Diego Valdes
CSCE 350
Sept. 7, 2016

Program 1 Analysis

Comparison of two greatest common denominator methods

I have written two algorithms, GcdNaive and GcdSubtract, designed to return the greatest common denominator of two numbers. GcdNaive accomplishes this task using only division, while GcdSubtract uses a combination of subtraction and division. Both were tested using three data files containing 100, 10,000, and 10,000,000 records. The purpose is to cut the number of divisions (preferably by a third), thus using less cpu resources to accomplish the same task.

- 100 records
  - GcdNaive: 394 divisions
  - GcdSubtract: 531 subtractions, 300 divisions
  - Cost of division: (394 – 300)/300 = 0.31 -> 3.22
- 10,000 records
  - GcdNaive: 94,291 divisions
  - GcdSubtract: 54,080 subtractions, 85166 divisions
  - Cost of division: (94,291 – 54,080)/54,080 = 0.10 -> 9.3
- 10,000,000 records
  - GcdNaive: 181,743,316 divisions
  - GcdSubtract: 54,123,849 subtractions, 172,618,478 divisions
  - Cost of division: (181,743,316– 54,123,849)/ 54,123,849 = 0.05 -> 18.9

Analysis

When comparing the first test of GcdNaive to GcdSubtract, it is determined that the cost of each division is 3.22 and that the difference in divisions between the two is roughly a third. As the sample size increased the disparity between divisions decreases, but the cost of the divisions rises significantly to 9.3 in test 2 and 18.9 in test 3.

Based on these results, GcdSubtract is a better algorithm, as the subtractions used are keeping the cost of the divisions down.

Further Analysis

The cost of divisions takes a dramatic rise as the test sample increased. It would be worth examining GcdNaive and try to determine what in the algorithim is causing such a drastic rise in the cost of divisions and what can be done to bring them to be more in line with GcdSubtract.