**Sudoku Final AI Report**

**Team Name:** Sudokask

**Member #1:** Raymond Tu; tur2; 18179402

**Member #2:** Mark Sulimov; msulimov, 18871498

1. **Minimal AI**

**I.A. Minimal AI Algorithm**

Our Minimal AI algorithm was forward checking along with assignment of a variable if it only had one value. For all assigned variables, we removed the assigned variable's value from its neighbors' domains. Then if there is only one value left for a variable, we would assign it, and then do forward checking on that variable. If there are no values left, then we don't have a valid solution and must backtrack. We improved our Forward Checking by adding an additional optimal parameter: last\_assigned\_arg, so we could make our algorithm more efficient by only checking one variable's neighbors.

**I.B Describe your Minimal AI algorithm's performance:**

|  |  |  |  |
| --- | --- | --- | --- |
| Board Size | Sample Size | Score (Average backtracks) | Worlds Complete |
| 9x9 | 100 | 42.52 backtracks | 100 |
| 12x12 | 100 | 6849.11 backtracks | 99 |
| 16x16 | 100 | 80646 backtracks | 90 |
| 25x25 | 100 | N/A | 0 |
| Total Summary | 400 | 29,179 backtracks | 289/400 |

1. **Final AI**

**II.A. Final AI Algorithm**

Our Final AI algorithm has many improvements compared to our Minimal AI. We used several additional pruning methods and heuristics in addition to forward checking. We implemented hidden pair pruning, which is when if a pair of values occur in exactly two neighbor variables, then we can eliminate all of the other values in those two neighbor variables. We also implemented naked pair pruning, which is when if a pair of values is alone in two neighbor variables, then these values can be removed from all other neighbor domains. We then implemented Norvig's Check, which is when if a constraint has only one possible place for a value, then we assign the only variable with that value.

We also utilized variable and value ordering heuristics. For our variable ordering heuristic, we found the smallest domain variable, then used minimum degree as a tiebreaker, and then the presence of the highest frequency assigned value as the second tie breaker. For our value ordering heuristic, we used the heuristic of most frequently assigned value.

These heuristics greatly improved the amount of boards solved, and greatly decreased the amount of backtracks.

**II.B. Describe your Final AI Algorithm's performance:**

|  |  |  |  |
| --- | --- | --- | --- |
| Board Size | Sample Size | Score | Worlds Complete |
| 9x9 | 100 boards | 0.24 backtracks | 100 |
| 12x12 | 100 | 0.2 backtracks | 100 |
| 16x16 | 100 | 0.73 backtracks | 100 |
| 25x25 | 100 | 202.13 backtracks | 100 |
| Total Summary | 400 | 50.825 backtracks | 400/400 |

1. **Suggestions for improving this project.**

This project could be improved by improving the way boards are generated, so that only solvable boards are generated. This could be done by starting with a solution, removing some values, and then performing transformations on the board, such as rotations and reflections. Another improvement could be to have more required heuristics to implement, such as hidden pair pruning or x-wings.