Assignment 1 - WRITEUP.pdf

Purpose:

Make 3 different graphs to help represent and understand the patterns of the Collatz sequence.

$$S_{k+1} = \begin{cases} 1+3S_k & \text{if } S_k \text{ is odd} \\ \frac{1}{2}S_k & \text{if } S_k \text{ is even} \end{cases}$$

Graph 1

```
if [ ! -f ./tmp/lengths.dat ]  && [ ! -f ./tmp/maximum.dat ];
then
        for x in {2..10000};do
                ./collatz -n $x > ./tmp/c.dat
                #sleep 1
                l=$(wc -l ./tmp/c.dat)
                m=$(sort -nr ./tmp/c.dat|head -1)

                echo "$x $l" >> ./tmp/lengths.dat        # Place the data points into a file.
                echo "$x $m" >> ./tmp/maximum.dat
        done
fi
```

The if statement checks if the file already exists, if not then make the file. *-f* means file.
To use Gnuplot the .dat file should be stored in the format "$x $l"
./collatz -m $x > c.dat takes the stdout and puts it into a "c.dat" file. The single > means it will be overridden so I do not need to check if it exists or not.
$x is the counter of the for loop from 2 to 10000
$l is the length of the sequence.

- *wc -l* counts the number of lines "c.dat" file. This is needed to find the length of each output produced

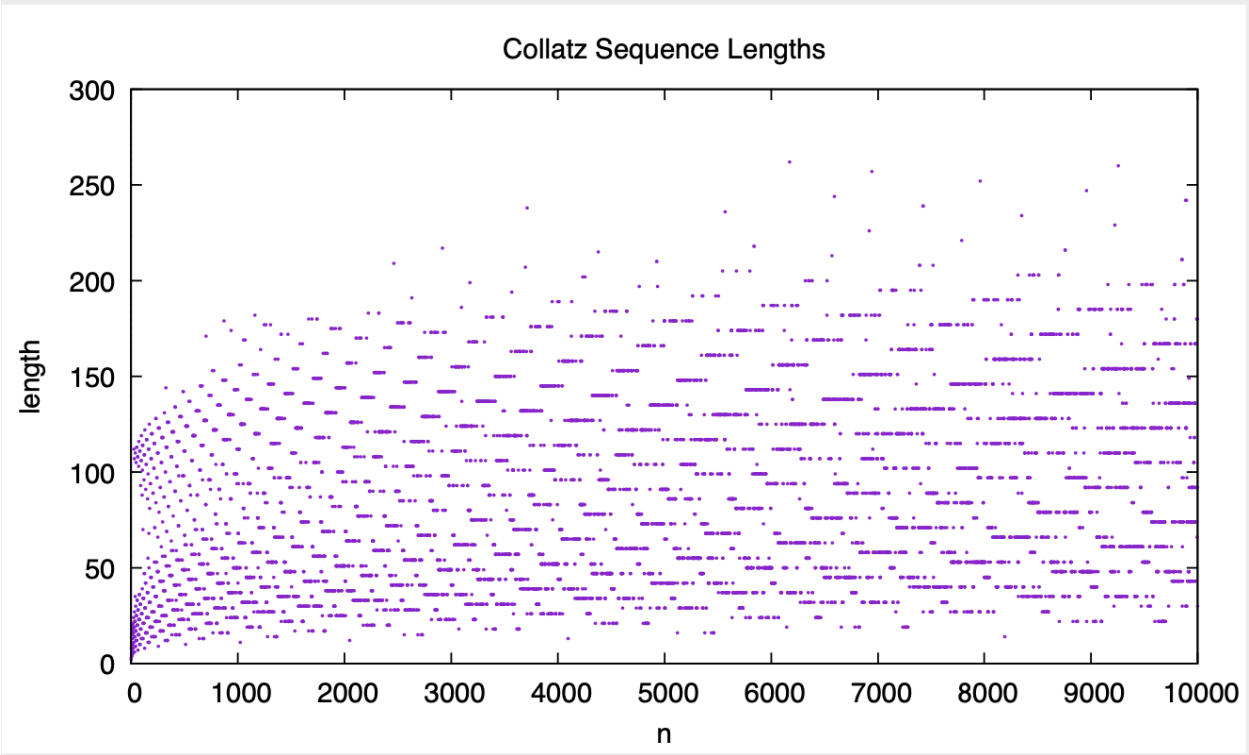
echo "$x $l" >> lengths.dat puts the two variables x and l into the format requires into the "lengths.dat" file
I did not need *sleep 1*. I feel very dumb. Waited 3 hours per run for no reason.

```
set terminal pdf

set output "sequence_lengths.pdf"
set title "Collatz Sequence Lengths"
set xlabel "n"
set ylabel "length"
set zeroaxis
set yrange [0:300]
plot "./tmp/lengths.dat" pt 0 ps 1
```

Graphing:

Pretty straightforward for this one. Label x as “n” and y “length”.
And *plot “lengths.dat”* which will, by default take the first column as x or n, the second column as y or l.



The points spread out from around 110-120 sequence lengths. The points cluster into progressively increasing lines. This implies as n increases, there are larger and larger ranges of n where the sequences are very similar or the same.

Since there is overlap and appear to be organized, the sequence lengths are an “either or” type situation where a range of n as n increases is likely to be one of some x possible sequence lengths with a few exceptions as shown by the plot.

Graph 2

To use Gnuplot the .dat file should be stored in the format “$x $m”
$x is the counter of the for loop from 2 to 10000
$m is the maximum of the file

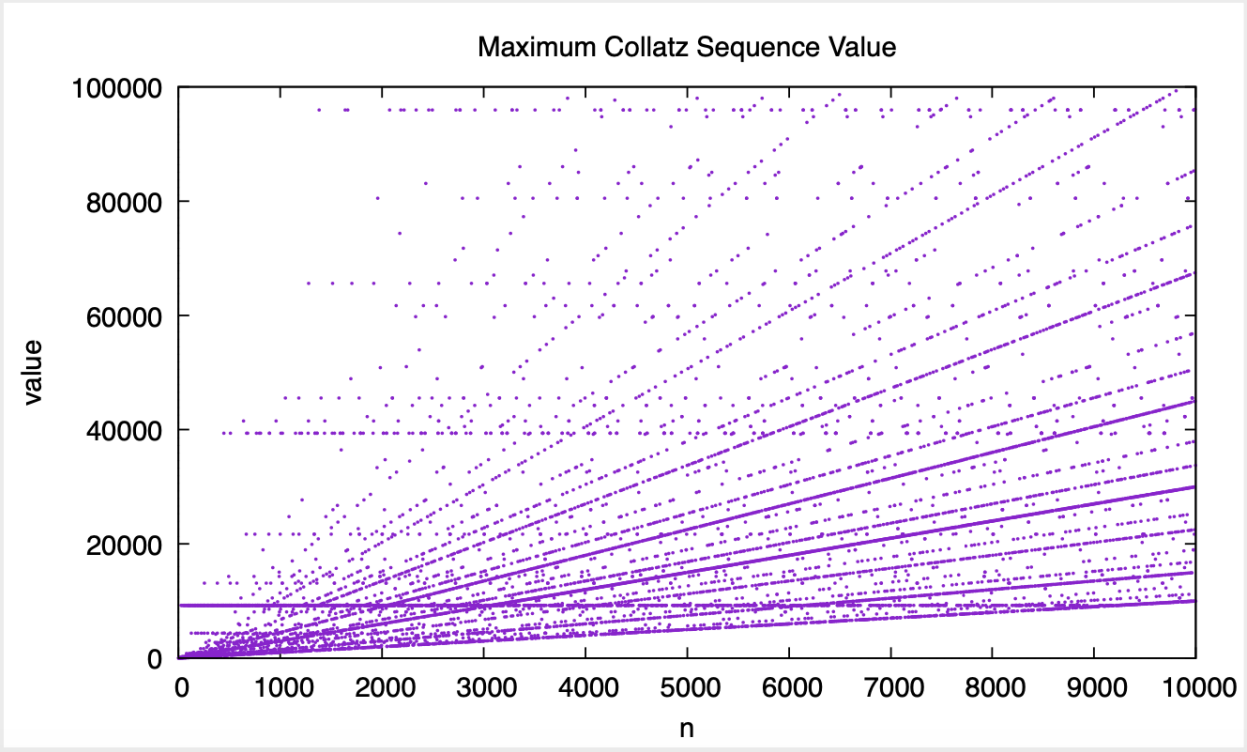
- *sort -nr* orders the file greatest to least. *head -1* takes the first line of the ordered file to get the highest number in the file

echo “$x $m” >> mamimum.dat puts the two variables x and m into the format required into the “maximum.dat” file

```
set output "maximum.pdf"
set title "Maximum Collatz Sequence Value"
set xlabel "n"
set ylabel "value"
set zeroaxis
set yrange [20000:100000]
plot "./tmp/maximum.dat" pt 0 ps 1
```

Graphing:
Same as Graph 1 except *ylabel* is named value because now it represents the maximum
yrange goes from 0 to a max of 10,000 because the values

are much larger



There is a solid line at the bottom the minimum value of a maximum is n.
There is a very solid line that fades as n increases going horizontally across the graph at a y-value around 10000.
There are also fewer solid lines around 5000, 20000, 40000 (the most distinct).

```
   69 190996
   71 4372
   87 21688
   92 95956
  143 250504
  197 39364
 1579 9232
```

```
awk '{print $2}' ./tmp/maximum.dat | sort -n| uniq -c| sort -n
```

Using command it formats the maximums in the format “$f $m”
The solid line is at y = 9232.
The 2nd most distinct line is y = 39364.
The 3rd most distinct line is an unseen line at y = 250504. Which is out of the 7 range of the graph.

The “lines” spread out more and more the higher the value. But it is likely if the range of n was increased the lines would be more distinct.

Graph 3

To use Gnuplot the .dat file should be stored in the format “$l $f”

```
echo "$(awk '{print $2}' ./tmp/lengths.dat | sort -n | uniq -c)" >> ./tmp/hist.dat
echo "$(awk '{print $2, $1}' ./tmp/hist.dat)" >> ./tmp/histogram.dat
```

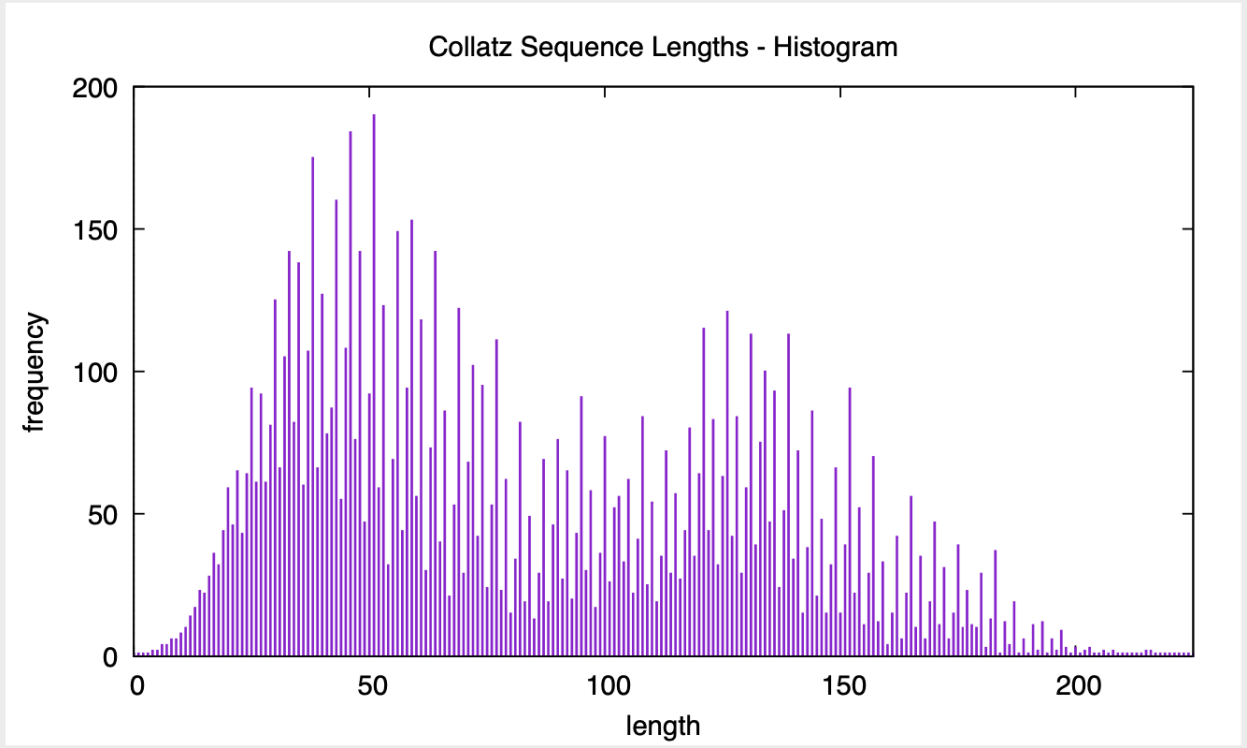
$l is the lengths
$f is the frequency

- *echo* is used to write an argument to a stdout and because of that I can use >> to trader it to “hist.dat”
- *awk ‘{print $2}’ lengths.dat* takes the second column of the “lengths.dat” file.
- *| sort -n* takes, pipes the now list of length at puts is told to put it in numerical order from the *-n*
- *| uniq -c* find repetitions *-c* tells it to put the number of repetitions before the number itself. So “hist.dat” is in the format “$f $l”
- *echo awk ‘{print $2, $1}’ hist.dat* puts the “$f $l” into “$l $f” and with >> puts the stdout into “histogram.dat”

Graphing:

```
set output "histogram.pdf"
#set style data histogram
#set style histogram clustered
set title "Collatz Sequence Lengths - Histogram"
set xlabel "length"
set ylabel "frequency"
set boxwidth 0.5
set zeroaxis
set yrange [0:200]
set xrange [0:225]
plot "./tmp/histogram.dat" using 2:xticlabels(25) with histogram
#plot "./tmp/histogram.dat" pt 0
```

yrange goes from 0 to a max of 200 because the frequency for n(2 to 10000) never goes above 200
xrange is 0 to 225 because that is what the assignment sheet example had.
Plotting a histogram slightly different, and will need to define the labels 2:xticlabels(25)



The graph is very dense around a frequency of 50. Meaning that most sequence lengths stay at or below 50 with the range n(2 to 10000).

```
asgn1 % awk '{print $2, $1}' ./tmp/histogram.dat |sort -n
```

The histogram has is bimodal, having two peaks. The most frequently appearing one can be found with the above command. It is easier if in descending order as the largest one will be right at the bottom without scrolling all the way up.