David Kopp

9-26-15

Comp496 ALG

Project #1: Stable Matching Problem

David Kopp

9-26-15

Comp496ALG

Project #1: Stable Matching Problem

Run Time Analysis of Brute Force Algorithm:

The run time for the brute force solution of the stable matching problem grows very quickly for every n added. See the graph below for run time execution of n = 6, 8, 10, 11, 12.

Prediction:

T(13) = ~1.05 trillion msecs

T(14) = ~17.09 trillion msecs

\*\*These predictions are based on the average time that each permutation takes per msec. Example: 6! / 38, 8! / 139, … , 12! / 2299612. (average) \* 13! = prediction

Big O Analysis: (Working by methods)

ReadInputFile() : O(C + 2n + 2n^2)

-Setting size of data structures and initializing a variable: constant = O(C)

-Initializing Arrays: O(2n)

initArray():

-Loop – O(n)

initPermutationArray():

-Loop – O(n)

-Inputting data into malePreferences[][] – O(n^2)

-Inputting data into femalePreferences[][] – O(n^2)

bruteForceSolution(): O(C +2n +Cn^2 + 6n^3)

checkStability(): O(n)

printPairs(): O(n)

-Loop – O(n)

isStable(): O(C+ Cn^2 + 4n^3)

-Initialize variables – O(C)

-Loop – O(n)

-loop – O(n)

2\*maleRanking(): O(C + 4n)

-Initialize variable – O(C)

-Loop – O(n)

2\*femaleRanking(): O(C + 2n)

-Initialize variable – O(C)

-Loop – O(n)

findPair(): O(C + n)

-Initialize variable – O(C)

-Loop – O(n)

permutations(): lexicographic Permutation Algorithm O(n^2\*n!) + O(C + Cn^2+6n^3)

checkStability(): – O(C + Cn^2+6n^3)

Total = O(C + 2n +2n^2 + 2\*(C + 2n + Cn^2 + 6n^3) + n^2\*n!)

Final Analysis = O(n^2\*n!)

Run Time Analysis of G-S Stable Matching Algorithm:

The run time for the Gale Shapley Algorithm for stable matching grows much slower than that of the brute force solution. Unlike the brute force solution it grows significantly every 200 for n rather than in brute force which is every n. See the graph on the next page for n = 50, 100, 200, 400, and 500.

Prediction:

T(1000) = ~1400msecs

\*\*This prediction is by looking at an exponential tread-line. (Note: data seems to almost follow the curve closely.)

Big O Analysis: (Working by methods)

ReadInputFile() : O(C + 2n + 2n^2) {From BFS solution}

-Setting size of data structures and initializing a variable: constant = O(C)

-Initializing Arrays: O(2n)

initArray():

-Loop – O(n)

initPermutationArray():

-Loop – O(n)

calculateMatches(): O(n^3)

-Loop – O(n)

-Loop – O(n)

femalePrefers(): O(n)

-Loop – O(n)

printPairs(): O(n)

-Loop – O(n)

Total = O(C + 5n + 2n^2 + n^3)

Final Analysis = O(n^3)

Conclusion:

In conclusion, the brute force solution was very slow as n increased by one. The analysis showed the brute force was O(n^2\*n!), which is very slow. The permutation algorithm method was the upper bound, but no matter how cleaver the permutation algorithm would be O(n!) at minimal. The Gale Shapley Algorithm was much faster and efficient than the brute force solution with a performance metric of O(n^3) and was much easier to predict at higher n values in a graph. In the Gale Shapley solution the calculation of the matches was the deciding factor on performance. In the end, brute force will get you an answer but it is not very time efficient when compared to a more direct algorithm, like the Gale Shapley Algorithm.