**Playing Sudoku on console**

*By Rui Huang and Yilong Zhang*

1. **Implementation**

We implemented the sudoku game in Java. The structure of our idea goes like this:

**class** backTrackingSearch

**{**

**Members:**

1. Basic information about size of the sudoku game, like its length and width;
2. Store the assignment, variables, and domain of the the variables in stacks;
3. Variables representing the set of current variables and their respective domains..

**Public methods:**

1. **boolean** backTrackingSearch\_(Character[][] assignment, int L, int W)

Following the pseudocode, the function backTrackingSearch takes in the assignment as well as the size info. defined by the user, and initialize the backtracking algorithm.

1. **Character[][]** getSolution()

Return the found assignment of sudoku to user.

**Private methods:**

1. **boolean** Recursive\_Backtracking(Character[][] assignment)

This is the core algorithm of back tracking algorithm following the recursive idea of the pseudocode in the textbook.

1. **void** sudokuSizeSet(int L, int W)

Set the size of the sudoku game.

1. **void** initialization(Character[][] assignment)

Initialize the variables, as well as the domain of each variable.

1. **boolean** isAssignmentComplete(Character[][] assignment)

Check whether the current assignment if complete. A complete assignment contains no blanks, and elements in the same row, same column, and same small square are different with each other.

1. **boolean** open3InARow(Seed theSeed)

check if there exists the case open 3-in-a-row for the player or the AI. If yes, place on the 4th position to block or to win.

1. **int**[] selectUnassignedVariable(Character[][] assignment)

This function takes in the current assignment and select one variable from the unassigned variables using Minimum Remaining Value heuristic and Degree Heuristic. If there are multi-variables with the same MRV and degree, we will randomly select one among them.

1. ArrayList<**int**[]> MRV()

Realization of the Minimum Remaining Value heuristic.

1. ArrayList<**int**[]> degreeHeuristic(Character[][] assignment, ArrayList<int[]> MRV\_indice)

Realization of the Degree Heuristic, used as the 2nd tie breaker.

1. **boolean** isConsistent(int[] var, Character value, Character[][] assignment)

Check whether or not the picked out value of the selected variable is consistent with the assignment up to now.

1. **boolean** forwardChecking(int[] var, Character value)

Forward checking the unassigned variables that are related to the current variable. Remove the values that are conflict with the current value of the selected variable from domains of the unassigned variables.

1. Character[][] deepCopyAssignment(Character[][] assignment)

Deep copy of the current assignment.

1. ArrayList<int[]> deepCopyVariables(ArrayList<int[]> variables)

Deep copy of the current variables.

1. ArrayList<Character[]> deepCopyDomains(ArrayList<Character[]> domains)

Deep copy of the domains.

**}**

**class** startHere

**{**

**Constants:**

sudoku\_length and sudoku\_width.

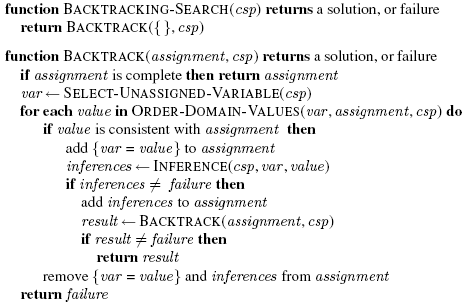
**Public methods:**

1. **static void** main(String[] args)

Prompt indications for user to input the initial state of sudoku game, one line at a time. Use the character ‘x’ for blanks, and no blank between each character.

**}**

1. **Ideas**



The main idea of our code follows the pseudocode in the textbook, which is shown on last page.

The current assignment is represented by a 2D array of characters with the size of . The remaining variables are stored in a 1D array, with each element representing its x and y indices. Accordingly, the domains of the remaining variables are stored in another 1D array, with each element representing the domain of each variable.

The minimum remaining values and degree heuristic are used in selecting the next variable from the group of unassigned variables. If there are still multi variables with the minimum remaining value and the maximum degree heuristic, one variable will be randomly selected from them. Once selected, each value in the domain of the selected variable will be tested whether it is conflict with the current assignment or not.

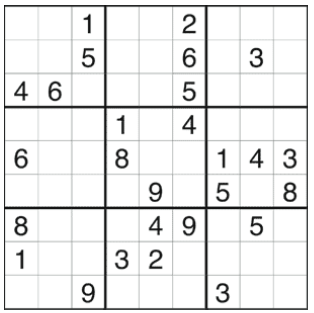
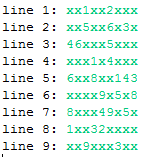
If yes, the top elements of each stack will be popped out, and the assignment, variables, as well as domains recover their previous status.

If not, this variable will be assigned with this value and it will be used to forward check and update the domains of rest variables. After these are done, this value of the current selected variable will be added into the assignment. Accordingly, the array of variables and domains will be updated. Then the current assignment, variables, and domains will be pushed into three stacks.

In this way, we have used a stack to help realizing the depth-first tree search.

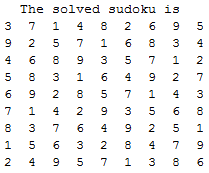
1. **Output**

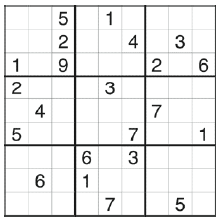
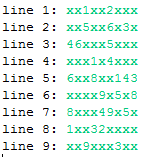
Number the row and column from 0 to 8. The MRV and Degree heuristic are used as the 1st and 2nd heuristic, respectively. If there are multi-variables with the MRV and same largest degree, then we randomly choose a variable among them.

For the first instance, the first selected variable is at [3, 1]. Domain size of this selected variable is 9, and its degree is 16; the second selected variable is at [8, 1]. Domain size of this selected variable is 8, and its degree is 14; the third selected variable is at [0, 1]. Domain size of this selected variable is 7, and its degree is 13.

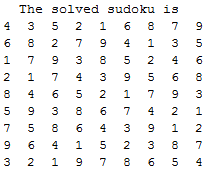
The CPU execution time is 703 ms.

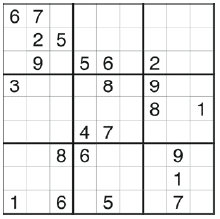
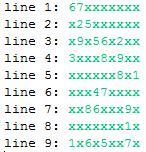


For the second instance, the first selected variable is at [8, 1]. Domain size of this selected variable is 9, and its degree is 16; the second selected variable is at [8, 6]. Domain size of this selected variable is 8, and its degree is 15; the third selected variable is at [8, 8]. Domain size of this selected variable is 7, and its degree is 14.

The CPU execution time is 3016 ms.



For the third instance, the first selected variable is at [7, 5]. Domain size of this selected variable is 9, and its degree is 17; the second selected variable is at [5, 5]. Domain size of this selected variable is 8, and its degree is 16; the third selected variable is at [0, 5]. Domain size of this selected variable is 7, and its degree is 14.

The CPU execution time is 9249 ms.

