Experiments

We used the Python 'trace' module to see how many times various methods were called when solving a CSP. We also used the Python 'timeit' module to observe the runtime performance of the program. We chose to take the average of the execution times of 500 executions of the solve_csp method to do so.

Trace Results

The following table represents how many times various methods were called during runs of our program.

What the binary_constraint_satisfied row represents may not be immediately clear. The binary_constraint_satisfied method is called whenever the program needs to check if a binary constraint is satisfied, so its called often when forward checking, when checking that the assignment is complete during a certain stage of the algorithm, etc.

Australia Map Coloring

Method	With Forward Checking	No Forward Checking
__backtracking_search	11	18
binary_constraint_satisfied	85	203

Sudoku

Method	With Forward Checking	No Forward Checking
__backtracking_search	82	*Massive amount*
binary_constraint_satisfied	6986	*Massive amount*

Note: without forward checking, the Sudoku problem could not be solved in a reasonable amount of time (it ran for 3-4 hours before it was stopped forcibly). This means that both __backstracking_search and binary_constraint_satisfied get called a very large number of times.

Timeit Results

Problem File	With Forward Cecking	No Forward Checking
Test Australia.txt	3ms	5ms
Test Sudoku.txt	735ms	*> 4 hours*

Note: as mentioned in **Trace Results**, the runtime of backtracking search without forward checking when applied to the Sudoku problem was too long to wait for. Thus, we do not have an exact value, but know it must be greater than 4 hours.

Analysis

Forward checking clearly has a massive effect on the efficiency of backtracking search. It may not make that much of a difference for small problems like the Australia map coloring problem, but for Sudoku it is essential that it be used if the CSP is to be solved in a reasonable amount of time.

This is as expected. The number of leaves generated by backtracking search in the worst case is d^n, and since forward checking reduces the number of values in the domain of variables (sometimes significantly), it reduces the base of the exponential expression and thus reduces greatly the leaves generated and thus the runtime of the algorithm.