CMPE 365 Lab 1 Report:

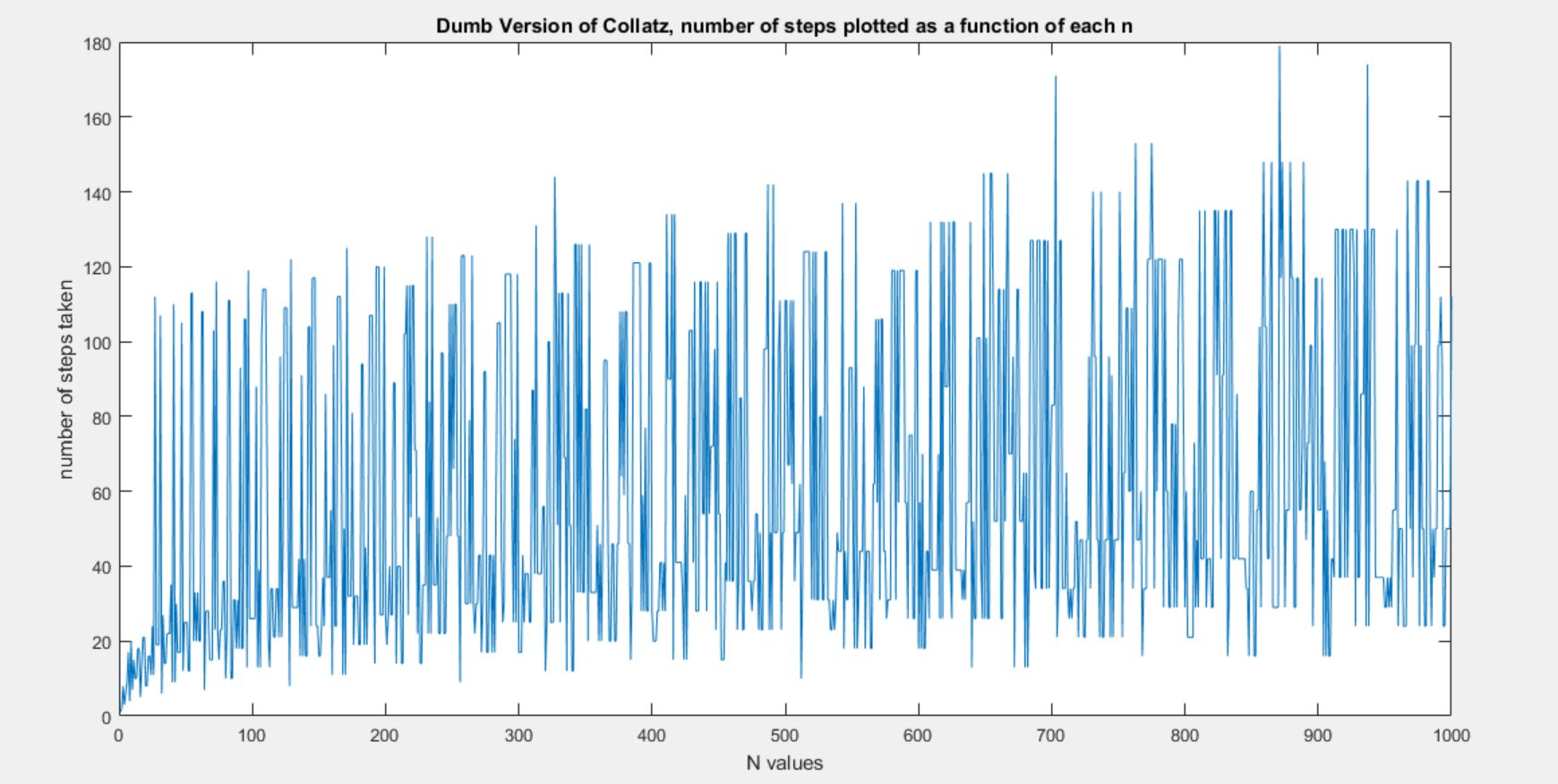
Paritosh Arya, 10143394

1.

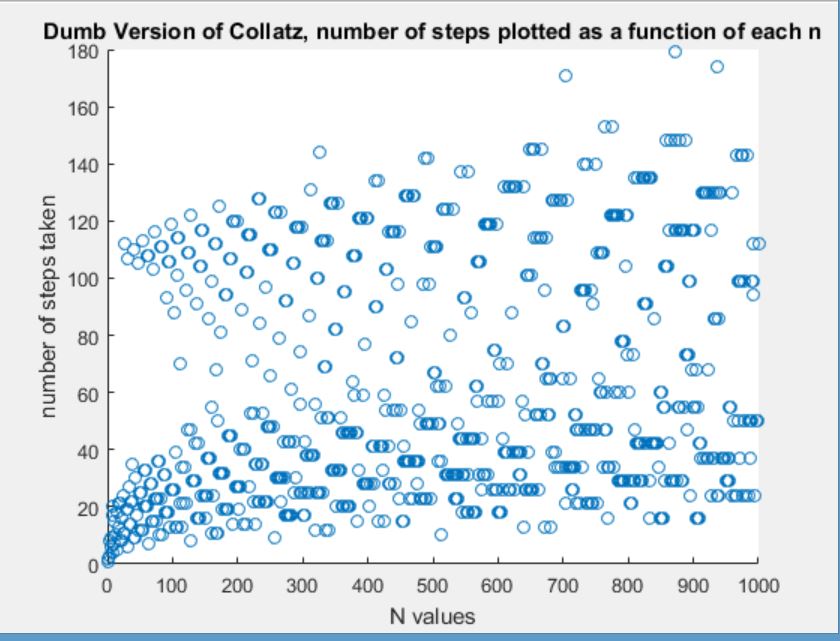
In the Dumb Collatz version of the algorithm, each value from 1 to n for Collatz’d individually in an isolated manner. After each n was run through, the number of steps were added to the overall count of number of steps take to prove that all numbers up to N were able to Collatz.

The function of steps taken to collatz each value was plotted as a function of each N.

The results show interesting features:



The plot() function shows a zig-zag pattern as some n-vales require much less steps to complete the Collatz programs. These values can be predicted to be numbers like 2,4,6,8,16,32,64….and so on. Values that regardless of their large size say for n =512 still collatz quicker.



The scatter() of the dumb Collatz shows a similar pattern as seen previously but now clear there is a logarithmic increase in steps taken.

The DumbVersion of my Collatz is ran from its Main(n) function on the command line, each value up to n is Collatzed individually and steps taken are added up to the total count. Then that information collected is discarded for the next higher ith value.

The Dynamic Collatz version differs because it saves the amount of steps taken to collatz each it value in a container map. This container map is then checked through each iteration to check if that lower value has come up in the latest version of the collatz run. If it has, we can conclude the program will converge after the amount of steps stored as the value of the specific key in the container.

To test this, lets Collatz(15) for both versions:

%For Collatz(15)

%Dynamic Progamming Results: Startnumber: 15 ,number of steps taken: 21 , time: 3.742143e-03 seconds

%Dumb Collatz Result Startnumber: 15,number of steps taken:148 time: start0.0023

The visualization of the Dynamic Program would look like:

%n=15 collatz, start from bottom at value 1:

%15 , 46, 23, 70,35,106 , 53 , 160, 80, 40,stop

% 14,STOP

% 13, STOP

%12 , 6 STOP

% 11 STOP

%10 STOP

%9,28,14,7 STOP

% 8 STOP

% 7,22,11,34,17,52,26,13,40,20,10 STOP

% 6,3 STOP

% 5 STOP

% 4 STOP

% 3,10,5,16,8,4,2 STOP

% 2,1 STOP

% 1 STOP

Starting from value = 1 up to 15, we can see where each comma represents an iteration of Collatz or a step taken. Ultimately adding up to 21 which is much less than the dumb implementation of 148 steps

Applying this same methodology to n =1000, we get the following results:

%%For Collatz(1000)

%DynamicProgamming Results: Startnumber: 1000 ,number of steps taken: 180 ,

%Dumb Collatz Result Startnumber: 1000,number of steps taken:60542

So we can see that the number of steps take to Collatz up to 1000 numbers takes on 180 steps with the smarter algorithm using dynamic programming.

2. Using an outer loop would decrease the amount of steps taken to Collatz all values up to LARGE n. I can see a dynamic programming model being the most helpful in this scenario.

3. We are not even sure that this process terminates for all inputs. It's possible that the sequence always terminates, and it's quite possible that there are inputs for which the sequence never terminates.

Power-of-2 numbers seem to be a pretty good case scenario of but there could be other worst-case scenarios of n values that would not converge.

However at this time, there is no sensible upper bound.

4.

Referring to the scatter plot of the Dumb Collatz implementation, there’s is defenetley a pattern on the lower half of the plot that converges much rapidly than the plot points on the upper half. These are the power-of-2 n values.

As for the upper half, a trend line can be projected to show that the majority lie around of ‘number of steps taken’ =120. This ~120 value could be used as an upper bound in certain situation in reference to question 3.