CSCE 350

Powell Fendley

9/12/16

Program 1 Results Summary

The program results allow for a better understanding of the cost of not paying attention to minor operations like division and multiplication. While performing under 100 operations, the amount of subtractions and the amount of divisions are close together, but considering that divisions can be 10-100 time more expensive that subtractions, it is slightly worth the extra ‘hack’ in the algorithm. When 10,000 operations are performed, divisions on the naïve algorithm balance out around 85,000, whereas subtractions on the second algorithm shoot up into the 300,000’s. The divisions on the second algorithm are at around 20,000. However, this is a big difference. If we assume divisions are worth 10 subtraction operations, the naïve algorithm hits 850,000 operations while the subtraction ‘hack’ leaves us with 500,000 operations. This is only exaggerated in the test that runs 100 million operations, as the numbers of subtractions go into the billions but still save operations when compared to the divisions\*10. This is also representing one of the better case scenarios. Depending on the software and hardware of a machine, divisions could be 100x as expensive as subtractions, which would make the savings of the subtraction algorithm priceless in terms of time.

Further, the naïve algorithm has a large order of magnitude for every run through. The naïve algorithm always makes Big O of (D:Unbounded) where D is the amount of time it would take to run a division operation and Unbounded representing how many passes it could potentially take to find the GCD. Similarly, the naïve Big Omega sits at (D:1), where the algorithm would find the GCD on the first division of every pair. The naïve Big Theta, from my data set, sits around (D:8.5), where it takes 8.5 divisions per pair to find the GCD. The subtraction algorithms orders of magnitude perform much better. With a Big O of (D:Unbounded) it could potentially take just as long as the naïve algorithm to chew through all of its pairs. However, the subtraction algorithm has a stunning Big Omega of (S:2), where it would take two subtractions (including the number comparison included in my algorithm that isn’t also present in the naïve one) to complete. Meaning for each pair it would only have to perform two subtractions per pair to find the GCD for each pair. This is much faster than the naïve algorithm, especially if hardware and software push the divide costs to 100x+ what their subtraction costs are. Most importantly, the Big Theta of the subtraction algorithm is (D:2, S:30), meaning it would take two divisions and thirty subtractions on average per pair to finish its GCD calculations. If we assume a division cost of 10x greater than subtraction, the subtraction algorithm beats the naïve algorithm by 3 divisions on average per pair. This is why the subtraction algorithm becomes more valuable the more pairs you process.