Project 2

Web-based Sudoku Puzzle Solver

Due October 30, 11:59pm

This project gives hands on practice with Javascript programming and updating an HTML page using the Document Object Model (DOM). In this project you will implement a backtracking recursive function to solve a partially completed Sudouku puzzle.

Sudoku Puzzles

The objective of Sudoku is to fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 sub-grids that compose the grid (also called ”boxes”, ”blocks”, or ”regions”) contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution. To learn more about Sudoku and how to solve by hand see this short youtube video: https://www.youtube.com/watch?v=OtKxtvMUahA

Depth-First Search Solution

The objective of this project is to solve the one given Sudoku puzzle using a recursive depth-first search function. The depth first search solution starts at the first empty grid cell and plugs in the lowest value between 1 and 9 that satisfies the requirements of Sudoku. It then continues to the next empty cell doing the same. Once it reaches a cell with no solution, it back tracks to the last grid cell it set, and resets that cell to 0. It then tries the next number that satisfies the requirements. It continues to back track and move forward until it finally finds a solution.

The algorithm you should use is as follows:

```
for each grid cell
   if grid cell is empty
     // we try possible numbers
     for numbers 1 – 9
        check if number satisfies Sudoku requirements
       if number satisfies Sudoku requirements
             set grid cell to number
             if(recursive call returns true)
                   return true; //we continue
             else
                 set grid cell back to 0 //try next number
      return false; //this is returned if no number works so we backtrack
return true; //returns true when all cells are filled with correct values
```

Note: I will give you the freedom to solve the puzzle any way you see fit, but the depth first search algorithm provided here is known to work and is recommended for this project. If you try a different approach, the TAs and Instructor may not be able to help.

Project Set Up

Create a folder called project2 under htdocs. Download starter code from the class github repository. The starter code is in the project2/sudoku_starter folder. There are 3 files - index.html, sudoku.js and sudoku.css. In the project2 folder in htdocs put the three files - index.html, sudoku.js and sudoku.css.

Your solution should go in sudoku.js only. If you'd like to modify the sudoku.css to change the design of the table or button feel free, but it will not be graded.

Class github repository URL:

https://github.com/kmaloneVillanova/CSC2053

To get the code from github, go to your local git class repository from the terminal or Git Bash. Issue the command: git pull origin master

Instructions

1. The index.html file contains the html of the table and the button. The original partially completed sudoku puzzle should be displayed on load.

2. When the user clicks the solve button, the table should display the solved puzzle.

Submission

Upload your code to webster.csc.villanova.edu. Submit sudoku.js to blackboard. In the text submission box, enter the URL to your sudoku website on webster.

Grading

80 points - Implementation

30 points - On load index.html displays the original partially filled grid sudoku puzzle.

50 points - The puzzle is solved and displayed when the user clicks the solve button.

20 points

The code is in the correct files and uploaded correctly to webster and/or blackboard.

EXTRA CREDIT – 10 points

Chapter 10 of the text book, Graphs, describes the depth first search of graphs using a stack instead of recursion. Implement a second function called solveStack() that solves the sudoku puzzle using a stack instead of recursion. Add a button to your page that gives the option to solve using recursion or solve using a stack.