

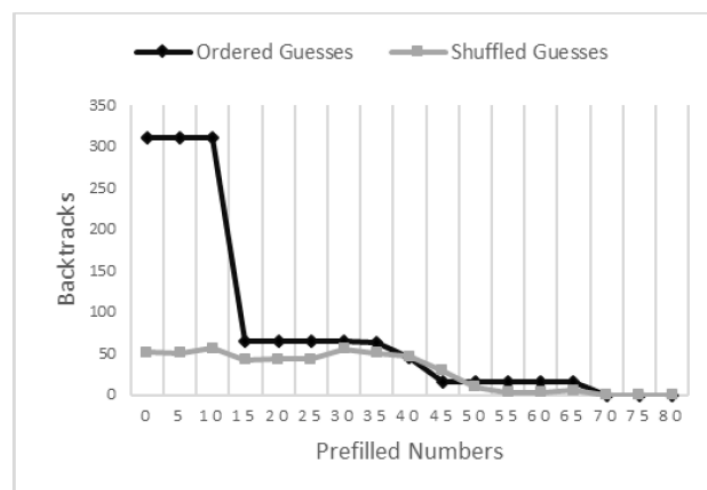
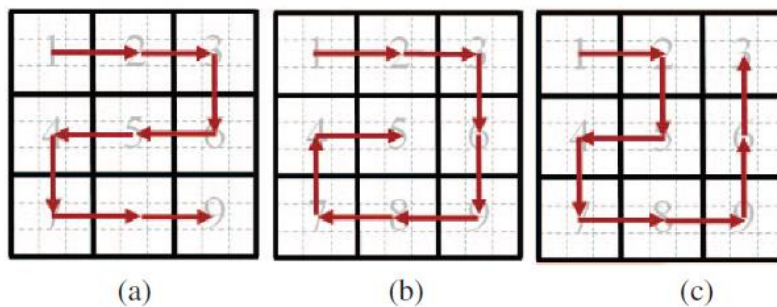
**Research Topic:** Sudoku Solver

**Objective:** Be able to solve a 9x9 sudoku puzzle in an efficient manner.

**Sample data:**

[1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	[1,7]	[1,8]	[1,9]
1			2			3		
[2,1]	[2,2]	[2,3]	[2,4]	[2,5]	[2,6]	[2,7]	[2,8]	[2,9]
[3,1]	[3,2]	[3,3]	[3,4]	[3,5]	[3,6]	[3,7]	[3,8]	[3,9]
[4,1]	[4,2]	[4,3]	[4,4]	[4,5]	[4,6]	[4,7]	[4,8]	[4,9]
4	7	1		5		6		4
[6,1]	[6,2]	[6,3]	[6,4]	[6,5]	[6,6]	[6,7]	[6,8]	[6,9]
	3		2			4	7	6
[9,1]	[9,2]	[9,3]	[9,4]	[9,5]	[9,6]	[9,7]	[9,8]	[9,9]
	4		8	7		3	9	

**Sample Approaches:**



## Sample Situations:

<b>6</b>	$\begin{smallmatrix} 2 & 4 \\ 5 \end{smallmatrix}$	<b>7</b>	$\begin{smallmatrix} 2 & 3 \\ 8 & 9 \end{smallmatrix}$	$\begin{smallmatrix} 2 & 3 \\ 8 & 9 \end{smallmatrix}$	<b>1</b>	$\begin{smallmatrix} 3 & 4 \\ 5 \end{smallmatrix}$	$\begin{smallmatrix} 3 & 4 \\ 5 \end{smallmatrix}$	$\begin{smallmatrix} 3 & 4 \\ 5 \end{smallmatrix}$
$\begin{smallmatrix} 2 & 4 \\ 5 \end{smallmatrix}$	<b>8</b>	$\begin{smallmatrix} 2 & 4 \\ 5 \end{smallmatrix}$	<b>7</b>	$\begin{smallmatrix} 2 & 3 \\ 2 & 3 \end{smallmatrix}$	$\begin{smallmatrix} 2 & 3 \\ 2 & 3 \end{smallmatrix}$	<b>1</b>	<b>6</b>	<b>9</b>
<b>9</b>	<b>3</b>	<b>1</b>	<b>6</b>	<b>4</b>	<b>5</b>	<b>2</b>	<b>8</b>	<b>7</b>
<b>1</b>	$\begin{smallmatrix} 2 & 4 \\ 2 & 4 \end{smallmatrix}$	<b>6</b>	<b>5</b>	$\begin{smallmatrix} 2 & 3 \\ 2 & 3 \end{smallmatrix}$	<b>7</b>	<b>8</b>	<b>9</b>	$\begin{smallmatrix} 2 & 3 \\ 4 \end{smallmatrix}$

(a)

<b>6</b>	$\begin{smallmatrix} 2 & 4 \\ 5 \end{smallmatrix}$	<b>7</b>	$\begin{smallmatrix} 2 & 3 \\ 8 & 9 \end{smallmatrix}$	$\begin{smallmatrix} 2 & 3 \\ 8 & 9 \end{smallmatrix}$	<b>1</b>	$\begin{smallmatrix} 3 & 4 \\ 5 \end{smallmatrix}$	$\begin{smallmatrix} 3 & 4 \\ 5 \end{smallmatrix}$	$\begin{smallmatrix} 3 & 4 \\ 5 \end{smallmatrix}$
$\begin{smallmatrix} 2 & 4 \\ 5 \end{smallmatrix}$	<b>8</b>	$\begin{smallmatrix} 2 & 4 \\ 5 \end{smallmatrix}$	<b>7</b>	$\begin{smallmatrix} 2 & 3 \\ 2 & 3 \end{smallmatrix}$	$\begin{smallmatrix} 2 & 3 \\ 2 & 3 \end{smallmatrix}$	<b>1</b>	<b>6</b>	<b>9</b>
<b>9</b>	<b>3</b>	<b>1</b>	<b>6</b>	<b>4</b>	<b>5</b>	<b>2</b>	<b>8</b>	<b>7</b>
<b>1</b>	$\begin{smallmatrix} 2 & 4 \\ 2 & 4 \end{smallmatrix}$	<b>6</b>	<b>5</b>	$\begin{smallmatrix} 2 & 3 \\ 2 & 3 \end{smallmatrix}$	<b>7</b>	<b>8</b>	<b>9</b>	$\begin{smallmatrix} 2 & 3 \\ 4 \end{smallmatrix}$

(b)

**Datasets:**

9 million sudoku puzzles and solutions: <https://www.kaggle.com/datasets/rohanrao/sudoku>

**Techniques:**

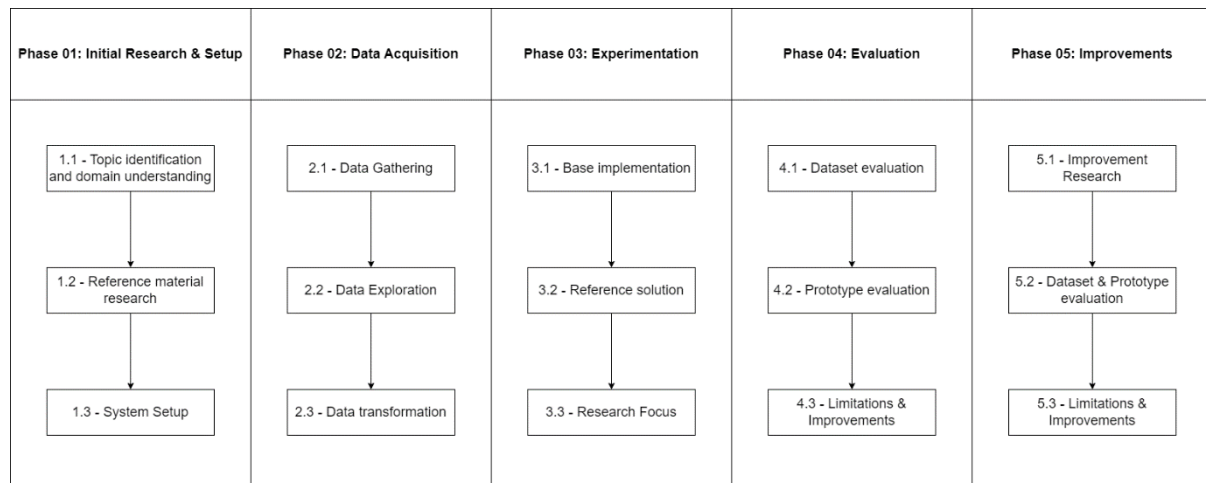
Brute Force (Naïve) with backtracking

Rule based with backtracking

Ant Colony Genetic Algorithm

Tree-based

Neural Network

**Suggested Approach:****1 Initial Research & setup**

- 1.1 Understand the rules of the game are, what data is available and what approaches exist.
- 1.2 Find academic literature about topic, find dataset(s), find reference material for techniques (code, tutorial, documentation).
- 1.3 Setup GIT, Python, Anaconda, Python Virtual Environment (optional), IDE (VSCode).

**2 Data acquisition**

- 2.1 Download dataset or scrape website, understand structure.
- 2.2 Create Jupyter notebook to load dataset. Explore it, identifying nulls, target variable, different target classes, explore correlation, explore distribution (is target variable balanced/unbalanced).
- 2.3 Prepare data for processing (convert from string to 2-dimensional array, create validation and completion check algorithms).

**3 Experimentation**

- 3.1 Create a simple implementation, even heavily based on 3<sup>rd</sup> party, naïve brute force with backtracking, searching by row and guessing in a sequential manner is a good start.
- 3.2 Identify another research and determine what they did differently. Add different searching and guessing algorithms. Revise your code to approach 3<sup>rd</sup> party research so you can compare.
- 3.3 The research focus is to be able to complete the puzzle in a reasonable time with the least backtracks.

**4 Evaluation**

- 4.1 Compare your dataset and the changes you made with 3<sup>rd</sup> party research.
- 4.2 Compare the outcome of your technique with that of other parties (it does not need to be better but be able to determine how yours compares).
- 4.3 Reflect and identify areas of improvements.

**5 Improvements**

- 5.1 Consider rule based and other approaches.
- 5.2 Consider pushing to an API.
- 5.3 Consider generating your own puzzles.
- 5.4 Consider adding tracking of solutions.
- 5.5 Consider code optimization or GPU use (NVIDIA CUDA).
- 5.6 Calculate the time complexity to solve problems using your algorithms (big O notation).