
 // If the new energy is better or equal to energy, save that to energy.  
 // Else  
 // Get a random number less than 100  
 // If this random number is zero, save the new energy  
 // to energy in order to pretend that the score did get better.  
 // Else,  
 // Undo the swapped locations. (Simply call swap again)  
  
// Return the board

Here's how to compute the energy of a grid.

* Imagine that you have a row containing this: [3 6 3 4 7 2 3 1 2]
* Since 4 appears once, that's an energy of 0 (for each duplicate).
* Since 2 appears twice, that's an energy of 1 (for each duplicate).
* Since 3 appears three times, that's an energy of 2 (for each duplicate).
* For each row an column, add up the cumulative energy and return it.
* A solved grid will have an energy of 0.

Here are the other functions that I need you to write.

* getRandomNumber(n): Returns a random number from 0 to n-1
* loadSectorWithInitialValues(sector, sectorSize, board): Loads one sector with a random order of values
* loadAllSectorsWithInitialValues(sectorSize, board): Loads all sectors with initial values
* computeEnergy(sectorSize, board): Computes the energy of the entire board. A solve board will have an energy of 0.
* getTwoCellsInASector(sector, sectorSize, original, &a, &b): Gets two random locations and deposits those locations into a and b
* swapTwoLocations(a, b, board): Swaps two locations within the board. Great for also undoing a move.

Code that you don't have to write.

* Code that reads in the board.
* Code that displays the board.
* Code that finds the open locations in a sector.
* Code that finds the unused values in a sector.

## Examples.

For each of these examples, I'm including the execution time using the Unix time command.

Example 1. My favorite puzzle. Just one blank!

5 6 8 4 7 2 3 1 9  
 2 3 9 6 5 1 8 4 7  
 1 4 7 8 3 9 6 5 2  
 6 2 1 3 8 5 7 9 4  
 7 9 3 1 0 6 2 8 5  
 4 8 5 9 2 7 1 3 6  
 9 7 6 5 1 3 4 2 8  
 8 1 2 7 9 4 5 6 3  
 3 5 4 2 6 8 9 7 1  
  
 5 6 8 4 7 2 3 1 9  
 2 3 9 6 5 1 8 4 7  
 1 4 7 8 3 9 6 5 2  
 6 2 1 3 8 5 7 9 4  
 7 9 3 1 4 6 2 8 5  
 4 8 5 9 2 7 1 3 6  
 9 7 6 5 1 3 4 2 8  
 8 1 2 7 9 4 5 6 3  
 3 5 4 2 6 8 9 7 1  
  
  
real 0m0.005s  
user 0m0.000s  
sys 0m0.000s

Example 2. An easy puzzle.

0 6 0 0 0 0 0 1 0  
 0 0 0 6 5 1 0 0 0  
 1 0 7 0 0 0 6 0 2  
 6 2 0 3 0 5 0 9 4  
 0 0 3 0 0 0 2 0 0  
 4 8 0 9 0 7 0 3 6  
 9 0 6 0 0 0 4 0 8  
 0 0 0 7 9 4 0 0 0  
 0 5 0 0 0 0 0 7 0  
  
 5 6 8 4 7 2 3 1 9  
 2 3 9 6 5 1 8 4 7  
 1 4 7 8 3 9 6 5 2  
 6 2 1 3 8 5 7 9 4  
 7 9 3 1 4 6 2 8 5  
 4 8 5 9 2 7 1 3 6  
 9 7 6 5 1 3 4 2 8  
 8 1 2 7 9 4 5 6 3  
 3 5 4 2 6 8 9 7 1  
  
  
real 0m0.502s  
user 0m0.488s  
sys 0m0.000s

Example 3. A hard puzzle

This took almost one minute to solve.

0 8 0 0 0 0 0 0 5  
 1 0 0 4 0 0 3 8 0  
 0 0 0 0 2 0 1 0 0  
 0 0 0 1 0 0 9 0 6  
 0 0 0 6 7 4 0 0 0  
 3 0 2 0 0 8 0 0 0  
 0 0 6 0 4 0 0 0 0  
 0 5 8 0 0 6 0 0 9  
 2 0 0 0 0 0 0 6 0  
  
 4 8 3 7 9 1 6 2 5  
 1 2 9 4 6 5 3 8 7  
 6 7 5 8 2 3 1 9 4  
 8 4 7 1 3 2 9 5 6  
 5 9 1 6 7 4 8 3 2  
 3 6 2 9 5 8 4 7 1  
 9 3 6 2 4 7 5 1 8  
 7 5 8 3 1 6 2 4 9  
 2 1 4 5 8 9 7 6 3  
  
  
real 0m58.339s  
user 0m58.136s  
sys 0m0.016s

Example 4. All zeros!

Suprisingly, all zeros is rather short. 2 seconds. This can have multiple answers.

0 0 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0 0  
  
 2 6 7 8 1 9 5 3 4  
 5 9 1 7 3 4 2 8 6  
 8 3 4 5 2 6 9 7 1  
 6 5 8 4 7 3 1 9 2  
 3 1 9 2 6 8 7 4 5  
 7 4 2 1 9 5 8 6 3  
 1 7 6 3 8 2 4 5 9  
 4 2 3 9 5 7 6 1 8  
 9 8 5 6 4 1 3 2 7  
  
  
real 0m1.756s  
user 0m1.740s  
sys 0m0.000s

## Turn it in

Upload a zip file containing the files that use used or wrote yourself to the drop box on D2L:

* Make sure your name, CSCI 4270, and the Programming Assignment Number appear in comments in all of your files.
* Note: NO CREDIT will be given to programming assignments that do not compile.
* Make sure you have compiled and tested your program before handing it in.