Group 9: Chuck and Travis

Place took: 7th

Sum of Median Times: 2784.0 ms

Description of our algorithm

Sorting

- Uses Bucket Sort to sort the prime products and TimSort to sort the actual values.
  - Bucket sort was created using a linked list of nodes. Each node contains the product of the primes, the number of items in the bucket currently and the node directly to the right of it..
  - All of the buckets are linked to a single ArrayList. Once the ArrayList has a length of the original array of values, all of the values in the ArrayList are copied over to an array, and TimSort is called on each individual bucket in the array.

Description of our algorithm

Computing Primes

- Uses the Sieve of Eratosthenes to generate the primes from 0 to n very quickly which we can use when checking for the product of prime factors of a long.
    - Done by using a BitSet of size n in which we set values in the range 2 to n- 1 as true. Then going through and setting multiples of any bit set as true to false.

Theoretical Efficiency of Worst Case

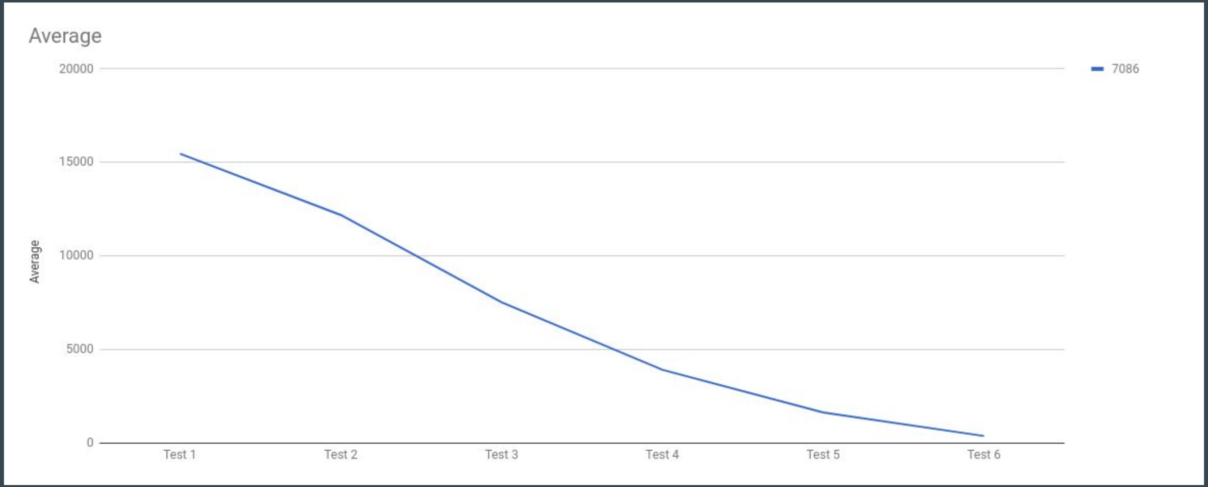
Our theoretical worst case dataset for our algorithm would be sorted and consist of all unique primes. The linked list would have the efficiency of Big-Theta($k * n^2$) with k being large.

Expected Efficiency

The expected efficiency for a normal case should be Big Theta ($k*n^2$), with the k value being substantially smaller than the worse case scenario.

What Worked and What We Would Have Done Differently

- The implementation of the modified sorting method and generation of primes dramatically decreased our time.
- Given more time, we would have optimized the generation of prime numbers for numbers over 100,000. This accounted for the largest percent of time taken.


Average
Average
Test 1
Test 2
Test 3
Test 4
Test 5
Test 6