# An Algorithmic approach to Solve Sudoku

Submitted To : Dr.Sumit Miglani

Submitted By:


| | | |
|---|---|---|
| Sachin Pandey | 101603291 | 2COE20 |
| Sachin Shandilya | 101603292 | 2COE20 |
| Sai Siddartha Maram | 101603293 | 2COE20 |
| Sajal Aggarwal | 101603294 | 2COE20 |

# Acknowledgement

My team would like to express great appreciation for Dr. Sumit Miglani for providing us with this opportunity to prove our skills in the form of a project which includes various concepts of Data Structures and Algorithms. Thapar Institute of Engineering and Technology has played a pivotal role in inculcating a practical approach and letting students explore their passion and come up with innovative ways to solve problems.

# Table of Contents

## History

The first Sudoku puzzle was created in 1979. In New York City the Sudoku puzzle appear first, which was published by the specialist puzzle publisher Dell Magazines in their magazine Dell Pencil Puzzles and word Games. First it was printed under the name “number place”. Howard Garns, a retired architect and freelance puzzle constructor designed this first puzzle. This mathematical construction is inspired by the Latin square, invention of Leonhard Euler. Later the puzzle was introduced in Japan by Nikoli during 1984 as "Suji wa dokushin ni kagiru", which can be translated as "the numbers must be single" or "the numbers must occur only once", later it abbreviated as Sudoku.

## Fundamental Rules of Sudoku

Solving a Sudoku puzzle can be rather tricky, but the rules of the game are quite simple. Solving a sudoku puzzle does not require knowledge of mathematics; simple logic suffices. The objective of sudoku is to enter a digit from 1 through 9 in each cell, in such a way that:

I. Each horizontal row contains each digit exactly once

II. Each vertical column contains each digit exactly once

III. Each subgrid or region contains each digit exactly once.

### Scope of Project

You can make your own Sudoku and at any Step you can go back to One Step as well as you can see the Solution of it. It is manually a very difficult job to perform and its need a lot of recalling , reminding and mathematical calculation. The game of “Sudoku” helps to increase mental thinking , vision e.t.c

## Objective and Problem Statement

The objective of the project is to solve sudoku using the our understanding of various algorithms and data structures. Taking multiple algorithms into consideration which we are equipped with in the process of learning we have put forward a solution which looks to solve the world famous sudoku puzzle in a few seconds.

The core algorithms and data structures include

- Graph using Adjacency List
- Graph Colouring also known as Chromatography
- Backtracking
- Recursion

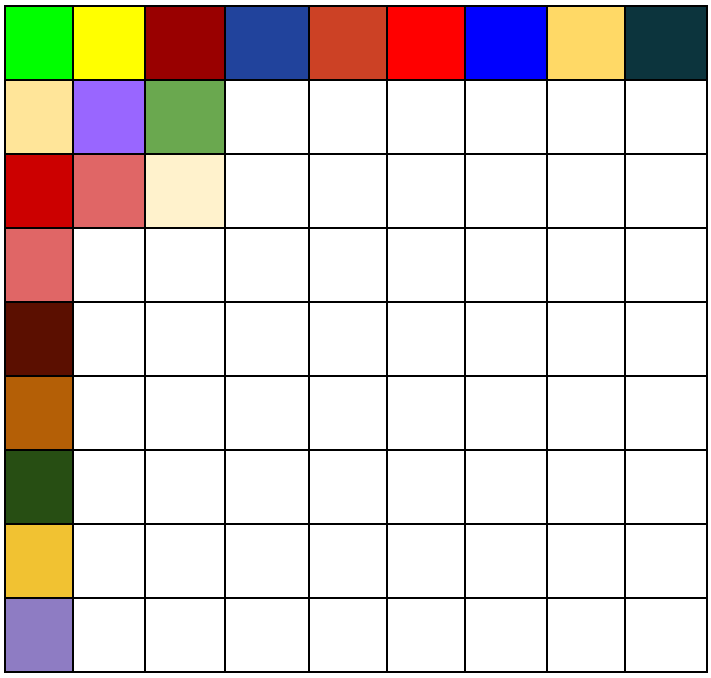
## Analysis and Techniques Used:

### Step 1:

Importing Required tools and establishing certain repeated statements the below code snippet of code calls in libraries required for input and output apart from that also creates statements which will be used for the allocation of memory of different vertices.

### Step 2:

After declaring the data structures it is equivalently important to initialize the data structures when called. A sudoku grid consists of 81 vertices with adjacent neighbours within a column, row and 3 x 3 box connected. On close observation in the above data structure of the vertices we see the the vertex number, the colour it carries, and an Admin Number only to identify the fact if the block is a previously given block in the question or not. If true, the player will not be able to change the state of the block.



### Step 3:

After declaring the vertices we should be able to connect these vertices with their neighbours. It is important to note the fact that adding a neighbour does not mean placing the whole node, but means only pointing to the address of it. Adding of an edge is mutual, Say we have 2 vertices A and B, we are made to add edges one from A -> B and also the vice versa.

### Step 4:

Since there exist only 9 numbers in the classical sudoku game, It is fine if we identify the vertex with colour. So we will develop a function to call display all the colours which is directly the number.

### Step 5:

In the perspective of the user the graph will be treated as a grid with coordinates, So to make the project easy to use, we need a way to convert 2D axis into graph vertices, for this the team has developed a function which stands as follows,

### Step 6:

While the user is playing the user is certain to try out a lot of combinations, and in this process it is important we check every move of his, to this we find all the neighbour colours and if the entered move by the user matches the colour with any of its neighbours then this move is treated wrong and is asked to play again on the same vertex or is free to move on and solve any other coordinate.

### Step 7:

In the vertex data structure it is seen we carry an array of 9 possibilities which indicate what could be the number, initially with -1 at every step, the possible solutions i.e which is not in the neighbour are made to zero, and hence indicating the fact that these are allowed.

### Step 8:

Application of Backtracking, Before we jump into the application of backtracking we have to take into consideration, if there is only a single possibility to enter, that is the correct answer is pretty straight forward, When the entering element is not straight forward, we will have to brute force and check each and every value which would be appropriate to fill in that particular space, This is where the concept of backtracking comes into the picture.

### Step 9:

Brute Force Solving Method A simple way to solve a Sudoku puzzle is to simply try filling each blank square with the numbers 1 to 9 until a valid solution is found. We brute-force every number a.k.a color and check weather this will lead to a better solution or not, incase it does not we stop and move on the next best combination, In this way we keep iterating through all the

blocks and see if it leads to a better solution, If it does we carry on with the procedure else we discard this particular route and try another combination.

### Step 10:

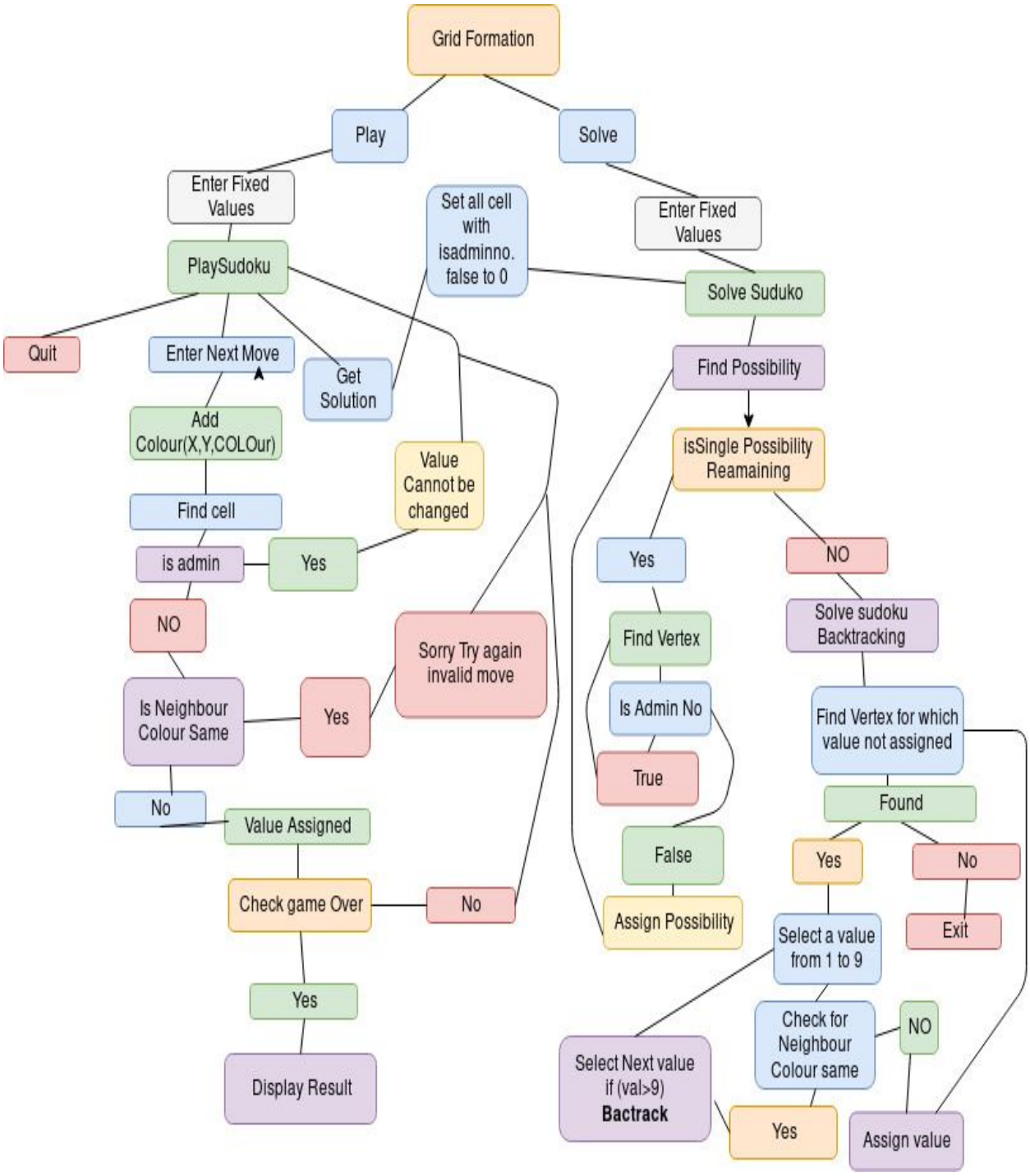
In this way, we enter the main function and allow the user two play in 2 modes, that is either enter his/her own question and let the computer solve it, or provide our own question and allow the user to solve it. The version of our game is interactive, challenging to solve and definitely a great way to spend time. With this our team would like to end the discussion on how our excellent algorithmic approach helped us solve the problem

## Project Requirements

The project to be fully functional expects the following requirements.

1. A decent C compiler since the code is executed in C.
2. A computer with a constant power supply.
3. A computer with enough processing power. Generally a regular 4GB Ram Computer with a processor to support it will solve the purpose.

## Project Flow Chart



## Conclusions and Results

### Achievements

Through generating this Sudoku solver as a team we have improved our programming ability. This was perhaps the largest program in terms of time invested and lines of code written that as a team we have created. The code is of the highest quality, and it is severely lacking in documentation, but some of the problems posed by the project were a good challenge to solve. Writing the solver has demonstrated the advantages of ‘ smart’ algorithms over naïve algorithms evidenced in the measurements above. Finally, we have experienced participating in what could be called a small research project (useful for future work). The application of various algorithms helped us to understand the importance of Data Structures and Algorithms in solving real life problems.

### Project Status:

The logic solver portion of the program is sufficient for solving many Sudoku puzzles. However, the implementation could likely be improved to execute faster. Furthermore, there are many logic solving techniques that have not been implemented. The generator, on the other hand, is currently rather poorly implemented. It works in that it successfully generates grids of a given number of initial entries, but the variability of the time taken to generate is a significant weakness.

## SUGGESTION FOR FUTURE IMPROVEMENT

As a team we look to improve on certain grounds. Instant help by giving hints and users progress while playing. II. Score system based on time and accuracy, and database to keep track of top ten record.

**References**

1.http://en.wikipedia.org/wiki/Sudoku

2.http://www.sudoku.com/

3.http://sudoku.sourceforge.net/

4.http://www.scanraid.com/Sudoku.htm

5.http://www.conceptispuzzles.com/products/sudoku/rules.htm

6.http://www.setbb.com/phpbb/index.php?mforum=sudoku