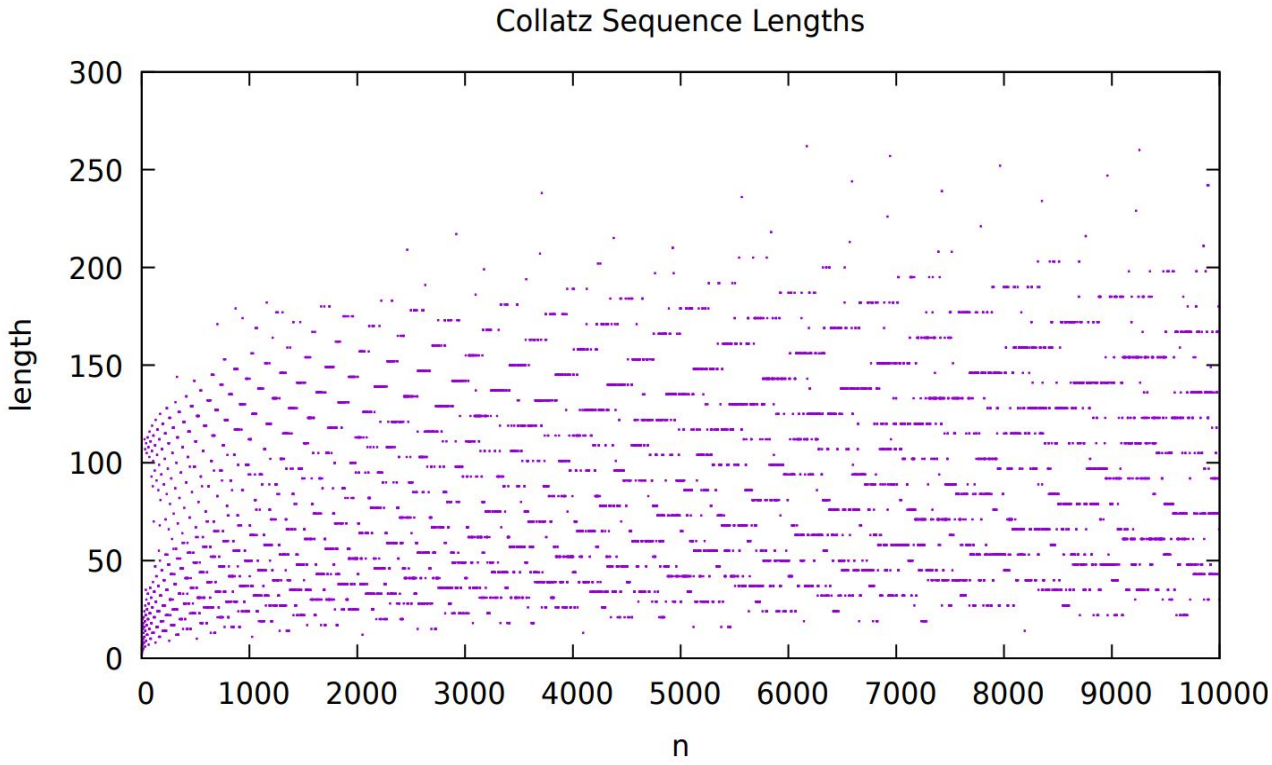
Graph One: Length



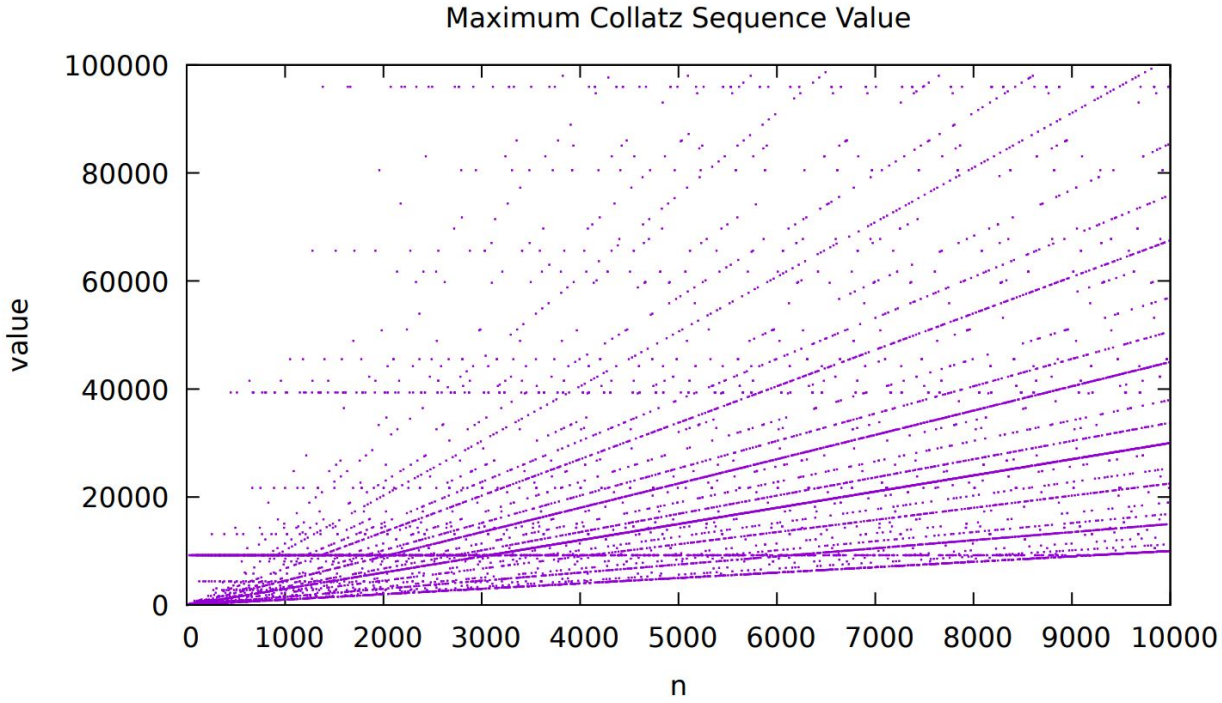
Graph one was made by finding the length of the Collatz Sequence for each value of N.

The Bash script I used for most of the formation of this graph was "***count=$((count+1))***". This is because I would loop through all returned values for a given Collatz sequence of N, by counting the values I was looping through, this would give me the length of the sequence.

The next script that was used for the formation of this graph was "***printf "$num $count\n" >> graph_one.dat***". This is because it wrote the current num in the loop and the length of the sequence to the data file in a form of "x y" that I used to plot this graph on Gnuplot.

Lastly, I used "***plot "graph_one.dat" using 1:2 ps 0.1***" as this is what plotted the points, creating the graph.

Graph Two: Max



Graph two was made by finding the maximum of the Collatz Sequence for each value of N.

The Bash script I used for most of the formation of this graph was

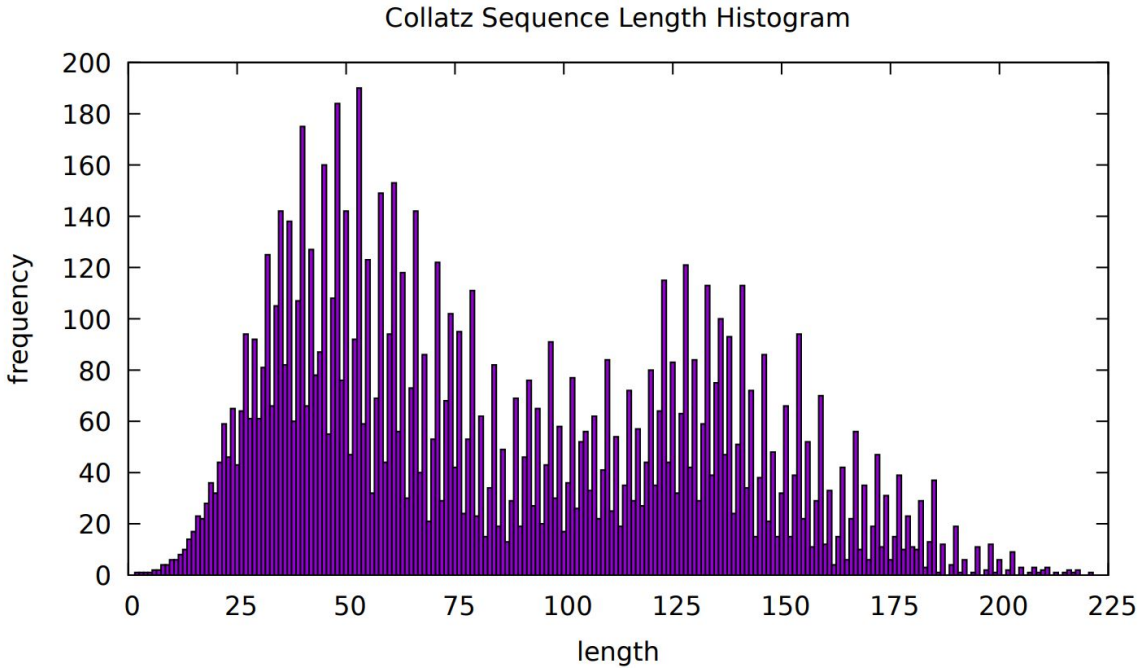
```
if [ $current -gt $max]
then
    max=$current
fi
```

This is because these couple of lines is what calculated the maximum value as I looped through all of the variables.

The next script that was used for the formation of this graph was "***printf "$num $max\n" >> graph_two.dat***". This is because it wrote the current num in the loop and the maximum value of the sequence to the data file in a form of "x y" that I used to plot this graph on Gnuplot.

Lastly, I used "***plot "graph_two.dat" using 1:2 ps 0.1***" as this is what ended up plotting the actual graph

Graph Three: Length Frequencies



Graph three was made by counting the occurrences of each length found for all Collatz values from 2 to 10,000.

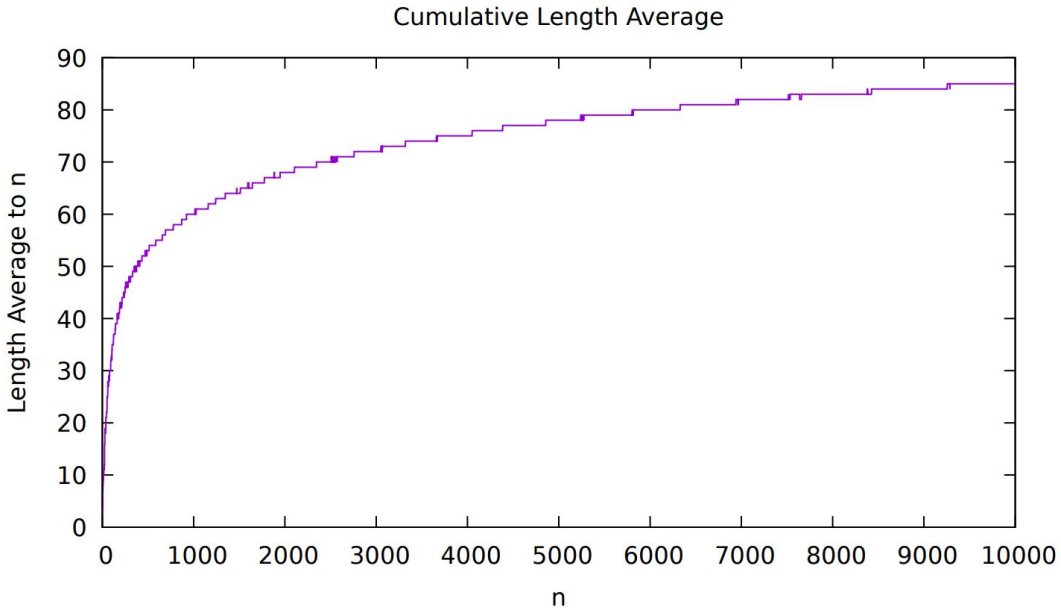
The Bash script that played a key role in the formation of this graph was
"***Printf "$(uniq -c testing.dat)" | awk '{print $2 " " $1}' > graph_three.dat***"

This is because to get the results that I wanted, I had to sort all of the lengths as the command “uniq -c” only counts the occurrences that are side-by-side. So by sorting it I was able to count all the occurrences. I then had to fix the output into being in a format that could be used by Gnuplot which included removing the whitespaces and swapping the two values so that “y x” was now “x y”. This command had the bonus of also creating the data points that could be put directly into graph_three.dat

The next command I used was ***printf “$count\n” >> temporary2.dat*** which created the file that would be sorted by the command above.

Lastly, I used ***plot “graph_three.dat” u 1:2:(1) w boxes*** as this is the command that created the histogram from my data.

Graph Four: Cumulative Average Lengths



The reason I decided to make this plot is actually because of an accident. While trying to graph the histogram above, I accidentally made a cumulative graph of the lengths. I noticed that this length looked strangely linear.

This got me thinking, Is the length of the Collatz sequence on average, logarithmic? As you choose higher and higher values, does the amount of steps approach some limit where the length is on average the same?

This made me google, “Are the steps required in the Collatz sequence logarithmic?” which lead me to this site https://gottwurfelt.com/2016/01/10/logarithmic-approximations-for-collatz/. I realized that I was not the first person to have this question. On this page, he graphed the cumulative average “steps” it would take to finish the Collatz sequence and noticed that the graph looked extremely logarithmic or at least radical. So I decided to plot this graph since my N went to twice the value.

After graphing this I noticed just how the graph looked like it could be logarithmic or radical! This got me thinking that you could possibly predict the number of steps (length) a Collatz sequence would be on average using a graph like this.

Because of this discovery, I decided to plot this graph as I found this extremely interesting.

Anyway, the main set of commands that enabled this plot was

totalSteps=$((totalSteps + count))
avg=$((totalSteps / (num-1)))
Printf “$avg\n” >> graph_four.dat

This is because this is how I managed to get the average length of the Collatz sequence up to the current point N and put it inside my graph_four.dat file

Lastly of course I used ***plot “graph_four.dat” with lines*** which created the actual plot of all the data.