King Saud University

College of Computer and Information Sciences

Department of Computer Science

CSC361 Programming Project

Second Semester 2024

Instructor: Nouf Alshenaifi

STUDENT NAME	STUDENT ID	Work distribution
Lujain Al Suleiman	443200947	Implementing BackwardChecking & TennerGrid class, Code description
Aliyah Aljarallah	443201214	Implementing ForwardChecking class
Shoug Alsaleem	443200641	Implementing ForwardCheckingMRV class

Table of Contents

Code Description:

We have written an AI program that solves 10 by 3 Tenner Grid puzzle as a Constraint satisfaction problem. We will be using three approaches, simple backtracking, forward checking, and forward with MRV heuristic. our code is written in Java, and it contains four classes, here is a brief description of each class:

TennerGrid class

It contains methods for filling grids, and printing grids.

ROWS and COLS are constants representing the number of rows and columns in the grid.

grid: a 2D array representing the Tenner grid.

used: a 2D Boolean array to keep track of used numbers in each row (for the row constraint).

SimpleBacktraking class

It extends the TennerGrid class, which presumably contains the grid representation and basic operations for the Tenner grid puzzle.

BTconsistency: Tracks the number of times the consistency check is performed.

BTassignments: Tracks the number of variable assignments made during the solving process.

finalAssignments: An array to store the final assignments for each variable in the grid.

counter: A counter to keep track of the number of final assignments made.

The simpleBacktrack is a recursive algorithm used to solve the Tenner grid puzzle. It iterates through each cell in the grid, trying different values from 0 to 9 for empty cells. The method checks if a value can be placed in a cell without violating constraints such as no repeated numbers in rows and columns, and certain adjacent cells have different values. If a valid value is found, it is placed in the cell, and the method moves on to the next cell. If no valid value is found, it backtracks to the previous cell and tries a different value. This process continues until the grid is fully filled or no valid solution exists.

The class also includes methods to check if a number has been used in a row (`isUsed`), if a column sum is valid (`isValid`), and if a number is adjacent to another in the grid (`isAdjacent`). Additionally, there is a method (`setDomain`) to pre-populate the `used` array based on the initial values in the grid.

ForwardChecking class

ForwardChecking class extends the TennerGrid class, which presumably contains the grid representation and basic operations for the Tenner grid puzzle.

FCconsistency: Tracks the number of times the consistency check is performed.

FCassignments: Tracks the number of variable assignments made during the solving process.

finalAssignments: An array to store the final assignments for each variable in the grid.

counter: A counter to keep track of the number of final assignments made.

forwardCheck Method Recursively explores possible assignments for each cell in the grid using forward checking, pruning branches that lead to invalid assignments.

It checks if a cell is unassigned and then iterates over possible values (0-9) that can be assigned to the cell.

For each valid assignment, it updates the grid, final assignments array, and possibilities array, and then recursively calls itself for the next cell.

If no valid assignment is found, it backtracks and tries a different value for the previous cell.

isSafe Method Checks if it's safe to place a number num at a position in the grid by ensuring it doesn't violate row constraints or exceed column sums.

updatePossibilities Method Updates the possibilities array based on the assigned value, setting it to false for the assigned value in the same row and the row above.

ForwardCheckingMRV class

The ForwardCheckingMRV class extends TennerGrid and implements a solution strategy for the Tenner grid puzzle using forward checking with the Minimum Remaining Values (MRV) heuristic. This heuristic selects the variable (grid cell) with the fewest remaining values in its domain, aiming to reduce the branching factor and improve efficiency.

consistency: Tracks the number of consistency checks performed.

assignments: Tracks the number of variable assignments made during the solving process.

finalAssignments: An array to store the final assignments for each variable.

counter: A counter to keep track of the current position in the finalAssignments array.

inDomain Method:

Checks if placing a number `num` at a given position `(row, col)` is safe based on the puzzle constraints. It considers row, column, and box constraints.

isSafe Method:

Similar to inDomain but also considers the target sum constraint for the column.

updatePossibilities Method:

Updates the possibilities array based on the current assignment. It marks `false` for any number that conflicts with the current assignment.

forwardCheckMVR Method:

Recursive method that performs forward checking with the MRV heuristic.

It selects the next variable (cell) to assign based on the MRV heuristic.

For each valid assignment, it updates the grid, increments the assignment counter, and calls itself recursively for the next variable.

If a solution is found, it returns `true`; otherwise, it backtracks and tries a different assignment.

findNextEmptyCellWithMRV Method:

Finds the next empty cell with the minimum remaining values (MRV) in its domain.

This method is crucial for the MRV heuristic in selecting the most constrained variable.

countRemainingValues Method:

Counts the remaining values in the domain of a given cell.

Used in conjunction with the MRV heuristic to determine the variable with the fewest remaining values.

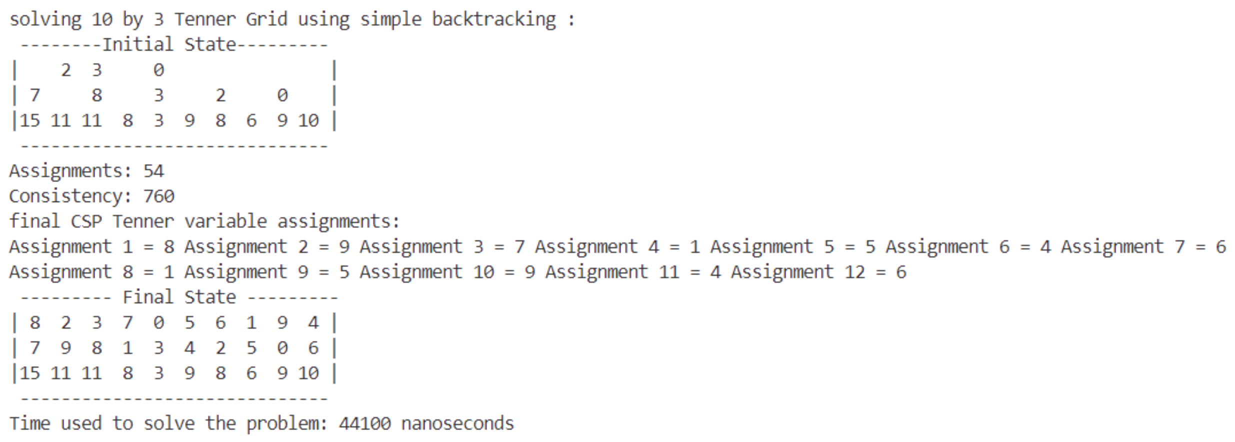
isValid Method:

Checks if the current assignment in a column satisfies the target sum constraint.

Sample Run:

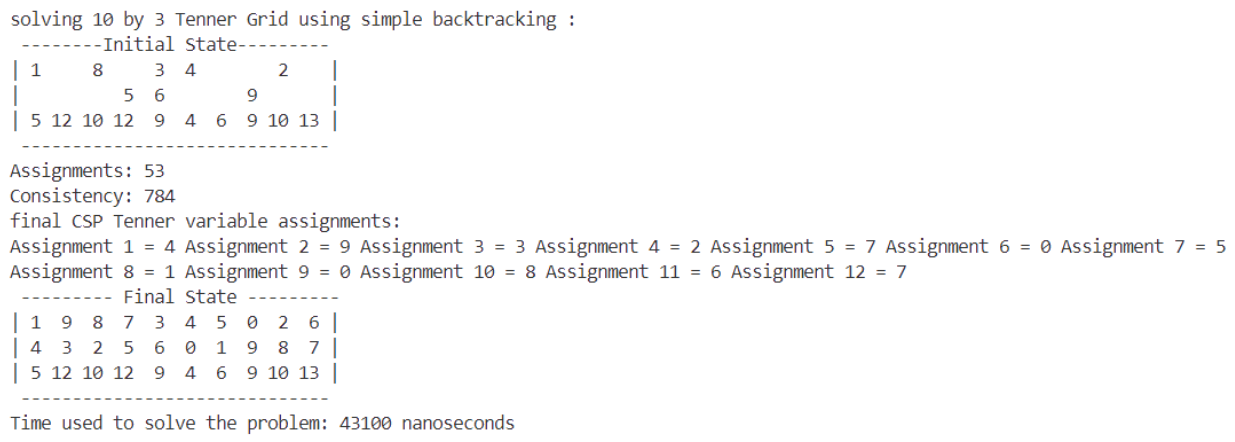
sample run for simple backtracking:

Grid 1:



```
solving 10 by 3 Tenner Grid using simple backtracking :
 --------Initial State---------
|    2  3     0                |
| 7     8     3     2     0    |
|15 11 11  8  3  9  8  6  9 10 |
 ------------------------------
Assignments: 54
Consistency: 760
final CSP Tenner variable assignments:
Assignment 1 = 8 Assignment 2 = 9 Assignment 3 = 7 Assignment 4 = 1 Assignment 5 = 5 Assignment 6 = 4 Assignment 7 = 6
Assignment 8 = 1 Assignment 9 = 5 Assignment 10 = 9 Assignment 11 = 4 Assignment 12 = 6
 --------- Final State ---------
| 8  2  3  7  0  5  6  1  9  4 |
| 7  9  8  1  3  4  2  5  0  6 |
|15 11 11  8  3  9  8  6  9 10 |
 ------------------------------
Time used to solve the problem: 44100 nanoseconds
```

Grid 2:



```
solving 10 by 3 Tenner Grid using simple backtracking :
 --------Initial State---------
| 1     8     3  4        2    |
|          5  6        9       |
| 5 12 10 12  9  4  6  9 10 13 |
 ------------------------------
Assignments: 53
Consistency: 784
final CSP Tenner variable assignments:
Assignment 1 = 4 Assignment 2 = 9 Assignment 3 = 3 Assignment 4 = 2 Assignment 5 = 7 Assignment 6 = 0 Assignment 7 = 5
Assignment 8 = 1 Assignment 9 = 0 Assignment 10 = 8 Assignment 11 = 6 Assignment 12 = 7
 --------- Final State ---------
| 1  9  8  7  3  4  5  0  2  6 |
| 4  3  2  5  6  0  1  9  8  7 |
| 5 12 10 12  9  4  6  9 10 13 |
 ------------------------------
Time used to solve the problem: 43100 nanoseconds
```

Grid 3:

```
solving 10 by 3 Tenner Grid using simple backtracking :
 --------Initial State---------
| 7                 2          |
| 6  4  0     7  8        2       |
|13  7  8  9 16  8  7 10  6  6 |
 -----------------------------
Assignments: 46
Consistency: 623
final CSP Tenner variable assignments:
Assignment 1 = 3 Assignment 2 = 8 Assignment 3 = 6 Assignment 4 = 3 Assignment 5 = 9 Assignment 6 = 0 Assignment 7 = 5
Assignment 8 = 1 Assignment 9 = 9 Assignment 10 = 4 Assignment 11 = 5 Assignment 12 = 1
 --------- Final State ---------
| 7  3  8  6  9  0  2  1  4  5 |
| 6  4  0  3  7  8  5  9  2  1 |
|13  7  8  9 16  8  7 10  6  6 |
 -----------------------------
Time used to solve the problem: 43700 nanoseconds
```

Grid 4:

```
solving 10 by 3 Tenner Grid using simple backtracking :
 --------Initial State---------
| 4  2  9     1                 8 |
|             4        1     6 |
| 9  9  9  9  5  3 14  4 14 14 |
 -----------------------------
Assignments: 48
Consistency: 701
final CSP Tenner variable assignments:
Assignment 1 = 5 Assignment 2 = 7 Assignment 3 = 0 Assignment 4 = 7 Assignment 5 = 2 Assignment 6 = 0 Assignment 7 = 3
Assignment 8 = 6 Assignment 9 = 8 Assignment 10 = 3 Assignment 11 = 5 Assignment 12 = 9
 --------- Final State ---------
| 4  2  9  7  1  0  6  3  5  8 |
| 5  7  0  2  4  3  8  1  9  6 |
| 9  9  9  9  5  3 14  4 14 14 |
 -----------------------------
Time used to solve the problem: 68900 nanoseconds
```

Grid 5:

```
solving 10 by 3 Tenner Grid using simple backtracking :
 --------Initial State---------
|    6  2  0           8  5  7 |
|    0  1  7  8           9    |
| 7  6  3  7 17  7  8 12 14  9 |
 -----------------------------
Assignments: 48
Consistency: 648
final CSP Tenner variable assignments:
Assignment 1 = 4 Assignment 2 = 3 Assignment 3 = 9 Assignment 4 = 1 Assignment 5 = 6 Assignment 6 = 3 Assignment 7 = 5
Assignment 8 = 4 Assignment 9 = 2
 --------- Final State ---------
| 4  6  2  0  9  1  3  8  5  7 |
| 3  0  1  7  8  6  5  4  9  2 |
| 7  6  3  7 17  7  8 12 14  9 |
 -----------------------------
Time used to solve the problem: 176700 nanoseconds
```

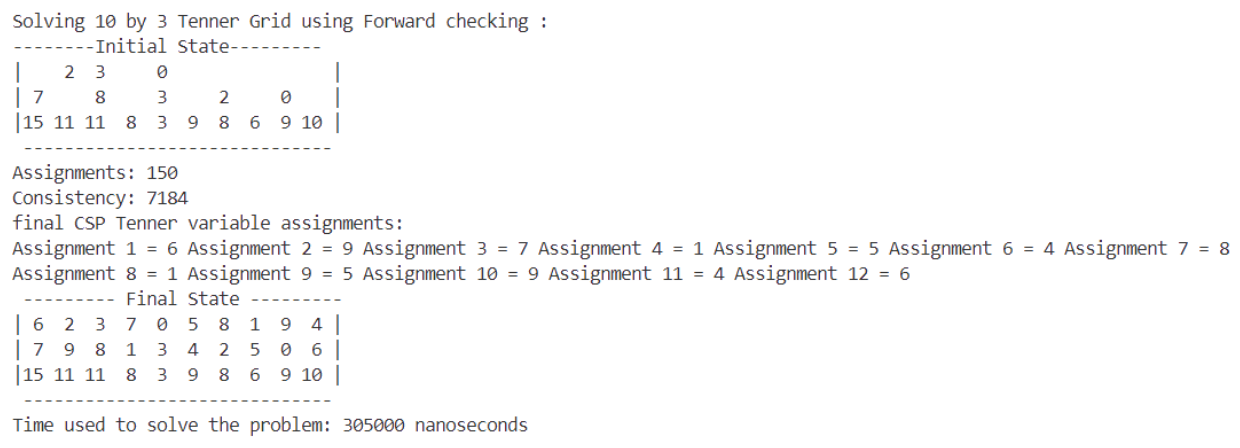
Calculating median for simple backtracking

Sorting the numbers in ascending order:

623, 648, 760, 784 the median is = 760

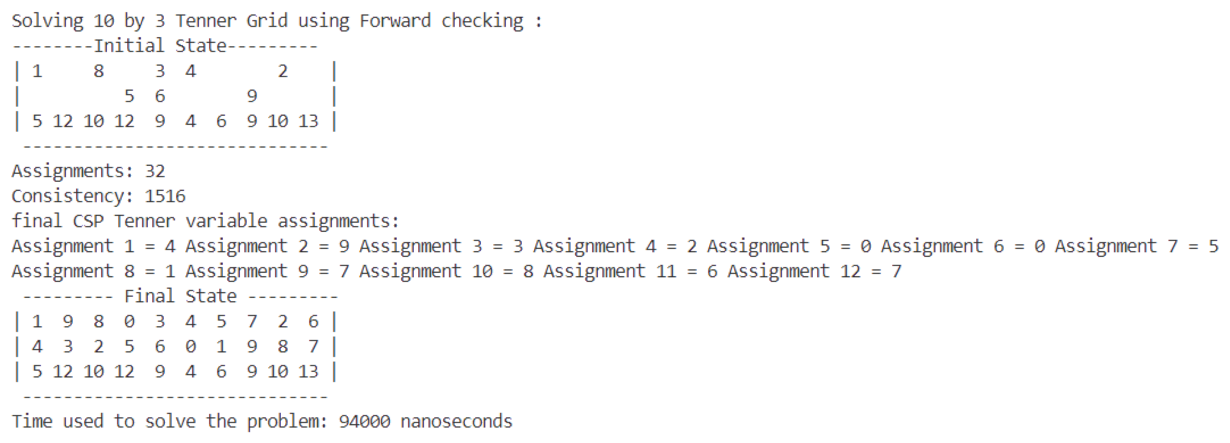
Sample run for Forward Checking:

Grid 1:



```
Solving 10 by 3 Tenner Grid using Forward checking :
--------Initial State---------
|    2  3     0                |
| 7     8     3     2     0    |
|15 11 11  8  3  9  8  6  9 10 |
 -----------------------------
Assignments: 150
Consistency: 7184
final CSP Tenner variable assignments:
Assignment 1 = 6 Assignment 2 = 9 Assignment 3 = 7 Assignment 4 = 1 Assignment 5 = 5 Assignment 6 = 4 Assignment 7 = 8
Assignment 8 = 1 Assignment 9 = 5 Assignment 10 = 9 Assignment 11 = 4 Assignment 12 = 6
 --------- Final State ---------
| 6  2  3  7  0  5  8  1  9  4 |
| 7  9  8  1  3  4  2  5  0  6 |
|15 11 11  8  3  9  8  6  9 10 |
 -----------------------------
Time used to solve the problem: 305000 nanoseconds
```

Grid 2:



```
Solving 10 by 3 Tenner Grid using Forward checking :
--------Initial State---------
| 1     8     3  4        2    |
|           5  6        9      |
| 5 12 10 12  9  4  6  9 10 13 |
 -----------------------------
Assignments: 32
Consistency: 1516
final CSP Tenner variable assignments:
Assignment 1 = 4 Assignment 2 = 9 Assignment 3 = 3 Assignment 4 = 2 Assignment 5 = 0 Assignment 6 = 0 Assignment 7 = 5
Assignment 8 = 1 Assignment 9 = 7 Assignment 10 = 8 Assignment 11 = 6 Assignment 12 = 7
 --------- Final State ---------
| 1  9  8  0  3  4  5  7  2  6 |
| 4  3  2  5  6  0  1  9  8  7 |
| 5 12 10 12  9  4  6  9 10 13 |
 -----------------------------
Time used to solve the problem: 94000 nanoseconds
```

Grid 3:

```
Solving 10 by 3 Tenner Grid using Forward checking :
--------Initial State---------
| 7                 2          |
| 6  4  0     7  8        2    |
|13  7  8  9 16  8  7 10  6  6 |
 -----------------------------
Assignments: 1154
Consistency: 54913
final CSP Tenner variable assignments:
Assignment 1 = 3 Assignment 2 = 8 Assignment 3 = 6 Assignment 4 = 3 Assignment 5 = 0 Assignment 6 = 9 Assignment 7 = 5
Assignment 8 = 1 Assignment 9 = 9 Assignment 10 = 4 Assignment 11 = 5 Assignment 12 = 1
 --------- Final State ---------
| 7  3  8  6  0  9  2  1  4  5 |
| 6  4  0  3  7  8  5  9  2  1 |
|13  7  8  9 16  8  7 10  6  6 |
 -----------------------------
Time used to solve the problem: 1220500 nanoseconds
```

Grid 4:

```
Solving 10 by 3 Tenner Grid using Forward checking :
--------Initial State---------
| 4  2  9     1              8 |
|             4        1     6 |
| 9  9  9  9  5  3 14  4 14 14 |
 -----------------------------
Assignments: 25
Consistency: 1002
final CSP Tenner variable assignments:
Assignment 1 = 5 Assignment 2 = 7 Assignment 3 = 0 Assignment 4 = 7 Assignment 5 = 2 Assignment 6 = 0 Assignment 7 = 3
Assignment 8 = 6 Assignment 9 = 8 Assignment 10 = 3 Assignment 11 = 5 Assignment 12 = 9
 --------- Final State ---------
| 4  2  9  7  1  0  6  3  5  8 |
| 5  7  0  2  4  3  8  1  9  6 |
| 9  9  9  9  5  3 14  4 14 14 |
 -----------------------------
Time used to solve the problem: 73400 nanoseconds
```

Grid 5:

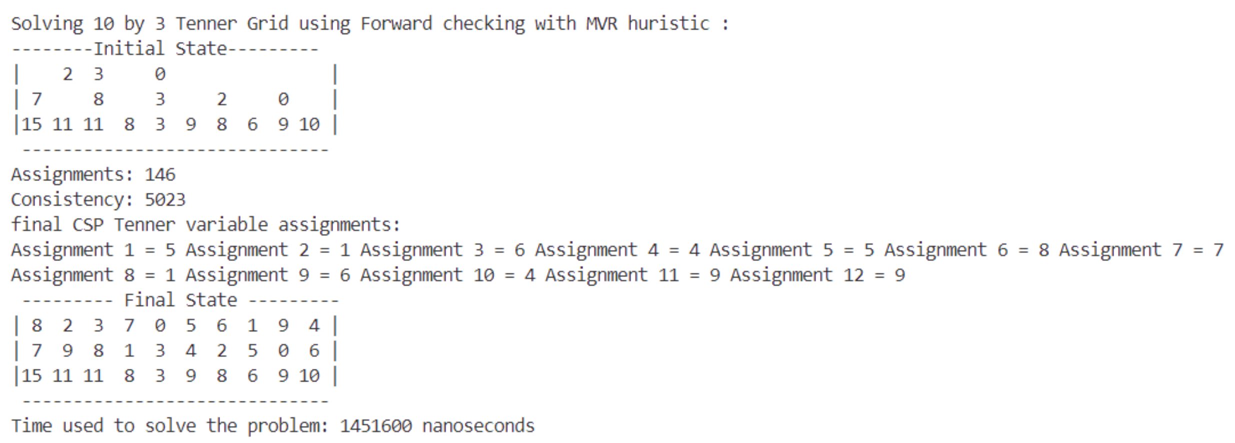
```
Solving 10 by 3 Tenner Grid using Forward checking :
--------Initial State---------
|    6  2  0           8  5  7 |
|    0  1  7  8           9    |
| 7  6  3  7 17  7  8 12 14  9 |
 -----------------------------
Assignments: 39
Consistency: 1929
final CSP Tenner variable assignments:
Assignment 1 = 4 Assignment 2 = 3 Assignment 3 = 9 Assignment 4 = 1 Assignment 5 = 6 Assignment 6 = 3 Assignment 7 = 5
Assignment 8 = 4 Assignment 9 = 2
 --------- Final State ---------
| 4  6  2  0  9  1  3  8  5  7 |
| 3  0  1  7  8  6  5  4  9  2 |
| 7  6  3  7 17  7  8 12 14  9 |
 -----------------------------
Time used to solve the problem: 195100 nanoseconds
```

Calculating median for forward checking

1002, 1516, 1929, 7184, 54913 the median is = 1929

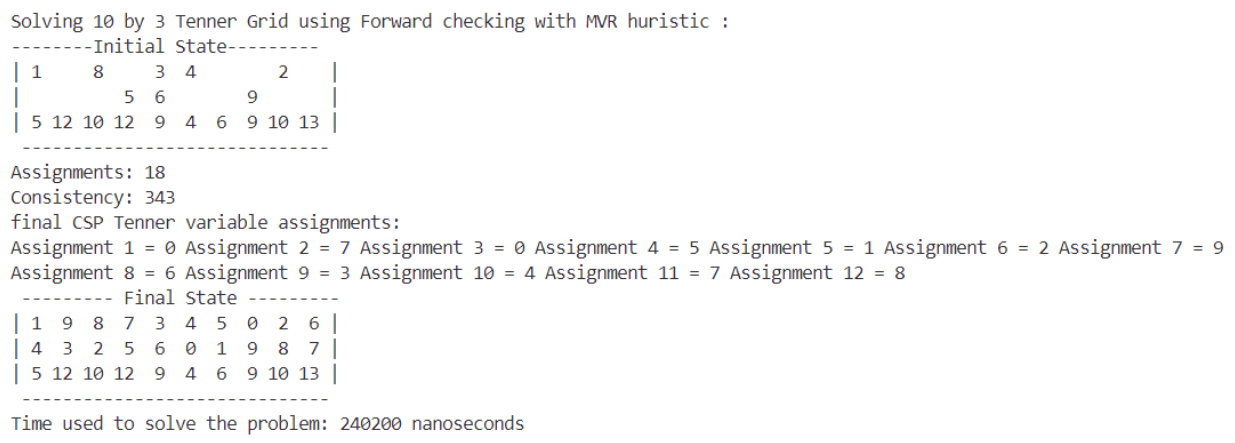
Sample run for Forward Checking with MRV:

Grid 1:



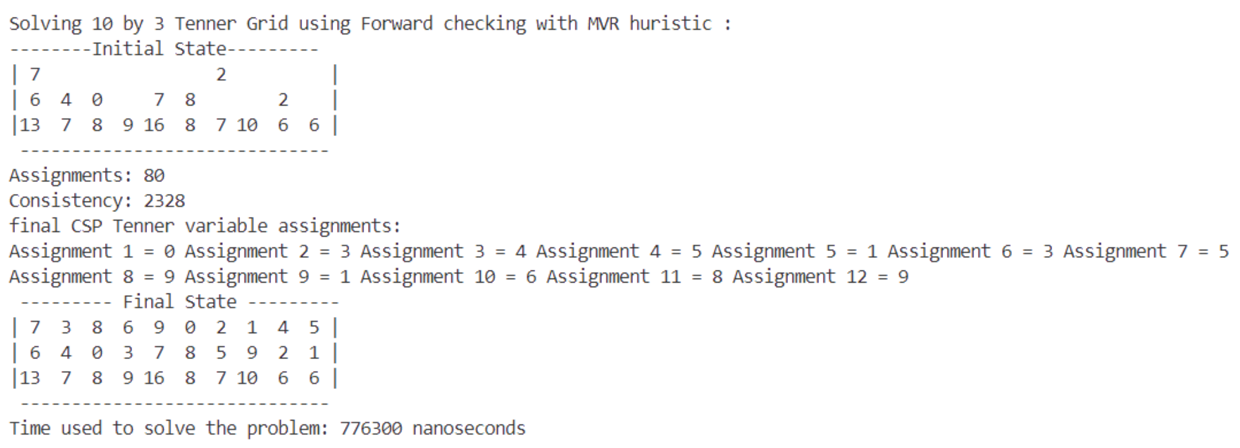
```
Solving 10 by 3 Tenner Grid using Forward checking with MVR huristic :
--------Initial State---------
|    2  3     0                |
| 7     8     3     2     0    |
|15 11 11  8  3  9  8  6  9 10 |
 -----------------------------
Assignments: 146
Consistency: 5023
final CSP Tenner variable assignments:
Assignment 1 = 5 Assignment 2 = 1 Assignment 3 = 6 Assignment 4 = 4 Assignment 5 = 5 Assignment 6 = 8 Assignment 7 = 7
Assignment 8 = 1 Assignment 9 = 6 Assignment 10 = 4 Assignment 11 = 9 Assignment 12 = 9
 --------- Final State ---------
| 8  2  3  7  0  5  6  1  9  4 |
| 7  9  8  1  3  4  2  5  0  6 |
|15 11 11  8  3  9  8  6  9 10 |
 -----------------------------
Time used to solve the problem: 1451600 nanoseconds
```

Grid 2:



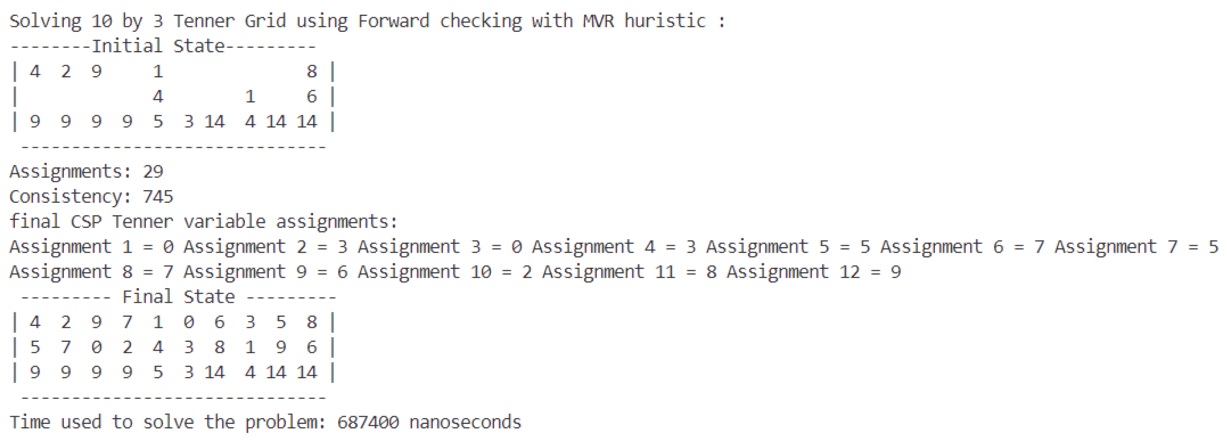
```
Solving 10 by 3 Tenner Grid using Forward checking with MVR huristic :
--------Initial State---------
| 1     8     3  4        2    |
|          5  6        9       |
| 5 12 10 12  9  4  6  9 10 13 |
 -----------------------------
Assignments: 18
Consistency: 343
final CSP Tenner variable assignments:
Assignment 1 = 0 Assignment 2 = 7 Assignment 3 = 0 Assignment 4 = 5 Assignment 5 = 1 Assignment 6 = 2 Assignment 7 = 9
Assignment 8 = 6 Assignment 9 = 3 Assignment 10 = 4 Assignment 11 = 7 Assignment 12 = 8
 --------- Final State ---------
| 1  9  8  7  3  4  5  0  2  6 |
| 4  3  2  5  6  0  1  9  8  7 |
| 5 12 10 12  9  4  6  9 10 13 |
 -----------------------------
Time used to solve the problem: 240200 nanoseconds
```

Grid 3:



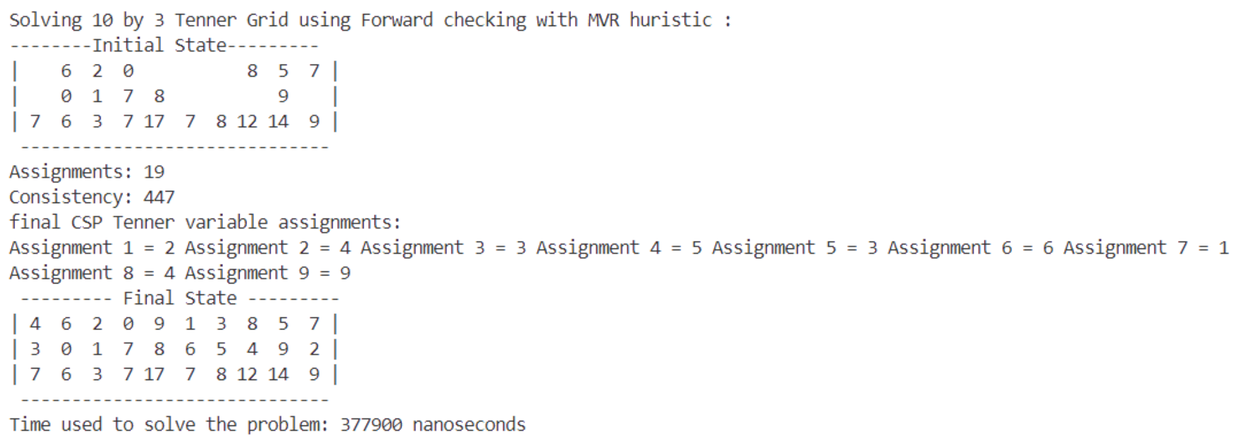
```
Solving 10 by 3 Tenner Grid using Forward checking with MVR huristic :
--------Initial State---------
| 7                 2          |
| 6  4  0     7  8        2    |
|13  7  8  9 16  8  7 10  6  6 |
 ------------------------------
Assignments: 80
Consistency: 2328
final CSP Tenner variable assignments:
Assignment 1 = 0 Assignment 2 = 3 Assignment 3 = 4 Assignment 4 = 5 Assignment 5 = 1 Assignment 6 = 3 Assignment 7 = 5
Assignment 8 = 9 Assignment 9 = 1 Assignment 10 = 6 Assignment 11 = 8 Assignment 12 = 9
 --------- Final State ---------
| 7  3  8  6  9  0  2  1  4  5 |
| 6  4  0  3  7  8  5  9  2  1 |
|13  7  8  9 16  8  7 10  6  6 |
 ------------------------------
Time used to solve the problem: 776300 nanoseconds
```

Grid 4:



```
Solving 10 by 3 Tenner Grid using Forward checking with MVR huristic :
--------Initial State---------
| 4  2  9     1              8 |
|             4        1     6 |
| 9  9  9  9  5  3 14  4 14 14 |
 ------------------------------
Assignments: 29
Consistency: 745
final CSP Tenner variable assignments:
Assignment 1 = 0 Assignment 2 = 3 Assignment 3 = 0 Assignment 4 = 3 Assignment 5 = 5 Assignment 6 = 7 Assignment 7 = 5
Assignment 8 = 7 Assignment 9 = 6 Assignment 10 = 2 Assignment 11 = 8 Assignment 12 = 9
 --------- Final State ---------
| 4  2  9  7  1  0  6  3  5  8 |
| 5  7  0  2  4  3  8  1  9  6 |
| 9  9  9  9  5  3 14  4 14 14 |
 ------------------------------
Time used to solve the problem: 687400 nanoseconds
```

Grid 5:



```
Solving 10 by 3 Tenner Grid using Forward checking with MVR huristic :
--------Initial State---------
|    6  2  0           8  5  7 |
|    0  1  7  8           9    |
| 7  6  3  7 17  7  8 12 14  9 |
 ------------------------------
Assignments: 19
Consistency: 447
final CSP Tenner variable assignments:
Assignment 1 = 2 Assignment 2 = 4 Assignment 3 = 3 Assignment 4 = 5 Assignment 5 = 3 Assignment 6 = 6 Assignment 7 = 1
Assignment 8 = 4 Assignment 9 = 9
 --------- Final State ---------
| 4  6  2  0  9  1  3  8  5  7 |
| 3  0  1  7  8  6  5  4  9  2 |
| 7  6  3  7 17  7  8 12 14  9 |
 ------------------------------
Time used to solve the problem: 377900 nanoseconds
```

Calculating median for forward checking with MRV

Sorting the numbers in ascending order:

343, 447, 745, 2328, 5023, the median is = 745

Conclusion

After running each algorithm five times and calculating the median for each, it appears that the Forward Checking with MRV heuristic algorithm is the most efficient in terms of number of consistency checks, followed by simple backtracking, and the least efficient in terms of number of consistency checks appears to be forward checking.