CSCI-3901-Assignment4

Name: Pratheep Kumar Manoharan

Banner Id: B00837436

Files: MainClass.java, Cell.java, Group.java, Mathdoku.java

Objective:

To learn and work on how to solve the state space exploration problems.

Assumptions:

1. Group description will always have space and the grid will not have space in between or at the front or at the end.
   Example: group description – "a<space>3<space>– "
   Grid – "aabb" – no space
2. Group description can have multiple spaces in between.
3. All operators must have more than one operand so the group names should be available in more than one cell except equal to (=) operator which accepts only one operand.
4. In group description, group result cannot be zero or negative for the valid operators. Even for subtraction we consider maximum value with others.

Algorithm applied:

1. Create a puzzle.
2. Set zero in all the cells.
3. Check if any cell is zero.
4. If yes, then loop all the possible values on the cell.
5. Check the value is already in the row. If the value is already in row, then go to the next looping value.
6. Check the value is already in the column. If the value is already in column, then go to the next looping value.
7. Check the value is suitable for the group. If the value is not suitable, then go to the next looping value.
8. If all conditions passed, then set the value in the cell.
9. At this point, call solve method again recursively.
10. When none of the value is set for a cell, then the method returns false.
11. When method returns false, zero is set as cell value.

12. When false is returned for a cell, the looping of the current cell stops and goes one cell back. Looping of the previous cell will continue till a value is set into a cell. If none of the value is valid fit, then control goes one more cell back till the first cell.
13. When looping ends without a valid fit, then solve method returns false.
14. When we can find at least one valid fit for each cell, then the solve returns true.
15. If the method returns true for all the cells, then puzzle is solved.
16. If no cell is zero, then the puzzle is solved.
17. Whenever we set a value into the puzzle, then choices field is incremented.

Algorithm applied to improve degree of efficiency:

1. Value will not be set into the cell, if it is already in row and column.
2. Value will not be set into the cell, unless it is a fit for the group. Value is checked with group result and operation.
3. Only one looping of the valid values is done and in that only values fit for the group is set into the puzzle.
4. When no value is set for the cell, method return false. Since it is a recursive call, while going one cell back, if the result is again false, then both cells will be set with zero in one go. In this way, if the puzzle is not solvable, we can find it faster.
5. In this approach, finding the first possible solution is faster. We can set a value and proceed till we get a valid value in all cells.
6. For addition operation, value greater than the result will not be considered.
7. For multiplication operation, value which are not factor of result will not be considered.

Technical code flow:

Used the same MainUI.java class file shared in the assignment 2 with slight modifications for this assignment.

Cell Class:

1. Cell object represents a cell in the puzzle.
2. rowNumber and columnNumber are the attributes of the class.
3. Constructor is used to create the object.
4. getRowNumber and getColumnNumber methods are used to get the value of this object.

Group Class:

1. Group object represents a group in the puzzle.
2. Each group will have an operator, result and name. All these are created as the attributes.
3. Setter and getter methods are created for the attributes.

Mathdoku Class:

Attributes:

1. size variable has the size of the puzzle.
2. choices variable has the number of attempts made to solve the puzzle.
3. puzzle list in list holds the values of each cell.
4. grpNameOfCell list in list holds the group name of each cell.
5. groups have the list of all the group objects.
6. grpNames has the list of all the group names.
7. grpCellsMap has the group name as the key and list of cells associated with the group as value.

Methods:

printString method:

1. Gets the string as input.
2. Prints the input message to the user.

clearAllDetails method:

1. This method is used in loadPuzzle method.
2. This method is used to clear all the details from the class attributes while loading newly.

getStringArray method:

1. Gets a line from input stream as input.
2. Returns an array of valid strings.
3. Ignores the empty spaces, tabs etc.,
4. Value in first index is taken as group name, second index is considered as group result and the third index is taken as the group operator.

loadPuzzle method:

1. Get the stream of data as input.
2. After doing null check on the stream, each line is taken for processing.
3. Line is trimmed and proceeded only if it has a valid content.
4. If the line does not have space in between characters, then the line is treated as the grid containing group names.
5. Each character is treated as a separate group name.
6. Group name and cell position is stored in grpCellsMap.
7. grpNameOfCell stores the group name as the grid.
8. If the line has space in between characters, then the line is treated as the group description.
9. Line is passed to getStringArray method and the values are stored into a group object.
10. If the array length is less than three, then the line is ignored.

checkRowColSize method:

1. This method is used to check whether the puzzle is valid n*n puzzle.
2. Compares the number of rows in the puzzle with the number of columns in each row.
3. Returns Boolean value.
4. This method is used in readyToSolve method.

checkGrpListWithGrid method:

1. This method is used to check whether all the group name in the grid is in the group description to solve.
2. Loops all the group name in the grid and check whether the group name is in the group description list.
3. Returns Boolean value.
4. This method is used in the readyToSolve method.

checkGrpResult method:

1. This method is used to check group result provided in the group description is an integer value.
2. In case if it is not an integer value, exception will be thrown.
3. Returns Boolean value.
4. This method is used in the readyToSolve method.

readyToSolve method:

1. This method checks below conditions

    a. Load method is called before this
    b. Puzzle is valid n*n
    c. All group names in grid has group description
    d. Group result from the input is valid
    e. Group operator is valid
    f. Each group has valid number of cells associated with it

2. The conditions are executed in sequence. If one condition is failed, then the method will be returned with false.
3. Method returns Boolean value as output.

setZeroInAllCell method:

1. Sets zero in each cell of the puzzle.

2. This method is used to create zero puzzle before solve or readyToSolve method to do basic validations.

calPosValGrp method:

1. This method is used to compute possible values that can be set for the group.
2. Stores the possible values in the group object.

validPuzzle method:

3. This method is used in readyToSolve method.
4. This method is used to check the number of cells associated with the group.

checkInColumn method:

1. Checks whether the value is already in the column.
2. Returns Boolean value.
3. This method is used in the readyToSolve method.

checkInRow method:

1. Checks whether the value is already in the row.
2. Returns Boolean value.
3. This method is used in the readyToSolve method.

checkGrpOpr method:

1. This method is used to check operators of the group.
2. Returns Boolean value.
3. This method is used in the readyToSolve method.

solve method:

1. This method is used to solve the puzzle.
2. In this method, all possible values are calculated.
3. Returns Boolean value.

trySolve method:

1. This method is used to solve the puzzle.
2. In this method, possible values are looped with row check, column check and group check. Values passing all conditions are set into a cell.
3. Recursive call is done for all cells.
4. Returns Boolean value.

checkInGrp method:

1. This method is used to check whether the value is fit for the group.
2. Based on the operation, the values are computed and checked against the group result.
3. Returns Boolean value.

nullCheck method:

1. Checks whether any cell has zero as value.
2. This is used to check whether the puzzle is solved or not.

print method:

1. Returns the output if the puzzle is solved in the form of grid in string.
2. Returns the group name if the puzzle is not solved in the form of grid in string.

choices method:

1. Returns the number of attempts made to solve the puzzle.

Test Cases:

S.No	Test case	Expected result
	loadPuzzle method	
	input validation	
1	Give null as input to load method. Program should not through a stack trace to users.	False should be returned.
2	Give valid data stream of text file. Program should load all the details.	True should be returned.
3	Give some invalid but data stream of different file format. Program should try to load.	True should be returned.
	readyToSolve method	
4	Provide a grid which is not n*n.	False should be returned.
5	Provide a grid which is n*n.	True should be returned assuming all other details are correct.
6	Give a blank line or lines in between matrix.	True should be returned assuming all other details are correct.
7	Give a blank line or lines in between group desc.	True should be returned assuming all other details are correct.
8	Give blank line at the top.	True should be returned assuming all other details are correct.

9	Give blank line at the bottom.	True should be returned assuming all other details are correct.
10	Give blank line in between matrix and grp desc.	True should be returned assuming all other details are correct.
11	Give one or more spaces/tabs in between group desc.	True should be returned assuming all other details are correct.
12	Give one or more spaces/tabs at front of group desc.	True should be returned assuming all other details are correct.
13	Give one or more spaces/tabs at the end of group desc.	True should be returned assuming all other details are correct.
14	Form grid with group name not given with operator description.	False should be returned.
15	Operator not valid.	False should be returned.
16	Have a group name that is not in grid but with operator.	False should be returned.
17	Give result as negative in input.	False should be returned.
18	Give result as zero in input.	False should be returned.
19	Give +, -, *, / group with only one operand.	False should be returned.
20	Give result as string and check.	False should be returned.
21	Give space in between group names in grid.	False should be returned.
22	Give space in front at first in grid.	True should be returned as it is handled using trim.
23	Give space at the end of group names in grid.	True should be returned as it is handled using trim.
24	Give = in same row duplicate value.	False should be returned.
25	Give = same column duplicate value.	False should be returned.
	solve method	
26	Check unsolvable puzzle.	False should be returned.
27	give solvable puzzle	True should be returned.
28	test group with same alphabet in different cases	
	print method	
29	print unsolvable puzzle after solve	Input group name grid should be returned.
	choices method	
30	call choices unsolvable puzzle after solve	number of attempts made should be returned.
	dataflow	
31	call load, ready, solve, print, choices	Normal data should work fine.

32	call ready	False should be returned.
33	call solve	False should be returned.
34	call print	null returned
35	call choices	0 returned
36	call load, solve and ready	puzzle is valid - true puzzle is invalid - false
37	call load, print, solve and print	first print - group name if puzzle is solved second print - results
38	call load, choices, solve and choices	first choice - 0 second choice - number of attempts

Further study:

1. Can improve the efficiency by finding the possible combination of values that can be set in a group. In this program, only found the possible values and not the combinations.