Group 4

Ryan and Gregory

Overview

- Times: was not working by deadline.
- Times post comp date: sum of medians 2485.0
- Placed 3rd (but not actually)
- Sorting Algo:
  - Timsort with modified comparator. Converts array to an array of objects before sorting. Data stored everything as a converted fraction with a numerator and denominator.

```
private static class Data {
    public String str;
    public long numerator;
    public long denominator;
    public BigInteger bignumerator;
    public BigInteger bigdenominator;
    public int numlength;
    public int denlength;
```

Comparison method violates its general contract!

In converting the array of strings to an array of objects Data, the length of decimals after the decimal was used to convert to fractional representations in the same fashion as group 0. This was done to calculate the denominator. Later on we used numerator and denominator length to determine if we could fit our representation into a long. Our logic didn’t set the numerator length to what it actually was at the end of this process. I.e. we were comparing numbers that weren’t small enough to fit in long.

Worst case

Time Complexity: worst case for Timsort O(n*log(n))

Converting to array of data: O(n)

Comparator time complexity? Why is our Algorithm significantly faster? Comparing doubles appears to be significantly faster than comparing BigIntegers. We had a BigInteger representation for each fraction but only had to compare those representations for a small portion of Data. Our sorting algorithm only converts each decimal into a fraction one time. Unlike Group 0 which may convert a decimal to a BigInteger denominator and numerator several times each time that decimal is compared to a fraction.