SUDOKU GAME

A PROJECT REPORT

BACHELOR OF TECHNOLOGY IN

INFORMATION TECHNOLOGY

Submitted by:

Prateek Jaiswal (2K19/IT/098)

Priyanka Aggarwal (2K19/IT/099)

Under the Supervision of:

Mrs. Swati Sharda

Asst. Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of

Engineering) Bawana Road, Delhi-

110042

NOVEMBER, 2020

CERTIFICATE

I hereby certify that the Project Dissertation titled “SUDOKU GAME ” which is submitted by Prateek Jaiswal - Roll No – 2K19/IT/098; & Priyanka Aggarwal - Roll No – 2K19/IT/99; INFORMATON TECHNOLOGY, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of the Bachelor of Technology, is a record of the project work carried out by students under my supervision.

Place: Delhi

Date: 30-11-2020

SWATI SHARDA

SUPERVISOR

ABSTRACT

In this report, we present the detailed development and implementation of simple Sudoku game. The Sudoku game consists of graphical user interface, solver and puzzle generator; implemented using HTML,CSS,JAVASCRIPT,NodeJS. The solver and generator is implemented using efficient algorithm. The solver finds the solution to the puzzles generated by the generator as well as to the puzzles entered by the user. Generator creates various number of different Sudoku puzzles. This project gives an insight in to the different aspects of java programming.

INTRODUCTION

Sudoku is the Japanese abbreviation of a phrase meaning the digits must remain single, also known as ***Number Place***, where Su means number, doku which translates as single or bachelor. Sudoku is not a mathematical or arithmetical puzzle. It works just as well if the numbers are substituted with letters or some other symbols, but numbers work best. The aim of the puzzle is to enter a numerical digit from 1 through 9 in each cell of a 9×9 grid made up of 3×3 subsquares or subgrids, starting with various digits given in some cells; each row, column, and subsquares region must contain each of the numbers 1 to 9 exactly once. Throughout this document we refer to the whole puzzle as the **grid/game board**, a 3x3 subgrid as a **block** and the individual grids that contains the number as a **cell**.

AIM: To make a Sudoku Game with Solver and Sudoku Generator.

ALGORITHM USED:

Method 1: Simple **(Not used in this Project)**

Approach:

The naive approach is to generate all possible configurations of numbers from 1 to 9 to fill the empty cells. Try every configuration one by one until the correct configuration is found, i.e. for every unassigned position fill the position with a number from 1 to 9. After filling all the unassigned position check if the matrix is safe or not. If safe print else recurs for other cases.

Algorithm:

1. Create a function that checks if the given matrix is valid sudoku or not. Keep Hashmap for the row, column and boxes. If any number has a frequency greater than 1 in the hashMap return false else return true;
2. Create a recursive function that takes a grid and the current row and column index.
3. Check some base cases. If the index is at the end of the matrix, i.e. i=N-1 and j=N then check if the grid is safe or not, if safe print the grid and return true else return false. The other base case is when the value of column is N, i.e. j = N, then move to next row, i.e. i++ and j = 0.
4. if the current index is not assigned then fill the element from 1 to 9 and recur for all 9 cases with the index of next element, i.e. i, j+1. if the recursive call returns true then break the loop and return true.
5. if the current index is assigned then call the recursive function with index of next element, i.e. i, j+1.

Complexity Analysis:

- Time complexity: O(9^(n*n)).
  For every unassigned index there are 9 possible options so the time complexity is O(9^(n*n)).
- Space Complexity: O(n*n).
  To store the output array a matrix is needed.

Method 2: Backtracking **(Used in this Project)**

Approach:

Like all other Backtracking problems, Sudoku can be solved by one by one assigning numbers to empty cells. Before assigning a number, check whether it is safe to assign. Check that the same number is not present in the current row, current column and current 3X3 subgrid. After checking for safety, assign the number, and recursively check whether this assignment leads to a solution or not. If the assignment doesn't lead to a solution, then try the next number for the current empty cell. And if none of the number (1 to 9) leads to a solution, return false and print no solution exists.

Algorithm:

1. Create a function that checks after assigning the current index the grid becomes unsafe or not. Keep Hashmap for a row, column and boxes. If any number has a frequency greater than 1 in the hashMap return false else return true; hashMap can be avoided by using loops.
2. Create a recursive function that takes a grid.
3. Check for any unassigned location. If present then assign a number from 1 to 9, check if assigning the number to current index makes the grid unsafe or not, if safe then recursively call the function for all safe cases from 0 to 9. if any recursive call returns true, end the loop and return true. If no recursive call returns true then return false.

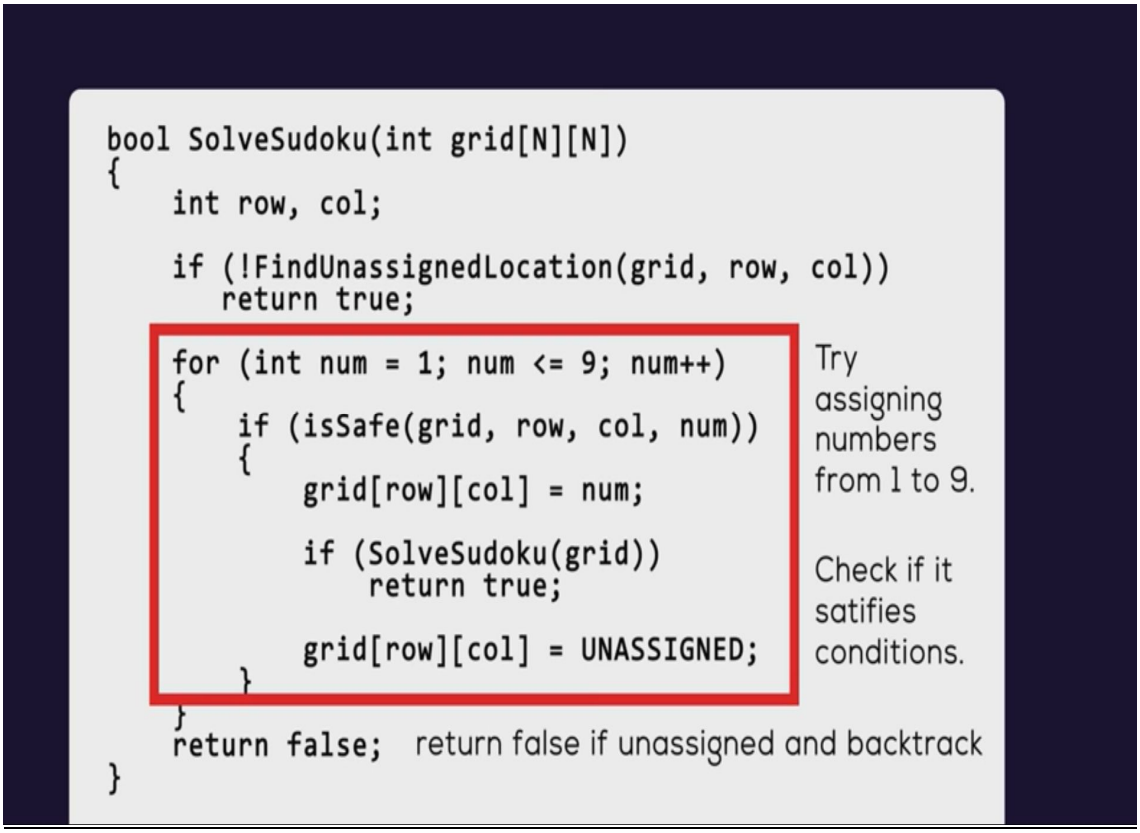
4. If there is no unassigned location then return true.

Complexity Analysis:

- **Time complexity:** O(9^(n*n)).
  For every unassigned index, there are 9 possible options so the time complexity is O(9^(n*n)). The time complexity remains the same but there will be some early pruning so the time taken will be much less than the naive algorithm but the upper bound time complexity remains the same.

- **Space Complexity:** O(n*n).
  To store the output array a matrix is needed.

ALGORITHM:



LANGUAGE:

HTML

CSS

JAVASCRIPT

FRAMEWORK:

Node.js

BootStrap

jQuery

DEPLOYMENT:

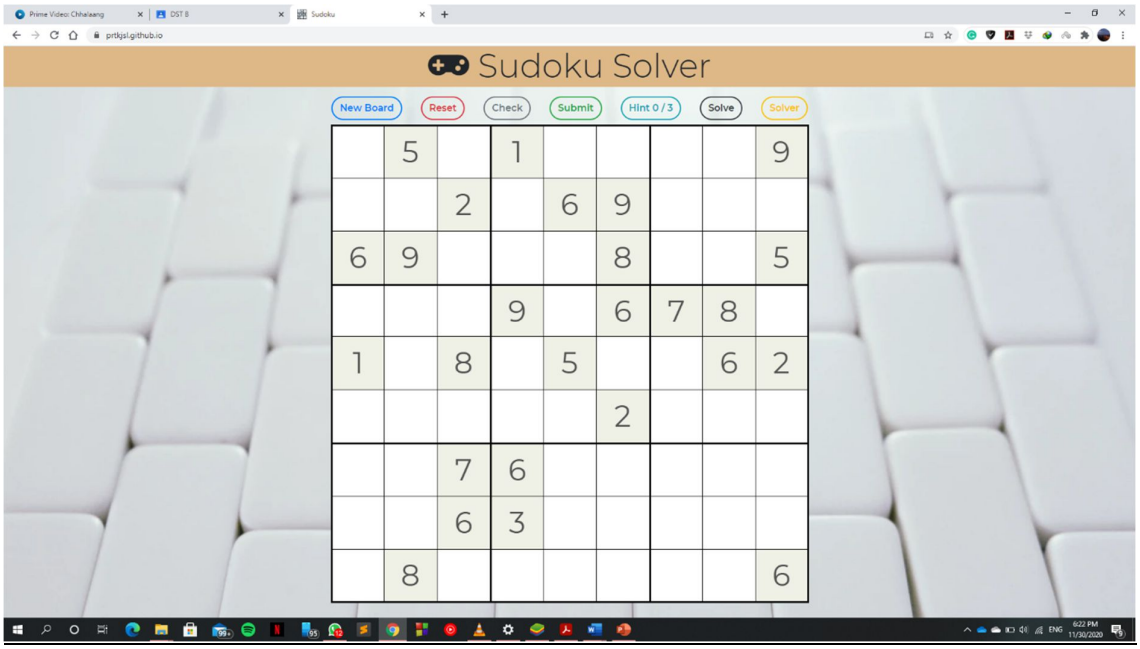
GitHub

WEBSITE:

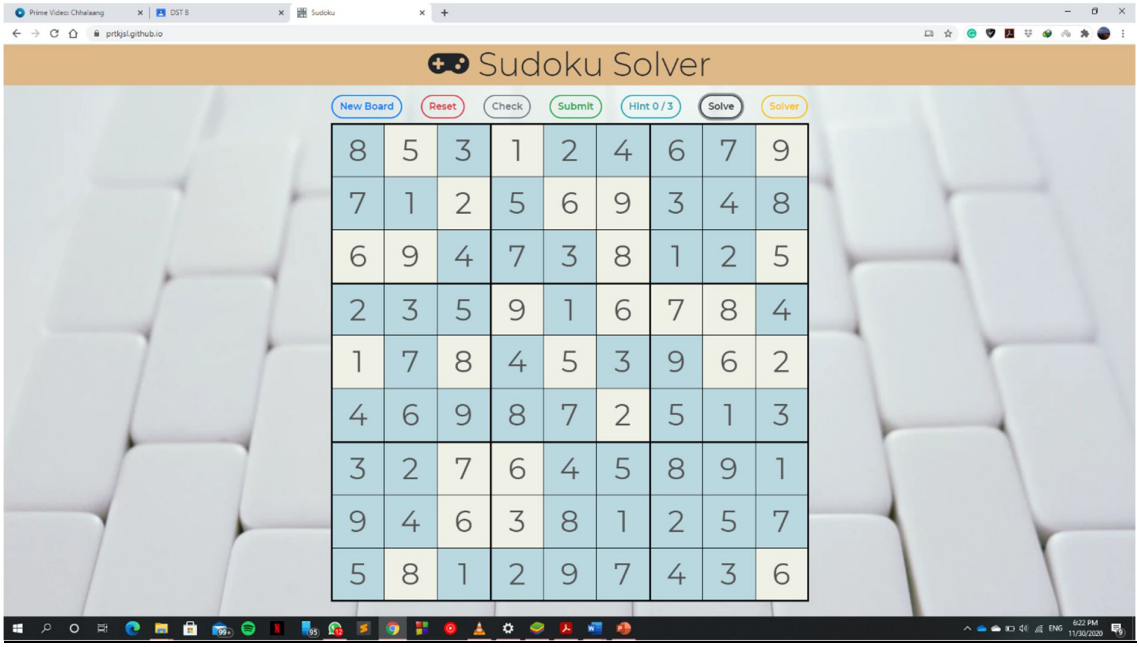
https://prtkjsl.github.io/

SCREENSHOTS:

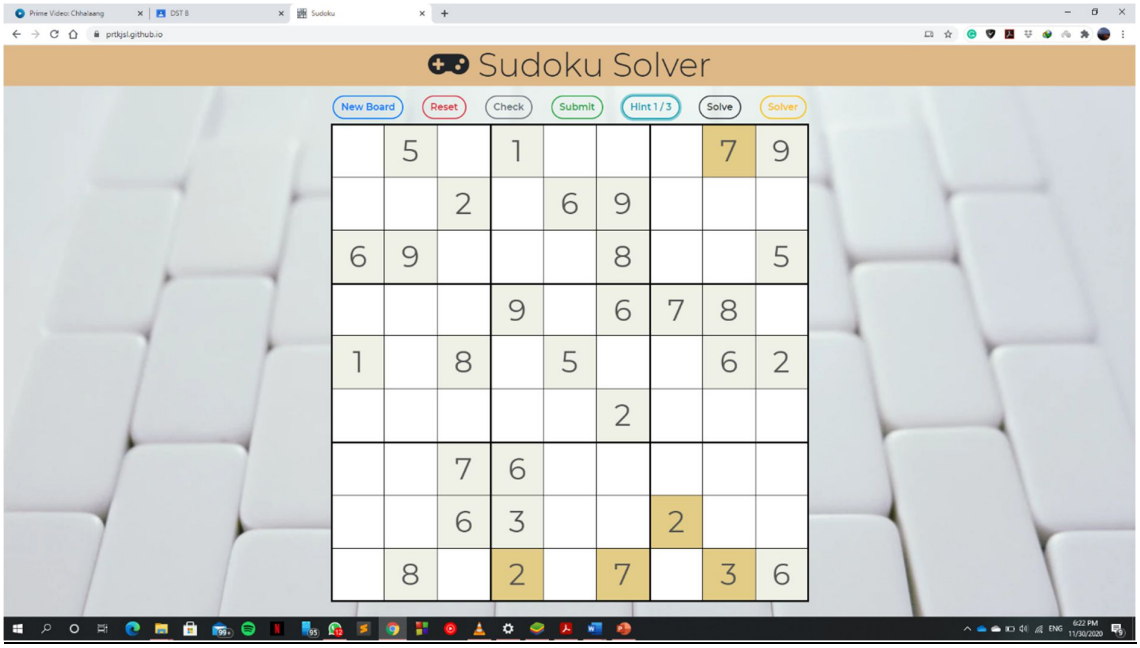
New Game



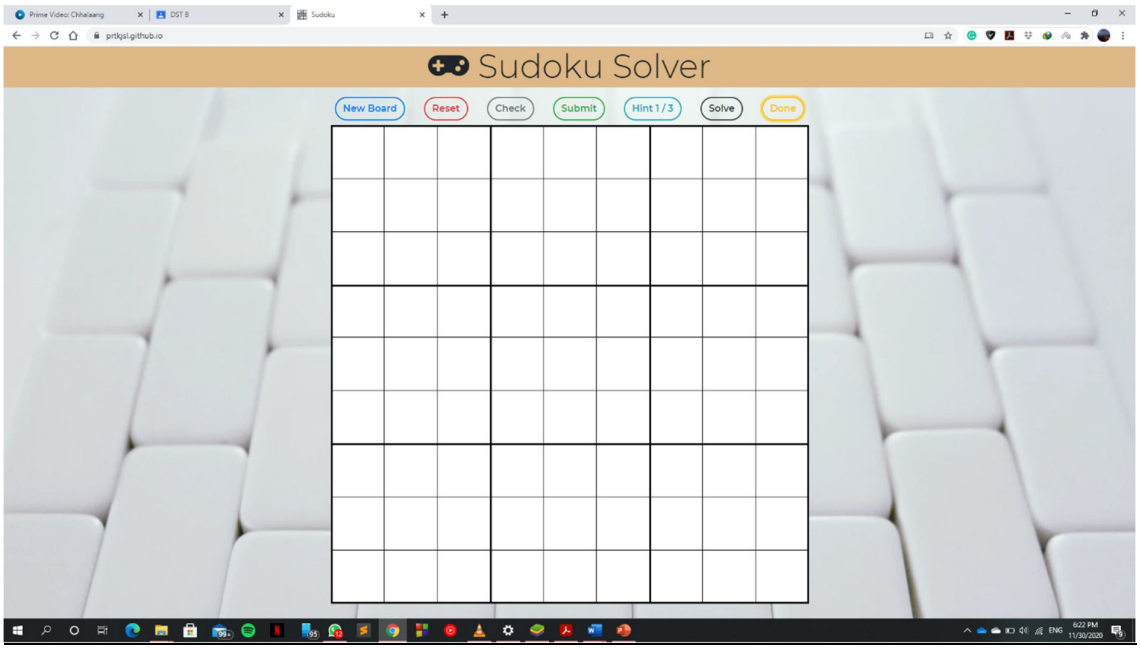
Solve



Hints (Max 3)



Solver (To solve our own Sudoku)



Check (Red are Wrong)