How to Plot Collatz Sequences Using gnuplot on Ubuntu 22.04

James Contini
10.2.22

The first step to plot these Collatz sequences is to set the script to interpret from bin/bash. Once this has been done, using the included Makefile, one can type "make clean && make collatz" to clear the local directory of unnecessary files and generate a new executable compiled from collatz.c.

After setting up the necessary initializations, we can now delve into the actual data computation and organization. To plot Figure 2 of assignment one, first create a .dat file named something suitable like collatz_2.dat. This file will house the coordinates for gnuplot to plot with. Next write a for loop with an in clause from {2..10000}. Inside the loop, run the collatz executable, with option -n and use the for loop's element variable as the option's argument so the executable computes the collatz sequence with that starting number. Then we can pipe the output to the command wc with option -l to count the number of lines of the data stream. This number will then get appended to the .dat file we created earlier. The amount of lines per collatz execution reveals the number of collatz computations it took a number to reach one. Now that our .dat file is populated and a newline is automatically added after each number, and because it is in order, we can feed this file to gnuplot.

Lastly, create a heredoc which you append to gnuplot. In the heredoc, set the y range from zero to three hundred and the x range from zero to ten thousand. Set the terminal to pdf, set the output to some_file_name.pdf and lastly plot the aforementioned .dat file with dots and no title.

To plot Figure 3 of assignment one, first initialize another .dat file with a suitable name like collatz_3.dat. Next, make a for loop with an in clause from {2..10000} and execute collatz with option -n with the for loop element variable as the argument. Then we can pipe this output to the command sort with the option -n to sort numerically. And then we can pipe output again to another command "tail" which we can use to take the last line of the file and append to our .dat file. In essence this sorts every collatz sequence from least to greatest and takes that greatest value and appends it to our data file in order. Now that our .dat file is populated we can feed this file to gnuplot.

Lastly, create a heredoc which you append to gnuplot. In the heredoc, set the y range to zero to one hundred thousand and the x range from zero to ten thousand. Set the terminal to pdf, set the output to some_file_name.pdf and lastly plot the aforementioned .dat file with dots and no title.

To plot Figure 4 of assignment one, first initialize another .dat file with a suitable name like collatz_4.dat with the string '2' in it without a trailing newline. Next sort Figure 2's coordinate data file such that it is least to greatest, and then pipe this to a tail command to get rid of the first line which is just a blank newline in collatz_2.dat. And finally this data will be redirected to another .dat file which you can call tmp_sort.dat. The last two things to initialize are two variables: counter and previous_line, counter should be set to 0 (our counter) and the previous_line to 2 (this variable will be used to remember the last loop element value). Finally to the computations:

Step one, create a while loop which reads the input lines from tmp_sort.dat. Then nest an if statement in it which checks whether the current element of the while loop equals the previous iteration's element, previous_loop. This will help the script know when to switch lines and append values. However if the previous element does equal the previous then increment the counter, else when the loop finds a new number, append a space to collatz_4.dat (no newline), append the counter's value and append the current element variable from the while loop which will be the new value. Also assign previous_line to the current loop element's value and assign the counter to one. In essence, the while loop will read through tmp_sort.dat which is just a vertical list of the number of computations it took every number to reach zero. But since it is numerically ordered that means the number of duplicates of a number *n* illustrates that it took that many numbers *n* computations to reach one.

Lastly, create a heredoc which you append to gnuplot. In the heredoc, set terminal to pdf, xrange to zero to two hundred and twenty five, y range from zero to two hundred, ytics to 20, xtics to 25 and plot collatz_4.dat with boxes and no title.

My last graph I made up, and answers the question: *For numbers in the range 2 - 10,000, does the average number of collatz computations differ depending on which digit is in the 10s place*?

The short answer is no, and this plot will prove it.

To plot this graph, initialize two variables to zero (these will be our counter and total), and a .dat file called collatz_cre.dat. Next, begin a for loop with an in clause for range {0..9}. Next, nest a while loop with a loop element variable which reads lines from collatz_2.dat. Inside the while loop check if the for loop element variable is equal to the second to last character of the while loop variable element. If it is, increment a counter and add the variable element from the while loop, which contains a number, to the total. When the while loop finishes, we re-enter the for loop, calculate the average by dividing our total by our counter and then append our for loop element and our average with a space in between to collatz_cre.dat. We do this for every starting digit, slowly appending their average number of computations and the starting digit itself to our .dat file until the for loop ends.

At last, create a heredoc which you append to gnuplot. In the heredoc, set the y range from zero to one hundred and fifty, the x range from negative one to ten, the xtics to one, the fill to totally opaque, the box width to .9 relative and finally plot collatz_cre.dat with boxes and no title.