Colin McAllister

Collin Smith

CPSC 3283

M6 Programming Assignment

In this assignment, we will show that a recursive algorithm with the same time complexity as its iterative counterpart () will take more time on average. This is caused by the cost of each recursive call. Not only do each of these calls take time, but they also take space on the stack, which makes them less efficient in both time and space complexity. In order to test this theorem, we wrote a program that captures the time it takes to run the recursive algorithm and the time it takes to run the iterative algorithm on the same arrays and . We chose to initialize both arrays to size , and then perform these algorithms on the first members. To see a more realistic trend, we took the average of 40 executions at each The code was written in Java and can be compiled using any modern IDE such IntelliJ or jGrasp. This program works as intended.

Figure 1 shows the total of all 40 executions at each which we will call We chose to include this graph because it shows how much more the recursive algorithm grows than the iterative one. In figure 2, we took for the recursive algorithms and divided it by of the iterative algorithms. By performing this calculation, we are showing that the time complexity of the recursive algorithm (in the numerator) is growing much faster than the time complexity of the iterative algorithm (in the denominator). When the numerator of a fraction grows faster than the denominator, the resulting coefficient grows larger, which is what is pictured in figure 2. If we had placed of the iterative over for the recursive, we would have seen a number that would be declining towards 0.

Based on the visualization in figure 2, there were no instances in which the recursive algorithm outperformed the iterative algorithm after . This proves that the iterative algorithm is consistently faster than the recursive counterpart, even though they both have the same asymptotic time complexity of

Figure : Recursive Time vs Iterative Time

Figure : Recursive Time / Iterative Time