CSC384 – Assignment 2 Report

Tanzeem Chowdhury - 997574726

**Brute Force Algorithm**

This implementation attempts to solve the sudoku puzzle using an exhaustive search method.

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparing performance of the three search algorithms for each of the test cases listed in the appendix** | | | |
| Algorithms: | Brute Force | Back-Tracking | Forward Checking with MRV |
| puzzle1.txt (Test case 1) | | | |
| Total clock time (ms) | N/A Too Long | 5.69796562195 | 11.7399692535 |
| Total search time (ms) | N/A Too Long | 5.65981864929 | 11.6810798645 |
| Nodes generated | ? | 687 | 41 |
| puzzle2.txt (Test case 2) | | | |
| Total clock time (ms) | N/A Too Long | 64.5699501038 | 20.1671123505 |
| Total search time (ms) | N/A Too Long | 64.5248889923 | 20.1210975647 |
| Nodes generated | ? | 8008 | 58 |
| puzzle3.txt (Test case 3) | | | |
| Total clock time (ms) | N/A Too Long | 117.506980896 | 62.9210472107 |
| Total search time (ms) | N/A Too Long | 117.410182953 | 62.8731250763 |
| Nodes generated | ? | 14340 | 167 |
| puzzle4.txt (Test case 4) | | | |
| Total clock time (ms) | N/A Too Long | 37.0609760284 | 33.0669879913 |
| Total search time (ms) | N/A Too Long | 37.0230674744 | 33.0278873444 |
| Nodes generated | ? | 4621 | 86 |
| puzzle5.txt (Test case 5) | | | |
| Total clock time (ms) | N/A Too Long | 20.4617977142 | 28.9080142975 |
| Total search time (ms) | N/A Too Long | 20.41888237 | 28.8689136505 |
| Nodes generated | ? | 2359 | 66 |

Results Analysis:

For all five puzzles, running brute force ran too long, unable to yield the clock/search time and number of nodes generated for any of them. This is expected, as trying every permuation possible has a worst case complexity of 9n , where n is the number of blank cells. Taking puzzle 1 as an example, which has 41 empty cells, there is 941 permuations of puzzle state, and surfing through these is an exaustive method.

When comparing the number of nodes generated between the back tracking anf FC-MRV methods, we get expected results for all puzzles where the number of nodes generated with FC-MRV are significantly smaller than BT (eg. For puzzle 3, FC-MRV only expanded 1.2% of the nodes expanded by BT. While theoretically, the worst case complexity for both BT and FC-MRV remains 9n (n = # of empty cells), best and average case significantly increase, with BT able to find solutions faster due to the fact it checks constraints after placing a possible value and back tracking when stuck, instead of starting from scratch, and FC-MRV faster due to the fact it checks contrainsts prior to placement, with the order of said placement determined dynamically based on the MRV heuristic, instead of a predefined static order.

While FC-MRV generated fewer nodes, unexpected results was observed where in some cases, the BT method performed faster time-wise compared to FC-MRV, even when generating substantially lesser nodes (e.g. puzzle 1 took double time with FC-MRV, puzzle 4 ran roughly the same time as BT, and puzzle 5 was a little longer than BT). This can attributed to the fact that in some scenarios, a significantly larger amount of computation is required to determine the MRV heuristic, compared to simply backtracking. Another interesting unexpected result was that most puzzles gave different solutions when using BT and FC-MRV, however both were correct valid solutions. With BT searching linearly and back tracking, it makes sense it was able to find an early solution out of several in it’s search, compared to FC-MRV searching optimally based on the heuristic, which could attribute to why in some cases, BT performed faster.