Assignment 1 - WRITEUP.pdf

Introduction:

For the write-up I will be focusing on data collection and graph generation, using *bash* and *gnuplot* respectively. Firstly I want to mention that in this assignment the code that produced the ‘Collatz Sequence’ was given to me, and my task was to focus on graphing the data, therefore I will go over the explanation of the sequence shortly.

The Collatz sequence is generated in the following manner:

Firstly the user inputs the argument integer that we will label ‘n’

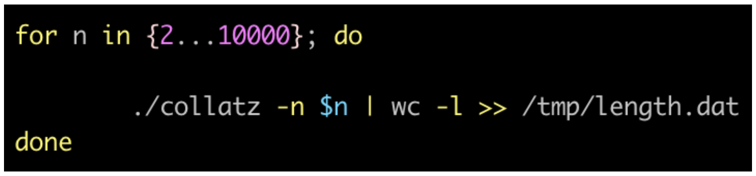
If n is odd → 3n +1
If n is even → n/2
The two above steps repeat indefinitely until ‘n’, the argument is equal to 1, in which case the program stop running. In addition, helping us collect the data, the program returns *n* after each iteration.

Collatz Sequence Lengths Graphs:

I have generated 2 separate graphs using *gnuplot* and also the same bash script method.

Bash Script

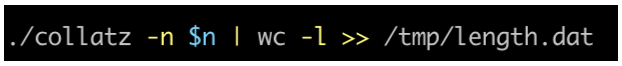
The following is the bash script used to generate the two graphs. Firstly it is important to point out that this code runs in a for loop which holds the integer *n* that symbolizes the current loop value - ranging from 2 → 10000
ATTRIBUTION: part of the code below was taken from Eugene Chou’s session



```
for n in {2...10000}; do

        ./collatz -n $n | wc -l >> /tmp/length.dat
done
```

Secondly, there is the script that pushes the data generated from the Collatz sequence into a length.dat file:
ATTRIBUTION: part of the code below was taken from Eugene Chou’s session



```
./collatz -n $n | wc -l >> /tmp/length.dat
```

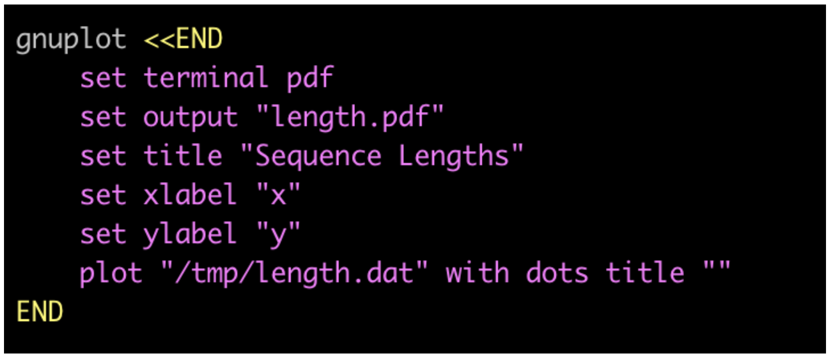
I will explain this code by parts below:

./collatz - n $n → generates the collatz sequence, with the deterministic starting point 'n'
wc -l → takes the output of the collatz sequence and return the 'newline' counts
>> /tmp/length.dat → takes the output from the previous command and puts it into a length.dat file, which is stored in the temporary storage path (.../tmp)

Point Graph

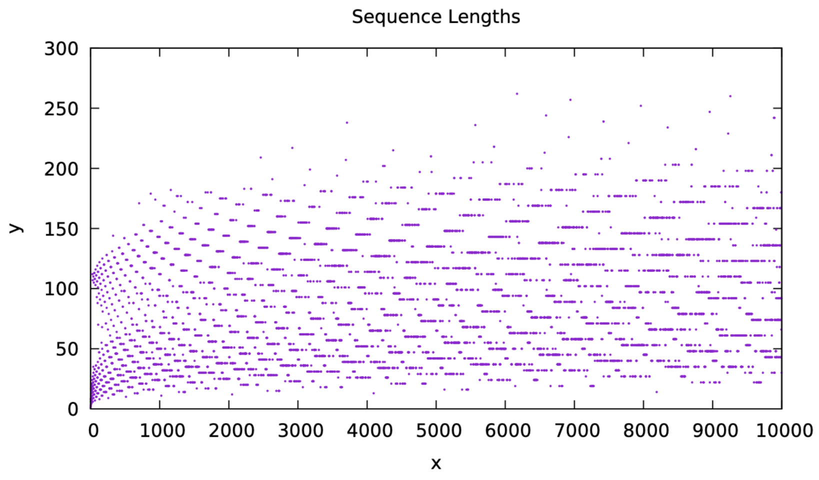
For the first part, I have used *gnuplot* to generate a graph that will be able to take the values inputted into the length.dat file and output the graph below, which follow a specific pattern -

Below are the *gnuplot* commands to generate this graph - they are formatted and therefore they are quite easily understandable by an English reader:
ATTRIBUTION: part of the code below was taken from Eugene Chou's session



```
gnuplot <<END
    set terminal pdf
    set output "length.pdf"
    set title "Sequence Lengths"
    set xlabel "x"
    set ylabel "y"
    plot "/tmp/length.dat" with dots title ""
END
```

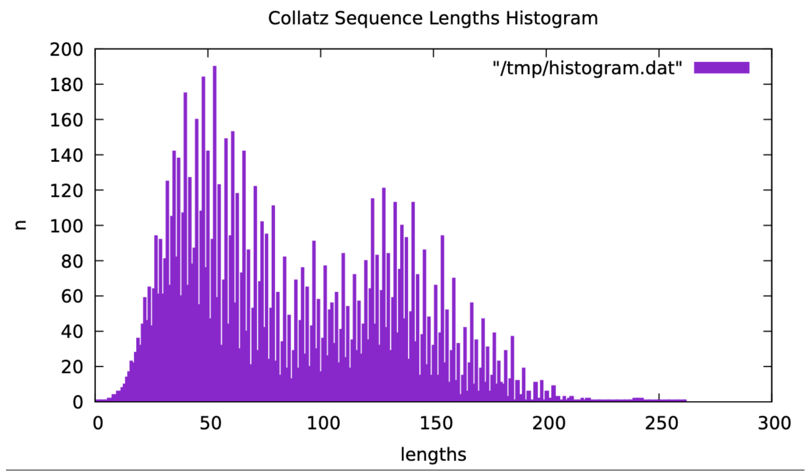
The graph below shows the different sequence lengths *'y'* for each iteration *'x'*. I have decided to use this naming convention not to be confused with the histogram below. We can see that x - being the starting point - follows a specific pattern that determines how many iterations the sequence will produce, which resembles a widened logarithmic function.



Histogram

NOTE: For the histogram, I have used the same data, however, I stored it in the *histogram.dat* file

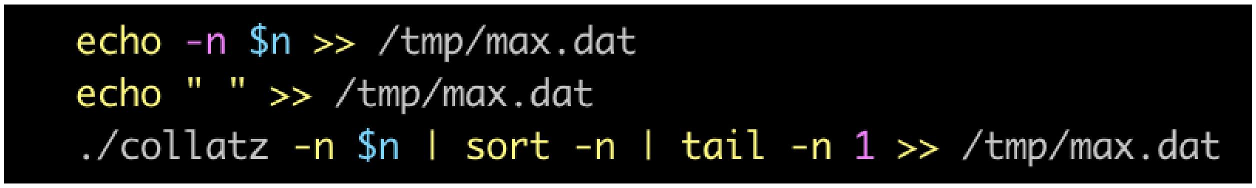
The Histogram below merges the number of times ‘n’ a certain length was in the data file, therefore it can be observed which sequence lengths are more common than others.



Collatz Maximum Value Graph:

Bash Script

Below is the bash script used for data collection from the collatz sequence. In this specific example, I sorted the values of the sequence numerically- and collected only the maximum value that occurs in the sequence, which of course does not have to be the starting point.
NOTE: this script is encapsulated in a for loop, such as the one described above
ATTRIBUTION: part of the code below was taken from Eugene Chou’s session

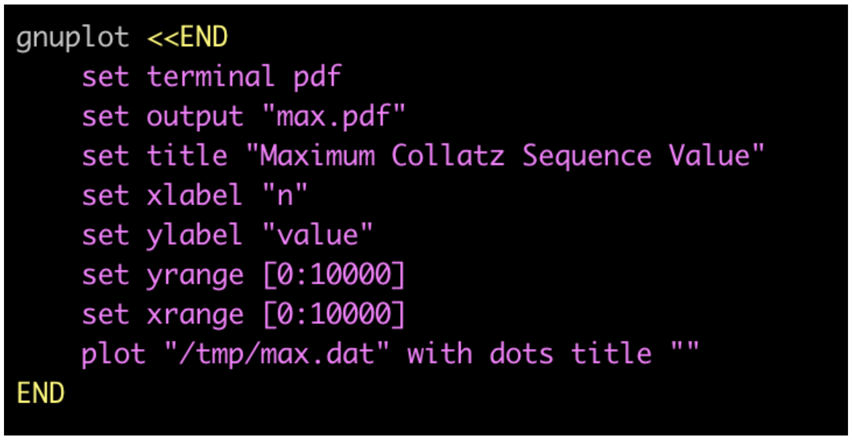


```
    echo -n $n >> /tmp/max.dat
    echo " " >> /tmp/max.dat
    ./collatz -n $n | sort -n | tail -n 1 >> /tmp/max.dat
```

Here I will explain how my bash script works by parts:
echo -n $n → adds the ‘n’th number of the sequence to max.dat document
echo -n “ ” → adds a space to the document to separate the line number with the data
./collatz -n $n → generates the collatz sequence, with the deterministic starting point ‘n’

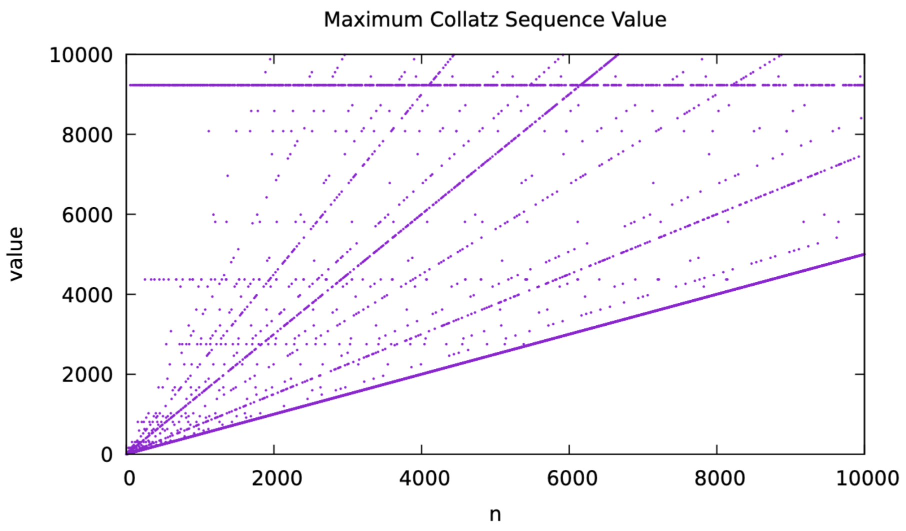
sort -n → sorts all the output from the previous command, by numerical order, so from the smallest to the largest value
tail -n 1 → returns the first value from the tail(therefore the bottom most values, which will allow me to grab the max value) from the sequence
>> /tmp/max.dat → takes the output from the previous command and puts it into a max.dat file, which is stored in the temporary storage path (.../tmp)

The command below informs *gnuplot* to create a point graph from the data explained above, with the x and y range being 0-10000.



```
gnuplot <<END
    set terminal pdf
    set output "max.pdf"
    set title "Maximum Collatz Sequence Value"
    set xlabel "n"
    set ylabel "value"
    set yrange [0:10000]
    set xrange [0:10000]
    plot "/tmp/max.dat" with dots title ""
END
```

This graph represents the maximum value in a sequence, shown in a particular pattern. This pattern shows that the data has a specific concordance to a sequence as we can observe the linear relationship between the starting point ‘n’ and the maximum value of a sequence ‘value’.



Conclusion:

From the data shown I have made the following conclusions

1. The length of collatz sequence plotted on a graph with a linearly increasing starting point, produces a pseudo-logarithmic sequence, meaning that its points are spread on the graph
2. Some lengths of the collatz sequence are much more common than others - lengths of around 50 or 130
3. The maximum collatz sequence plotted on a graph with a linearly increasing starting point, produces a pseudo-linear sequence, meaning that its points are spread on the graph