King Saud University

CCIS- CS Dept.- Spring 2023

CSC361

Assignment#2

Programming Assignment

Team	*ID*	*Section#*	*Work*
Rawan Alotaibi	442201374	44093	Zoom Meeting
Sadeem Albudairi	442200458		
Waad Alahmed	442200402		

Description:

We can solve the tenner grid problems in many ways such as Backtracking and Backtracking with MRV, forward checking, and more.

First: Backtracking

In the beginning, the program assigns a random value from zero to nine so that it fulfills the rules of the game in the first empty place. The search starts from the first row, the first column, then the second row, the second column, until it reaches row n column n. After he reaches the first empty place, he starts checking the available values for the assignment, starting from zero to nine. If he finds a valid value and the rules of the game are met, he returns true, and if he does not find a valid value for the assignment, he returns false, then backtracks to the previous and changes the values down to a solution that fulfills the rules for each game.

First: Backtracking with MRV

Like backtracking, the difference is in the process of choosing the place to start from.

The code first determines whether the filled method is true before it begins and since it starts in the cell with the lowest probability and fills it first, its objective is to find the minimum remaining value.

Third: Forward Checking

The forward checking method is a method that takes four parameters (grid, row

Column, possibilities) grid is 2 dominions array and

possibilities in three dimensions array and return a Boolean value that is true or false at first check if the row is the last row and return false if the sum of the columns does not equal the last row then check if the row was the second row and the column is out of bounds it will return true it will go to for loop witch assigned the possibilities value to the cells then call the method that update domain to its neighbors by using 3 dimension array that we received it as a parameter after that it will call the forward checking method again and increments the column and let it in if statements if it goes to the last cell and returns true that's mean the method find the solution otherwise it will update the neighbors and then unassigned the cell and try another value.

Some important parts of the code:

Backtracking

```
private static void backtrackSolver(int row, int col) {
111     if (row == ROWS_SIZE) {
112        printMain("Backtrack");
113        return;
114     }
115     int [] nextCellCoordinates = getNextCell(row, col);
116     int nextRow = nextCellCoordinates[0];
117     int nextColumn = nextCellCoordinates[1];
118     if (isEmptyCell(row, col)) {
119        for (int selectedValue = 0; selectedValue < COLUMNS_SIZE; selectedValue++) {
120           if (consistencyCheck(row, col, selectedValue)) {
121              grid[row][col] = selectedValue; ckNumOfAss10++;
122              backtrackSolver(nextRow, nextColumn);
123              grid[row][col] = EMPTY_CELL; ckNumOfAss10++;
124           }
125        }
126        return;
127     }
128     backtrackSolver(nextRow, nextColumn);
129  }
```

BT+MRV

```
public static void MRVSolver() {
159
160     if (isFilled()){
161        printMain("MRV");
162        return;
163     }
164     int [] coordinates = getMinimumChoicesCell();
165     int x = coordinates[0];
166     int y = coordinates[1];
167     for (int choice = 0; choice < COLUMNS_SIZE; choice++ ){
168        if (consistencyCheck(x, y, choice)) {
169           grid[x][y] = choice; ckNumOfAss11++;
170           MRVSolver();
171           grid[x][y] = -1; ckNumOfAss11++;
172        }
173     }
174  }
```

Forward

```
 private static void ForwardCheckingSolver() {
184     int []coordinates = getFirstEmptyCell();
185     int x = coordinates[0];
186     int y = coordinates[1];
187     if (x == -1 && y == -1) {
188        printMain("Forward Checking");
189        return;
190     }
191     for (int choice = 0; choice < COLUMNS_SIZE; choice++ ){
192        if (consistencyCheck(x, y, choice)) {
193           grid[x][y] = choice; ckNumOfAss12++;
194           ForwardCheckingSolver();
195           grid[x][y] = -1; ckNumOfAss12++;
196        }
197     }
198  }
```

Sample run:

Initial state:

```
199 //Run1: Example#1
200 static int [][]sample1 = { { -1, 6, 2, 0, -1, -1, -1, 8, 5, 7 },
201                          { -1,0, 1, 7, 8, -1, -1, -1, 9, -1 },
202                          { -1, 4, -1, -1, 2, -1, 3, 7, -1, 8 },
203                          { 13, 10, 8, 7, 19, 16, 11, 19, 15, 17 },};
204
205 //Run2: Example#2
206 static int [][]sample2 = {  { -1, -1, -1, 8, 5, 0, 9, -1, 6, 1 },
207                           { 1, 9, -1, -1, -1, 8, 3, 5, -1, 2 },
208                           { -1, 2, -1, 5, 4, -1, -1, -1, 8, -1 },
209                           { 10, 13, 13, 13, 16, 8, 19,13,18,12}};
210
211 //Run3: Example#3
212 static int [][]sample3 = { {-1,-1,4,-1,-1,-1,5,-1,-1,9},
213                          {-1,8,-1,-1,-1,4,-1,9,-1,-1},
214                          {-1,4,2,0,8,-1,-1,5,3,-1},
215                          {17,15,7,9,13,11,15,21,11,16}};
216
217 //Run4: Example#4
218 static int [][]sample4 = { {-1,-1,-1,-1,-1,-1,-1,6,-1,4},
219                          {-1,-1,0,7,-1,-1,-1,-1,-1,-1},
220                          {16,6,3,15,15,4,7,14,4,6}   };
221
222 //Run5: Example#5
223 static int [][]sample5 = { {2,3,6,8,-1,-1,9,1,-1,-1},
224                          {4,7,-1,-1,9,-1,-1,-1,-1,-1},
225                          {6,0,8,3,-1,7,-1,-1,-1,9},
226                          {8,3,1,9,6,5,-1,-1,-1,2},
227                          {20,13,17,25,19,17,15,8,22,24}};
```

Initial state	Final state (Solution)		
	Backtracking	**BT+MRV**	**Forward**
Run1:	S Solution Found = = = = = = = = = = = = 4 6 2 0 9 1 3 8 5 7 3 0 1 7 8 6 5 4 9 2 6 4 5 0 2 9 3 7 1 8 ---------- 13 10 8 7 19 16 11 19 15 17 = = = = = = = = = = = Backtrack Consistency Checks Count = 3112 Backtrack Took = 1070800(ns) Backtrack Number of assignments = 622	Solution Found = = = = = = = = = = = = 4 6 2 0 9 1 3 8 5 7 3 0 1 7 8 6 5 4 9 2 6 4 5 0 2 9 3 7 1 8 ---------- 13 10 8 7 19 16 11 19 15 17 = = = = = = = = = = = MRV Consistency Checks Count = 7032 MRV Took = 299000(ns) MRV Number of assignments = 32	Solution Found = = = = = = = = = = = = 4 6 2 0 9 1 3 8 5 7 3 0 1 7 8 6 5 4 9 2 6 4 5 0 2 9 3 7 1 8 ---------- 13 10 8 7 19 16 11 19 15 17 = = = = = = = = = = = Forward Checking Consistency Checks Count = 10212 Forward Checking Took = 597800(ns) Forward Checking Number of assignments = 622
Run2:	Solution Found = = = = = = = = = = = = 3 2 4 8 5 0 9 7 6 1 1 9 6 0 7 8 3 5 4 2 6 2 3 5 4 0 7 1 8 9 ---------- 10 13 13 13 16 8 19 13 18 12 = = = = = = = = = = = Backtrack Consistency Checks Count = 213 Backtrack Took = 59900(ns) Backtrack Number of assignments = 248	Solution Found = = = = = = = = = = = = 3 2 4 8 5 0 9 7 6 1 1 9 6 0 7 8 3 5 4 2 6 2 3 5 4 0 7 1 8 9 ---------- 10 13 13 13 16 8 19 13 18 12 = = = = = = = = = = = MRV Consistency Checks Count = 2153 MRV Took = 172900(ns) MRV Number of assignments = 54	Solution Found = = = = = = = = = = = = 3 2 4 8 5 0 9 7 6 1 1 9 6 0 7 8 3 5 4 2 6 2 3 5 4 0 7 1 8 9 ---------- 10 13 13 13 16 8 19 13 18 12 = = = = = = = = = = = Forward Checking Consistency Checks Count = 2803 Forward Checking Took = 67600(ns) Forward Checking Number of assignments = 248
Run3:	Solution Found = = = = = = = = = = = = 6 3 4 2 0 1 5 7 8 9 2 8 1 7 5 4 3 9 0 6 9 4 2 0 8 6 7 5 3 1 ---------- 17 15 7 9 13 11 15 21 11 16 = = = = = = = = = = = Backtrack Consistency Checks Count = 84562 Backtrack Took = 4765300(ns) Backtrack Number of assignments = 17324	Solution Found = = = = = = = = = = = = 6 3 4 2 0 1 5 7 8 9 2 8 1 7 5 4 3 9 0 6 9 4 2 0 8 6 7 5 3 1 ---------- 17 15 7 9 13 11 15 21 11 16 = = = = = = = = = = = MRV Consistency Checks Count = 199132 MRV Took = 5933300(ns) MRV Number of assignments = 1758	Solution Found = = = = = = = = = = = = 6 3 4 2 0 1 5 7 8 9 2 8 1 7 5 4 3 9 0 6 9 4 2 0 8 6 7 5 3 1 ---------- 17 15 7 9 13 11 15 21 11 16 = = = = = = = = = = = Forward Checking Consistency Checks Count = 321382 Forward Checking Took = 4155900(ns) Forward Checking Number of assignments = 17324
Run4:	Solution Found = = = = = = = = = = = = 7 5 3 8 9 0 2 6 1 4 9 1 0 7 6 4 5 8 3 2 ---------- 16 6 3 15 15 4 7 14 4 6 = = = = = = = = = = = Backtrack Consistency Checks Count = 75039 Backtrack Took = 2270600(ns) Backtrack Number of assignments = 40396	Solution Found = = = = = = = = = = = = 7 5 3 8 9 0 2 6 1 4 9 1 0 7 6 4 5 8 3 2 ---------- 16 6 3 15 15 4 7 14 4 6 = = = = = = = = = = = MRV Consistency Checks Count = 142419 MRV Took = 749000(ns) MRV Number of assignments = 2982	Solution Found = = = = = = = = = = = = 7 5 3 8 9 0 2 6 1 4 9 1 0 7 6 4 5 8 3 2 ---------- 16 6 3 15 15 4 7 14 4 6 = = = = = = = = = = = Forward Checking Consistency Checks Count = 217529 Forward Checking Took = 3073500(ns) Forward Checking Number of assignments = 40396
Run5:	Solution Found = = = = = = = = = = = = 2 3 6 8 0 4 9 1 7 5 4 7 2 5 9 1 0 6 3 8 6 0 8 3 4 7 2 1 5 9 8 3 1 9 6 5 4 0 7 2 ---------- 20 13 17 25 19 17 15 8 22 24 = = = = = = = = = = = Backtrack Consistency Checks Count = 1272 Backtrack Took = 100100(ns) Backtrack Number of assignments = 51198	Solution Found = = = = = = = = = = = = 2 3 6 8 0 4 9 1 7 5 4 7 2 5 9 1 0 6 3 8 6 0 8 3 4 7 2 1 5 9 8 3 1 9 6 5 4 0 7 2 ---------- 20 13 17 25 19 17 15 8 22 24 = = = = = = = = = = = MRV Consistency Checks Count = 14322 MRV Took = 436300(ns) MRV Number of assignments = 3106	Solution Found = = = = = = = = = = = = 2 3 6 8 0 4 9 1 7 5 4 7 2 5 9 1 0 6 3 8 6 0 8 3 4 7 2 1 5 9 8 3 1 9 6 5 4 0 7 2 ---------- 20 13 17 25 19 17 15 8 22 24 = = = = = = = = = = = Forward Checking Consistency Checks Count = 16812 Forward Checking Took = 64400(ns) Forward Checking Number of assignments = 51198

Some Results:

	Backtracking	BT+MRV	Forward
Final CSP Tenner variable assignments: Number of variable assignments			
For each Run	Run1: 622 Run2: 248 Run3: 17324 Run4: 40396 Run5: 511198	Run1: 32 Run2: 54 Run3: 1753 Run4: 2984 Run5: 3106	Run1: 622 Run2: 248 Run3: 17324 Run4: 40396 Run5: 511198
Average Sum Run1 to Run5 then divide it by 5	**113957.6**	**1585.8**	**113957.6**
Final CSP Tenner variable assignments: Number of Consistency check			
For each Run	Run1: 3112 Run2: 213 Run3: 84562 Run4: 75039 Run5: 1272	Run1: 7032 Run2: 2153 Run3: 199132 Run4: 142419 Run5: 14322	Run1: 10212 Run2: 2803 Run3: 32182 Run4: 217529 Run5: 16812
Average Sum Run1 to Run5 then divide it by 5	**32839.6**	**73011.6**	**74289.4**
Time used to solve each problem			
For each Run	Run1: 1070800 Run2: 59900 Run3: 4765300 Run4: 2270600 Run5: 100100	Run1: 299000 Run2: 172900 Run3: 593300 Run4: 749000 Run5: 436300	Run1: 597800 Run2: 67600 Run3: 4155900 Run4: 3075500 Run5: 64400
Average Sum Run1 to Run5 then divide it by 5	**1653340**	**396100**	**1592240**

Analysis of the results:

We can see that the backtracking with MRV Method assigns the least number of values compared to the rest methods, and consume significantly less time than the rest, but in terms of check of consistency, the backtracking performs the least number of checks compared to the rest methods. On the one hand, reaching the solution in all methods is able to reach the solution or the required result.

Therefore, backtracking and backtracking with MRV are more distinguished than forward.