Daniel Hanna, Garrett Hope
Project 2 Report

Overview & Implementation

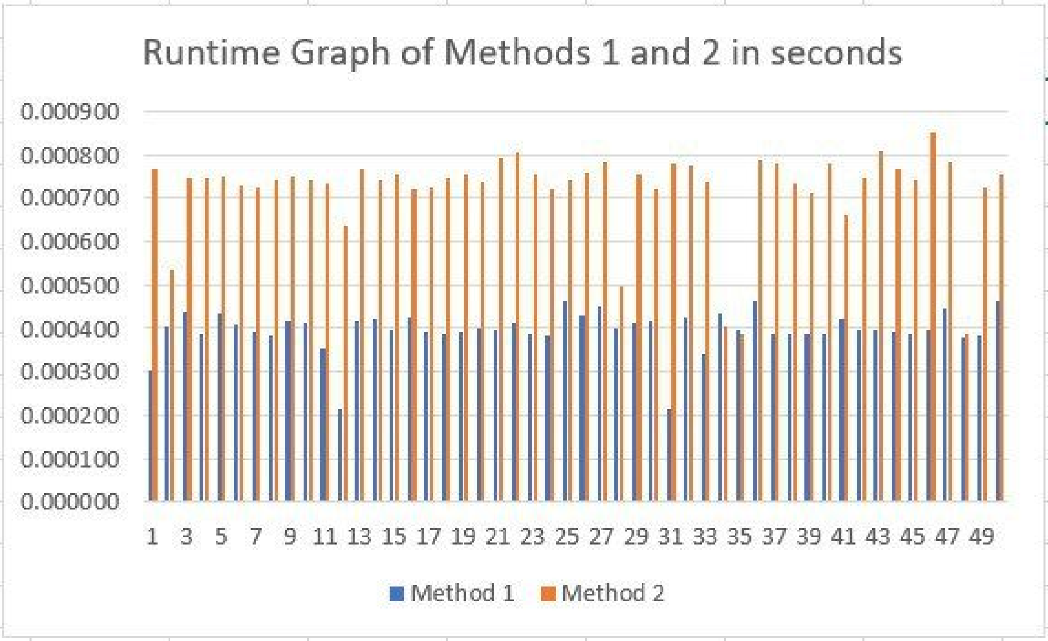
This project involved writing a multithreaded application that splits the task of checking a Sudoku board for validity. In this code, the sudoku board is simulated via a text file which contains a 9x9 grid of numbers. A structure variable is utilized to keep track of the row, column, and sudoku board, and appropriate row, column, and square variables are managed by utilizing this structure to define which section of the board is being analyzed. Two methods are then implemented. In the first method, 11 total threads are utilized to check the board. One thread is utilized to check all rows. Another thread is utilized to check all columns. The remaining nine threads have been utilized to check each of the 9 sub-grids that exist within a typical sudoku board. In the second method, a total of 27 threads are utilized. Each of the first 9 threads check each row of the board. The next 9 threads check each column of the board. And the last 9 columns check each of the sub-grids that exist within the board (similar to the first method implementation). The user is capable of selecting which implementation to utilize by inputting either a “1” or a “2” into the command line when running the program.

In method 1, to actually check the validity of the sudoku board, three functions are used. There is one function that ensures each row has the numbers 1-9. If this is true, the function returns a value of 1. The same methodology is used to ensure each column has the values 1-9. Then a function is used to check that each 3x3 square has the values 1-9. This function is called 9 times, once for each square. Once this is complete, a check is performed to see if any of the functions returned a value of 0, meaning the board is not valid.

In method 2, a very similar approach is taken, however now each individual row and column, are assigned a thread. This means that in the check row and check column functions, the board is only iterated over in one dimension.

<u>Comparison & Results</u>

After both methods have been fully implemented, a statistical test is implemented to test the project's null hypothesis, which states that "there is no statistically significant difference between the two methods implemented." To test this hypothesis, a global time variable was utilized, which kept track of the time elapsed between the start of the program and the program outputting its findings (where the board is either found to be a valid sudoku solution, or it is not found to be a valid solution). Each implementation was run 50 times, and the global results recorded. The average of the 50 runs was then calculated for each method. The results are depicted below.



Avg 1:	0.000396
Avg2:	0.000719

It can clearly be seen that method 1 (the method involving only 11 threads) consistently outperformed method 2 (the method involving 27 threads) in terms of execution time. When averaging the run times, it could be seen that method 1 held an average time of 0.000396 seconds, where as method 2 held an average time of 0.000719 seconds. Both methods result in a difference of execution time that is negligible to the human eye. However, upon this numerical analysis, it can be seen that method two’s execution time is nearly double that of method 1. Thus, as a result, it can be concluded that there is a significant difference between the two methods, consequently rejecting the null hypothesis.