Final Report: Sudoku Solver

Members:

1. Shaheer Kamal
2. Badar Azeemi
3. Ammar Khawaja

Goal:

	a	b	c	d	e	f	g	h	i
1			9				7	2	8
2	2	7	8			3		1	
3		9					6	4	
4		5			6		2		
5			6				3		
6		1			5				
7	1			7		6		3	4
8				5		4			
9	7		9	1			8		5

Fig.1. *An example of a Sudoku puzzle.*

Sudoku is a widely known board game where the goal is to fill numbers in empty boxes such that no number reappears in any row, column or box. There are currently different variants of Sudoku such as 4x4 grids, 9x9 grids and 16x16 grids; our work is focused on a classic and regular Sudoku of 9x9 board. Our purpose is to develop two different effective algorithms and compare their computing times to analyze which method is more efficient and how they both differ. The two algorithms we have implemented are the:

1. The Brute Force/Naive Approach
2. The Backtracking Algorithm

Algorithms:

The Brute Force/Naive Approach:

The simplest method randomly produces a solution to the board by trying all possible combinations in the empty spaces; once the board is completely filled, it checks whether the board is valid. If it is not valid, a different combination is tried from the pool of permitted numbers i.e. 1 to 9; otherwise, the output is returned. This algorithm can take a considerable amount of time if the solution to the board is complex and occasionally fails to solve the board effectively. The complexity of this algorithm is O(n^2) since there are n (9 for a common board) possibility in a particular cell with n spaces.

Back-Tracking Approach:

This algorithm finds an empty space and assigns the lowest valid number by considering the numbers in the same row, column and box. However, if none of the numbers from 1 to 9 fit in a particular empty space, the algorithm backtracks recursively to the previous empty space that was filled recently. For a traditional sudoku board, as mentioned earlier, there are 9 different possibilities at every index, hence the time complexity is O(n^2). This might seem odd but is simple the upper bound for the runtime. This algorithm relatively takes lesser time compared to the brute force approach.

Challenged Faced:

Thinking of two different recursive algorithms for the same problem was challenging and took some time to be executed efficiently.

DSA Techniques Used:

The nested list was used to simulate our sudoku board which was then converted into a proper board with rows and columns using a helper function. Dictionaries were used in helper functions to keep track whether any number was being repeated in a row, column and box. The time library was used to record the runtime of the different approaches and matplotlib was used to graph them for our analysis.