Sorting Competition Group 12

Francisco Montanez
Luz Lopez

Results

In the sorting competition we took place 13.

Our sum of places was 26.

The sum of our medians was 36072.0.

Correctness

Our code was not entirely correct in the sense that it did not sort correctly, Group 8 pointed out to us that our method “productOfPrimes” was incorrect. It can be fixed by changing

```
for (int i = 0, j = 1; i < primes.size() - 1; i++)
        {
           if (primes.get(j) != primes.get(0))
           {
              product = primes.get(j) * primes.get(0);
              return product;
           }
```

To this:

```
for (int i = 0, j = 1; i < primes.size() - 1; i++)
                {
                   if (primes.get(j).compareTo(primes.get(0)) != 0)
                   {
                      product = primes.get(j) * primes.get(0);
                      return product;
                   }
```

What worked, What didn’t

Our code sorted incorrectly. That is because of the use of “!=” instead of “.compareTo”.

```
if (primes.get(j) != primes.get(0))

if (primes.get(j).compareTo( primes.get(0)))
```

Sorting Method Used

We used quicksort because it works best with large amounts of data, and because we had already implemented it for class.

Storing

We wrote a method to calculate the two smallest different prime factors of a number. In this method, as we find prime factors, we store them in an ArrayList.

We wrote our methods to take in longs but after we had finished writing them and testing them with long arrays we remembered that we are suppose to take in an array of strings. Due to this, we create an arraylist of longs the length of string array given. We add all the elements in string array to the long array. That is where we use our sorting method. Afterwards, we add the output to a string array and return that.

Pre-computing

I do not believe we had any pre-computations.

Theoretical Worst Case

Quicksort's theoretical worst case efficiency is O(n^2).

This happens when the array is already sorted in , reverse order or all elements are the same (if pivot is rightmost).

Its expected efficiency was O(n^2).

Most Interesting Feature

The most interesting feature about our algorithm is that it works.

A cool feature about it would be our “productOfPrimes”. It took us a while before we had it working properly.

What we would have done differently

We would have implemented one of our improved quicksort algorithms written for lab 3.

We would have used the random pivot quicksort.