**CSE537 Artificial Intelligence**

**Project 3 – Sudoku Solver**

**Team Details**

**Name SBU ID Net ID**

Aman Raj 109939872 amraj

Dhanendra Jain 110369635 dhjain

**Sample Test Results-**

**INPUT #1:**

**12,3,4;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

**-,-,-,-,-,-,-,-,-,-,-,-;**

Backtracking:

Execution Time: 0.179972501801

Consistency Checks: 7425

Solution: [[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], [5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4], [9, 10, 11, 12, 1, 2, 3, 4, 5, 6, 7, 8], [2, 1, 4, 3, 6, 5, 8, 7, 10, 9, 12, 11], [6, 5, 8, 7, 10, 9, 12, 11, 2, 1, 4, 3], [10, 9, 12, 11, 2, 1, 4, 3, 6, 5, 8, 7], [3, 4, 1, 2, 7, 8, 5, 6, 11, 12, 9, 10], [7, 8, 5, 6, 11, 12, 9, 10, 3, 4, 1, 2], [11, 12, 9, 10, 3, 4, 1, 2, 7, 8, 5, 6], [4, 3, 2, 1, 8, 7, 6, 5, 12, 11, 10, 9], [8, 7, 6, 5, 12, 11, 10, 9, 4, 3, 2, 1], [12, 11, 10, 9, 4, 3, 2, 1, 8, 7, 6, 5]]

backtrackingMRV:

Execution Time: 0.985359668311

Consistency Checks: 144

Solution: [[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], [5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4], [9, 10, 11, 12, 1, 2, 3, 4, 5, 6, 7, 8], [2, 7, 1, 3, 10, 9, 4, 5, 11, 12, 8, 6], [10, 11, 12, 5, 7, 8, 2, 6, 3, 1, 4, 9], [4, 8, 9, 6, 11, 12, 1, 3, 7, 5, 2, 10], [3, 1, 6, 10, 2, 4, 9, 11, 8, 7, 12, 5], [12, 5, 4, 7, 8, 3, 10, 1, 6, 11, 9, 2], [8, 9, 2, 11, 6, 5, 12, 7, 4, 3, 10, 1], [6, 3, 8, 2, 12, 7, 5, 9, 10, 4, 1, 11], [7, 12, 5, 1, 4, 11, 8, 10, 2, 9, 6, 3], [11, 4, 10, 9, 3, 1, 6, 2, 12, 8, 5, 7]]

backtrackingMRVfwd:

Execution Time: 1.07198498077

Consistency Checks: 144

Solution: [[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], [5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4], [9, 10, 11, 12, 1, 2, 3, 4, 5, 6, 7, 8], [2, 7, 1, 3, 10, 9, 4, 5, 11, 12, 8, 6], [10, 11, 12, 5, 7, 8, 2, 6, 3, 1, 4, 9], [4, 8, 9, 6, 11, 12, 1, 3, 7, 5, 2, 10], [3, 1, 6, 10, 2, 4, 9, 11, 8, 7, 12, 5], [12, 5, 4, 7, 8, 3, 10, 1, 6, 11, 9, 2], [8, 9, 2, 11, 6, 5, 12, 7, 4, 3, 10, 1], [6, 3, 8, 2, 12, 7, 5, 9, 10, 4, 1, 11], [7, 12, 5, 1, 4, 11, 8, 10, 2, 9, 6, 3], [11, 4, 10, 9, 3, 1, 6, 2, 12, 8, 5, 7]]

backtrackingMRVcp:

Execution Time: 1.31171838081

Consistency Checks: 144

Solution: [[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], [5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4], [9, 10, 11, 12, 1, 2, 3, 4, 5, 6, 7, 8], [2, 7, 1, 3, 10, 9, 4, 5, 11, 12, 8, 6], [10, 11, 12, 5, 7, 8, 2, 6, 3, 1, 4, 9], [4, 8, 9, 6, 11, 12, 1, 3, 7, 5, 2, 10], [3, 1, 6, 10, 2, 4, 9, 11, 8, 7, 12, 5], [12, 5, 4, 7, 8, 3, 10, 1, 6, 11, 9, 2], [8, 9, 2, 11, 6, 5, 12, 7, 4, 3, 10, 1], [6, 3, 8, 2, 12, 7, 5, 9, 10, 4, 1, 11], [7, 12, 5, 1, 4, 11, 8, 10, 2, 9, 6, 3], [11, 4, 10, 9, 3, 1, 6, 2, 12, 8, 5, 7]]

minConflict:

Execution Time: 76.8309788803

Consistency Checks: 100001

Solution: [[3, 8, 4, 9, 6, 10, 11, 5, 12, 6, 2, 7], [1, 12, 6, 11, 9, 2, 3, 7, 4, 1, 11, 10], [7, 2, 5, 10, 8, 4, 1, 12, 3, 9, 5, 8], [10, 4, 2, 1, 5, 12, 6, 8, 7, 3, 9, 11], [8, 6, 9, 12, 11, 3, 7, 4, 1, 2, 10, 5], [4, 3, 7, 5, 2, 9, 10, 1, 11, 8, 6, 12], [9, 1, 3, 4, 10, 5, 8, 11, 2, 12, 7, 6], [2, 5, 12, 6, 3, 7, 4, 9, 10, 11, 8, 1], [11, 7, 10, 8, 12, 1, 2, 6, 9, 5, 3, 4], [6, 10, 8, 2, 4, 8, 9, 3, 5, 7, 12, 9], [5, 11, 4, 3, 1, 6, 12, 10, 8, 4, 1, 2], [12, 9, 1, 7, 7, 5, 11, 2, 6, 10, 4, 3]]

**INPUT #2:**

4,2,2;

-,-,-,-;

-,-,-,-;

-,-,-,-;

-,-,-,-;

Backtracking:

Execution Time: 0.000298502994652

Consistency Checks: 16

Solution: [[1, 2, 3, 4], [3, 4, 1, 2], [2, 1, 4, 3], [4, 3, 2, 1]]

backtrackingMRV:

Execution Time: 0.00483292599221

Consistency Checks: 16

Solution: [[1, 2, 3, 4], [3, 4, 1, 2], [2, 1, 4, 3], [4, 3, 2, 1]]

backtrackingMRVfwd:

Execution Time: 0.0132979235225

Consistency Checks: 16

Solution: [[1, 2, 3, 4], [3, 4, 1, 2], [2, 1, 4, 3], [4, 3, 2, 1]]

backtrackingMRVcp:

Execution Time: 0.00552016712746

Consistency Checks: 16

Solution: [[1, 2, 3, 4], [3, 4, 1, 2], [2, 1, 4, 3], [4, 3, 2, 1]]

minConflict:

Execution Time: 0.00102722663776

Consistency Checks: 14

Solution: [[1, 2, 3, 4], [3, 4, 1, 2], [2, 1, 4, 3], [4, 3, 2, 1]]

**INPUT #3:**

6,2,3;

-,-,-,-,-,-;

-,-,-,-,-,-;

-,-,-,-,-,-;

-,-,-,-,-,-;

-,-,-,-,-,-;

-,-,-,-,-,-;

Backtracking:

Execution Time: 0.000588452894902

Consistency Checks: 36

Solution: [[1, 2, 3, 4, 5, 6], [4, 5, 6, 1, 2, 3], [2, 1, 4, 3, 6, 5], [3, 6, 5, 2, 1, 4], [5, 3, 1, 6, 4, 2], [6, 4, 2, 5, 3, 1]]

backtrackingMRV:

Execution Time: 0.0274071080491

Consistency Checks: 36

Solution: [[1, 2, 3, 4, 5, 6], [4, 5, 6, 1, 2, 3], [2, 3, 1, 5, 6, 4], [5, 6, 4, 2, 3, 1], [3, 1, 2, 6, 4, 5], [6, 4, 5, 3, 1, 2]]

backtrackingMRVfwd:

Execution Time: 0.0328896415612

Consistency Checks: 36

Solution: [[1, 2, 3, 4, 5, 6], [4, 5, 6, 1, 2, 3], [2, 3, 1, 5, 6, 4], [5, 6, 4, 2, 3, 1], [3, 1, 2, 6, 4, 5], [6, 4, 5, 3, 1, 2]]

backtrackingMRVcp:

Execution Time: 0.0356582782193

Consistency Checks: 36

Solution: [[1, 2, 3, 4, 5, 6], [4, 5, 6, 1, 2, 3], [2, 3, 1, 5, 6, 4], [5, 6, 4, 2, 3, 1], [3, 1, 2, 6, 4, 5], [6, 4, 5, 3, 1, 2]]

minConflict:

Execution Time: 14.284328379

Consistency Checks: 100001

Solution: [[3, 5, 1, 2, 4, 6], [2, 6, 4, 1, 5, 3], [5, 3, 1, 4, 6, 2], [2, 4, 6, 3, 1, 5], [4, 1, 3, 6, 2, 5], [6, 5, 2, 1, 3, 4]]

**Conclusion**

**Backtracking**

This is a simple brute force approach which will try out all possibilities. If fail, it will backtrack and check other possibilities

**Backtracking + MRV + LCV**

This method uses CSP method – MRV (Minimum remaining values). We find the cell in the Sudoku board which has least number of possible legal values, this cell is filled first and every time in recursive call we find such MRV cell that apply our backtracking algo. Also we are filling our MRV cell in LCV values order.

By this we can eliminate many branches that would probably lead to failure. Consistency checks using this method are very much less than in normal backtracking method

**Backtracking + MRV + LCV + Forward Checking**

This method uses CSP method – Forward checking, a extra check is put over Backtracking + MRV + LCV.

In forward checking we check that after assignment of a LCV value in selected MRV cell, if any unassigned cell in corresponding domain of MRV cell has some legal values remaining to put, if 0 legal values for that cell we return false and backtrack.

By this method we prune search branches which are going to fail eventually

**Backtracking + MRV + LCV + CP**

As soon as we add Constraint Propagation the code starts checking Arc consistency. This means that the code would prune higher number of potential failure branches thus exploring less branches. The number of pruned branches is higher than forward checking.

**Min-Conflicts**

Min conflicts work on a board state where all cells have some assigned values. At a particular board state it will it will allow values with unsatisfied constraints. This is an iterative algorithm unlike other recursive algorithms. This algorithm picks a random conflicted cell and calculate some value for that cell which leads to minimum conflicts in that cell’s corresponding domain. Implemented iteratively rather than recursively. This is bounded by a limited number of iterations (100000 iteration check in our code).