# *CS 1632 - DELIVERABLE 3: PERFORMANCE TESTING*

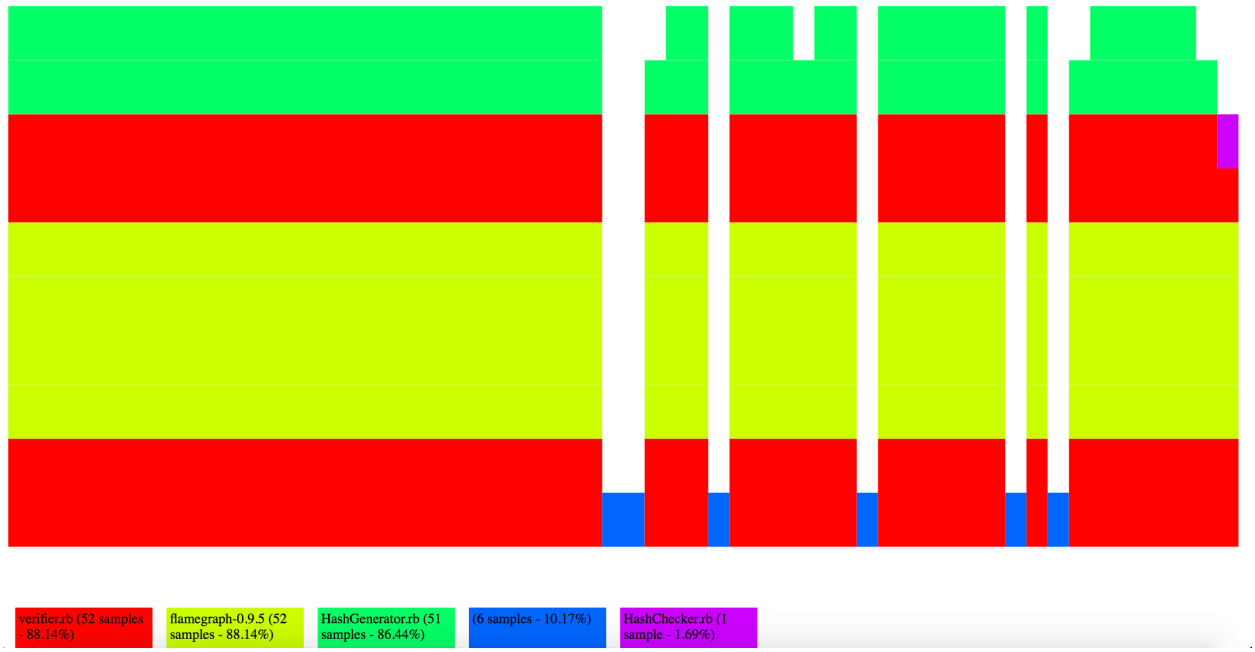
Link: https://github.com/Thonyvb/D3/tree/master
GitHub user: thonyvb

*George Anthony Villacres Botta*

# Summary

The most challenging part of the application was to find ways that the code will be both easy to test and really efficient. For instance, when generating a hash, I had to be able to parse the string in order to get its first four elements. In order to avoid, matching every single character or dividing each single string manually, I was able to take advantage of methods within ruby such as split with the use of regex, map, reduce, and shift that made the code faster to run. Likewise, I took advantage of Ruby's hash map in order to link the addresses with their respective number of billcoins. I also took into consideration the order of the cases that I was checking. It is easier and faster to first check if the order of the blocks is correct and then check if their hashes match because the matching takes considerable amount of time. Furthermore, I considered failures modes such as transactions including extra symbols and addresses taking bigger than 6 characters among others edge cases. I took precaution by checking if the amount of billcoins transferred was an integer, and I also checked if the file from where we are reading do exists. The flamegraph demonstrates that generating the hashes is really expensive computationally. Therefore, I considered several ways to make the computation less expensive such as avoiding nested for loops and relying on map and reduce ruby functions in order to generate the hashes. I also avoided generating the hashes more than one time. I stored all the hashes into an array, so I could reuse them. That improved the timing of my algorithm.

# Flamegraph




# Timing

Long.txt

real 0m35.474s
user 0m35.162s
sys 0m0.246s

real 0m35.752s
user 0m35.198s
sys 0m0.303s

real 0m36.704s
user 0m35.865s
sys 0m0.377s

mean of long.txt

real 0m35.976s