Assignment 1 - DESIGN.pdf

Description of Program:
Plots 3 graphs to help study/understand the Collatz sequence starting from a random integer n.

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

Directory Files:

- plot.sh - Bash script that produces 3 graphs
- collatz.c - A provided file that outputs a Collatz sequence from a random starting number
- Makefile - A provided file that complies Collatz file
- README.md - In Markdown, describes how to use script and Makefile
- DESIGN.pdf - This, logic behind the program
- WRITEUP.pdf - PDF containing UNIX commands used to produce each plot and why they were chosen

Graph 1: Example From Assignment Sheet

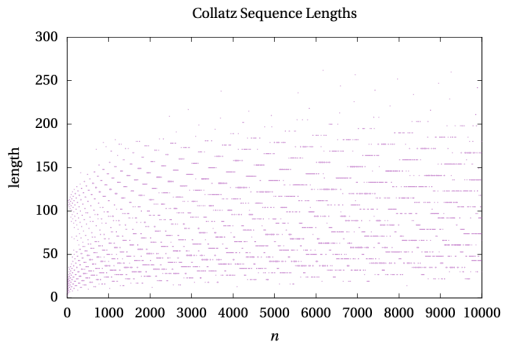


Figure 2: The length of Collatz sequences starting from $n \in \{2, \ldots, 10000\}$.

- Each point on the graph is 1 run-through of the Collatz. Each number has a linebreak
- Sequence length is equal to the number of lines in the output
- Create an output file (lengths.dat) “n length” and plot it from n{2..10000}

Trial Error 1: Randomness is based on time and running it multiple times will result in repeating the same value.
Resolution: Put sleep 1 between each run of collatz

Trial Error 2: A jumbled mess with no obvious pattern

.collatz, example results (not actually following equation)
run n = 2
1, 2, 5, 6, 8, 1 > wc -l => len = 6
run n = 3
6, 5, 7, 9, 0, > wc -l => len = 5
…

.dat
2 6
3 5
…

Pseudocode Graph 1:
for (int n, 2 to 10000):

l = length(./collatz)
“n l” put into > length.dat

gnuplot < plot “lengths.dat”

Graph 2: Example From Assignment Sheet

- Sort output of collin descending order and take only the first output for each run. Store it in
- Maximum is numerical sort in descending order, the first line
- Create an output (maximum.dat) file “n max” and plot it from n{2..10000}

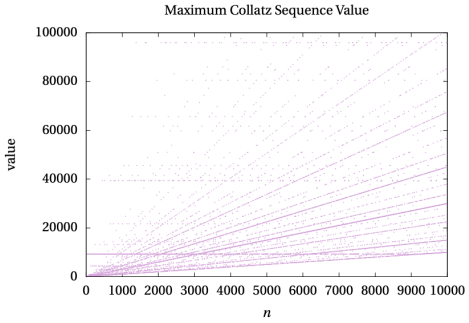


Figure 3: The maximum values for Collatz sequences starting from $n \in \{2, \ldots, 10000\}$.

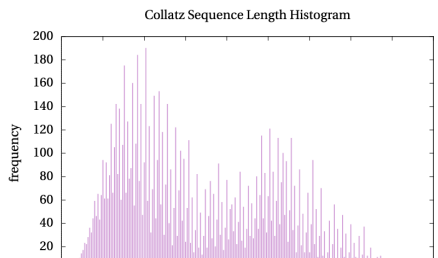
.collatz, example results (not actually following equation)
run n = 2
1, 2, 5, 6, 8, 1 > sort -r
8, 6, 5, 2, 1, 1 > head -1 => max = 8
run n = 3
6, 5, 7, 9, 0, 0 > sort -r
9, 7, 6, 5, 0, 0 > head -1 => max = 9
…

.dat
2 8
3 9
…

Pseudocode Graph 2:
for (int n, 2 to 10000):
put “n m” into> “maximum.dat”
m = first line (sort reverse (./collatz))

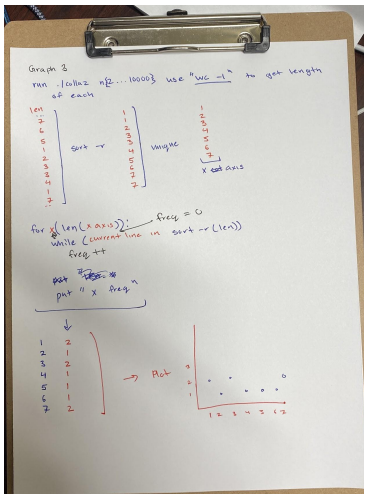
gnuplot
#Set up
plot “maximum.dat”

Graph 3: Example From Assignment Sheet



- The minimum length sequence is 1 since the sequence ends at 1.

- Take the lengths and put them into a separate “len.dat” sort them in descending order. Use uniq
- Loop through the entire “lengths.dat” and count the repetitions of each number in “len.dat”
- Create an output file “frequency.dat” format it in “len(x) freq(y)”



Pseudocode Graph 3:

```
for (int n, 2 to 10000):
        “values.dat” sort(./collatz)
        “len.dat” = unique(sort(./collatz))
counter = 1
for (int x, 1 to length(“len.dat”)):
        freq = 0
        while(“values.dat” [counter]):
                freq++
                counter++
        put “x freq” into> “histogram.dat”
gnuplot <plot “histogram.dat”
```