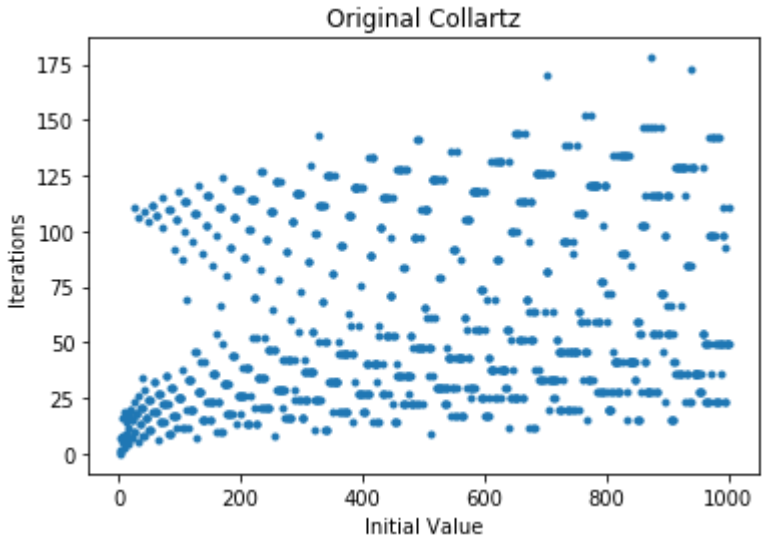
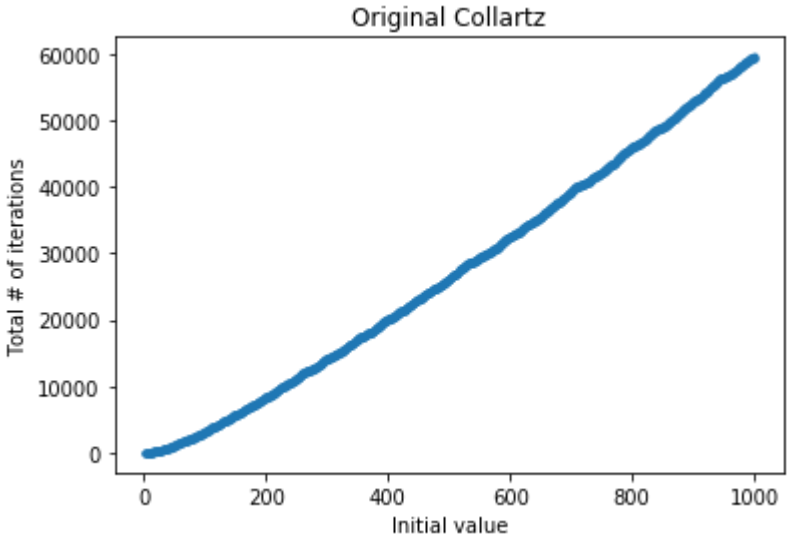
CMPE 365 - Lab 1

Max Gillham - 10183941 - 14mfg2

Decreasing Iterations by Checking Previous Values

Consider the original collartz problem in its basic form. Now, let's add an additional loop iterating the initial value from 1 to 1000 and observing the amount of iterations taken at each initial value. Let's also observe the accumulative number of iterations at each initial value from 1 to 1000.

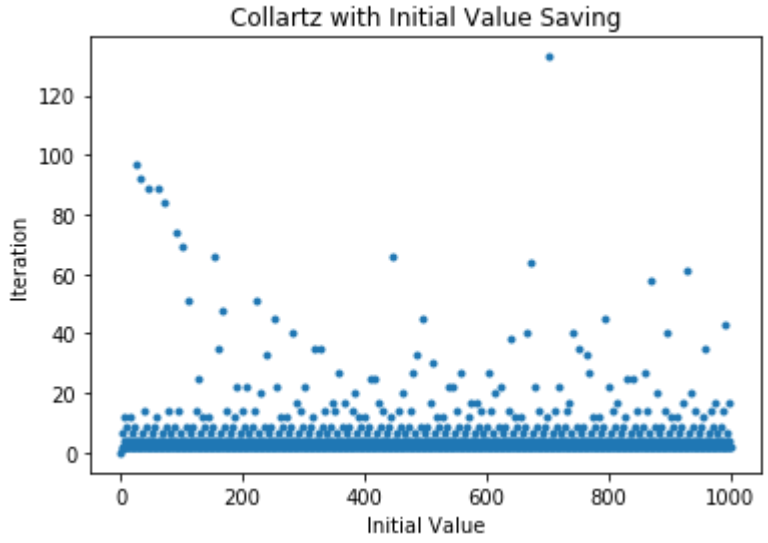


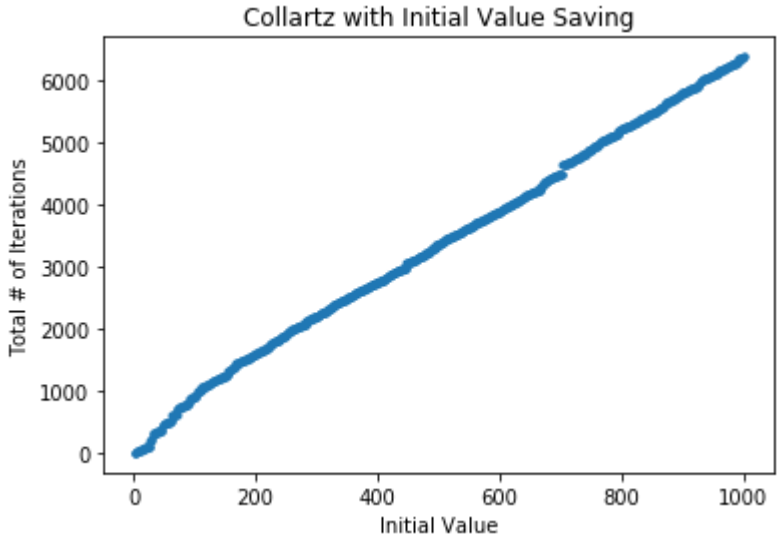


In the original form, using this outer loop from 1 to 1000 takes 59542 steps to terminate the entire process. As the initial value is incremented from 1 to 1000 and passed to collartz(x), the value of x fluctuates within the algorithm. In turn, the number of iterations to terminate for each given initial

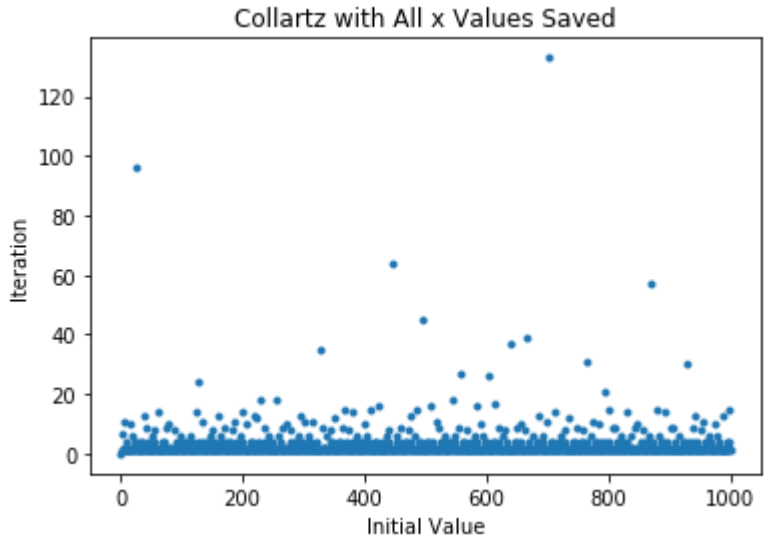
value is not very intuitive. The curve for the original collartz in this nature appears to increase exponentially.

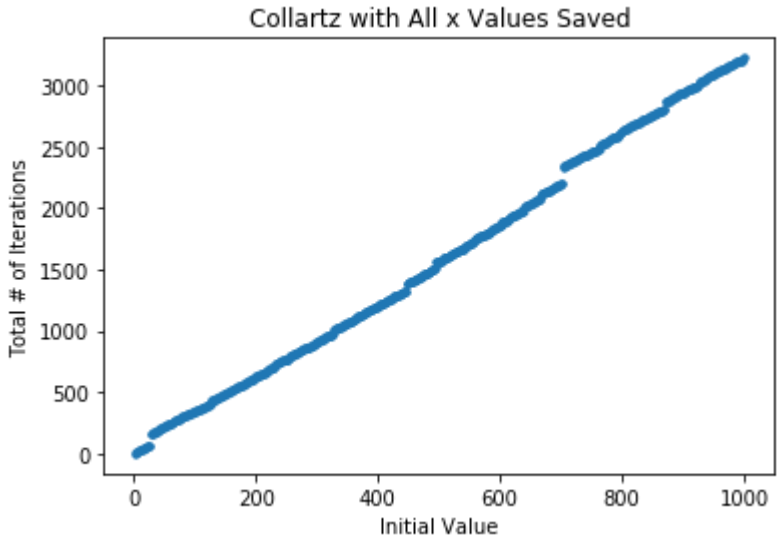
To reduce iteration steps, lets save the initial values where the algorithm terminates, and check hits these values when inside the algorithm.





Total number of iterations has not decreased to 6377. The graph now appears to have more linear behaviour. However, the value of x often will fluctuate above the initial value, which means the algorithm is often checking numbers more than ones which is increasing the iteration count. Let's observe what happens when we flag every value of x that is calculated in the algorithm and consider this a termination point for future initial values.

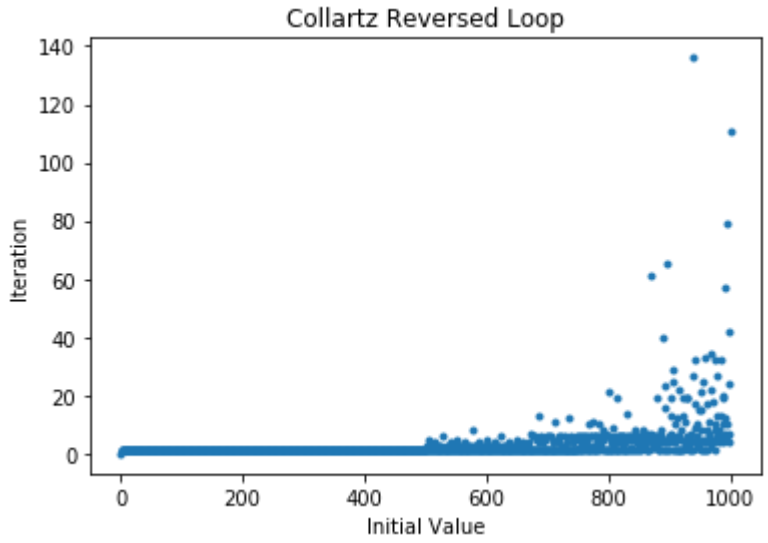


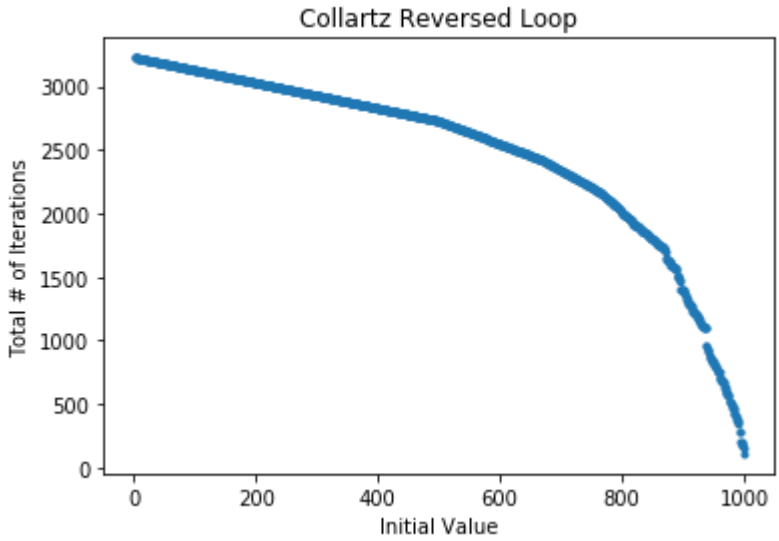


Now, checking values 1 to 1000 only takes 3225 steps, a major improvment from the original algorithm.

Order of the Outer Loop

Let's see if the order of the outer loop has an effect on the total number of iterations. From reversing the order, so 1000 down to 1, the following grpahs are produced.

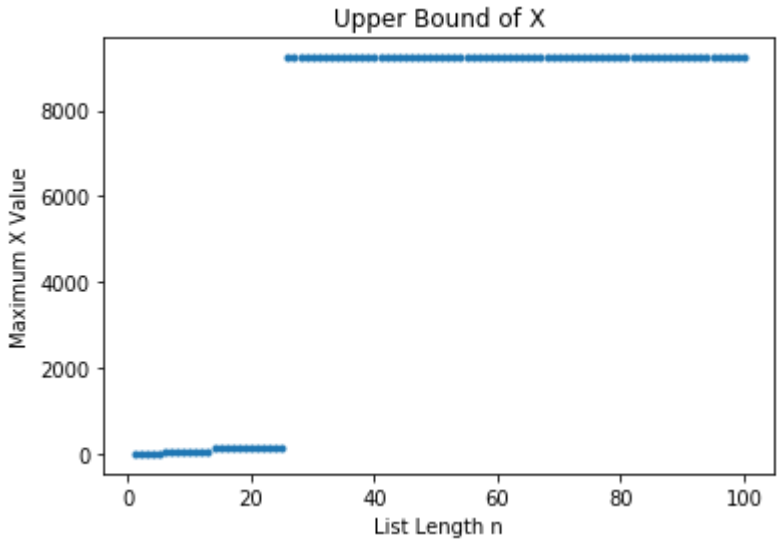




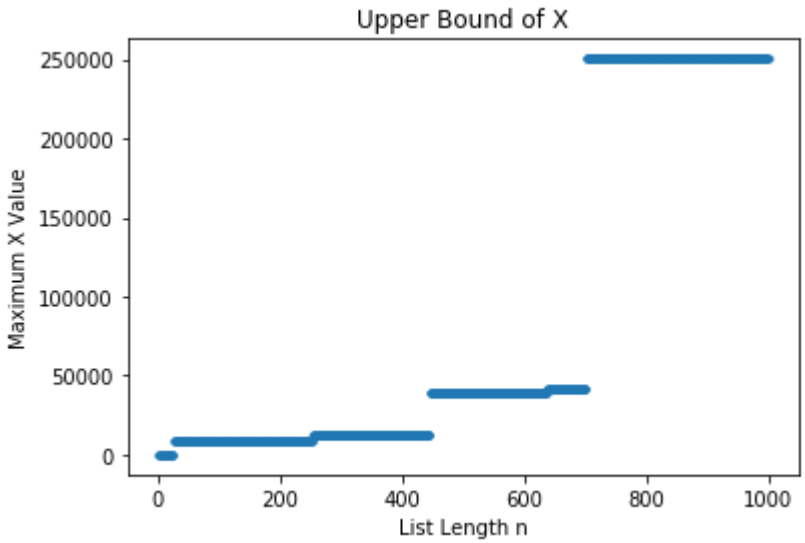
The total number of iterations remains constant at 3225. The shape of the graph changes, as the iterations by initial value are concentrated at the start of the loop, now being 1000. Depspite this, the total iteration count remains the same as the values of x which the algorithm has to test is constant. Therefore, the order of the outerloop does not have an effect on the total number of iterations, as the possible values of x are consistent.

Upper Bound For x

When iterating the initial value from 1 to n, the algorithm checks values of x that are larger than n. Lets see if there is a maximum value of x. Below is a plot of values for n, where for a given n, the algorithm gets input 1 to n in the outer loop and returns the maximum value of x. So for n=10, the y axis denotes the largest n when the initial value is 1 to 10.



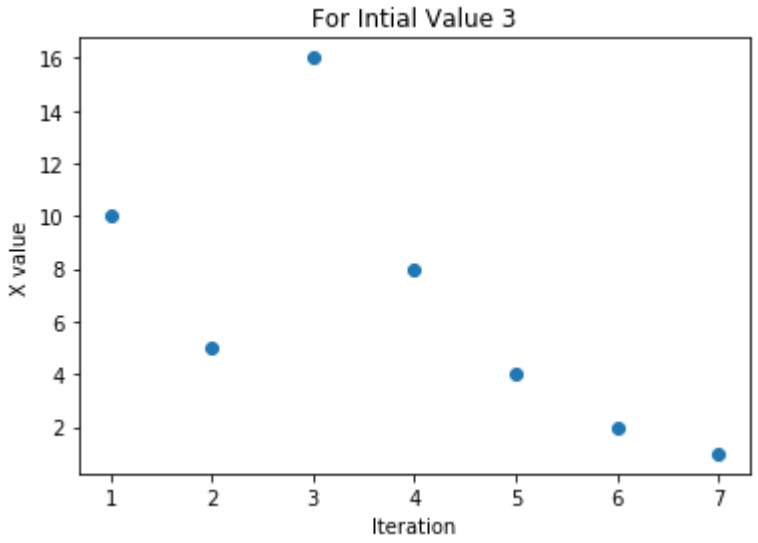
It appears the maximum value of x is 9232 when n is iterated from 1 to 100. However note that when n is larger, this upper bound changes and increases more.

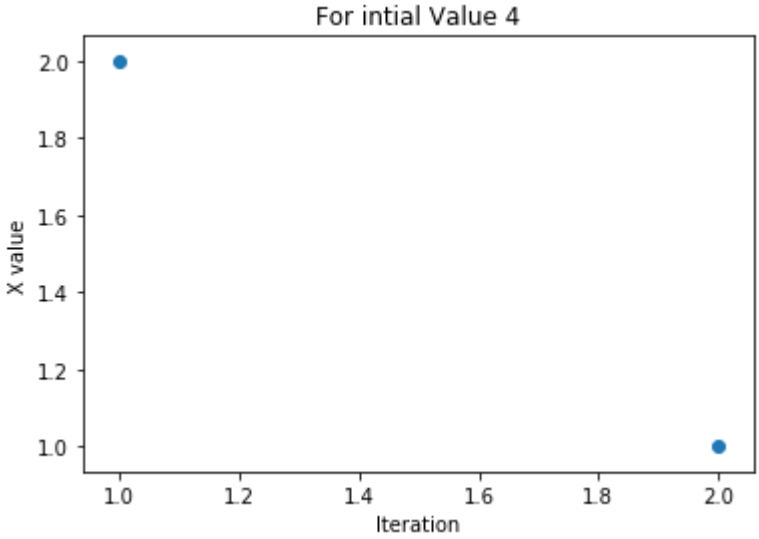


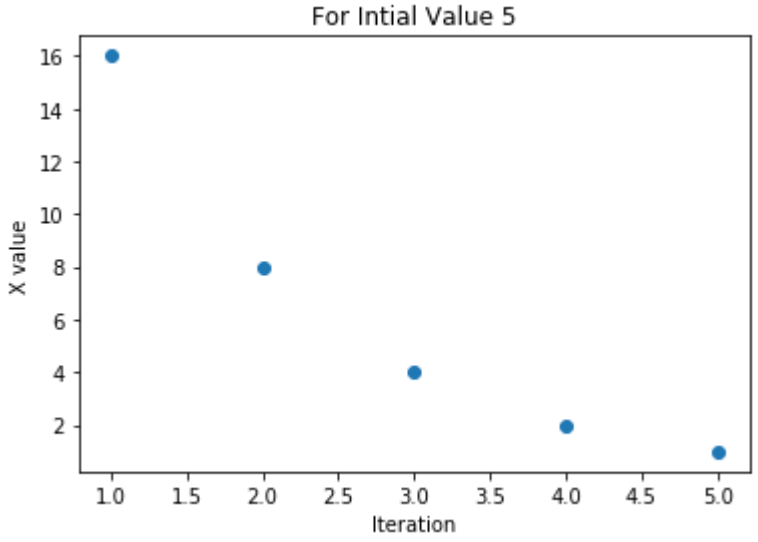
Therefore, there does not exist an upper bound for x.

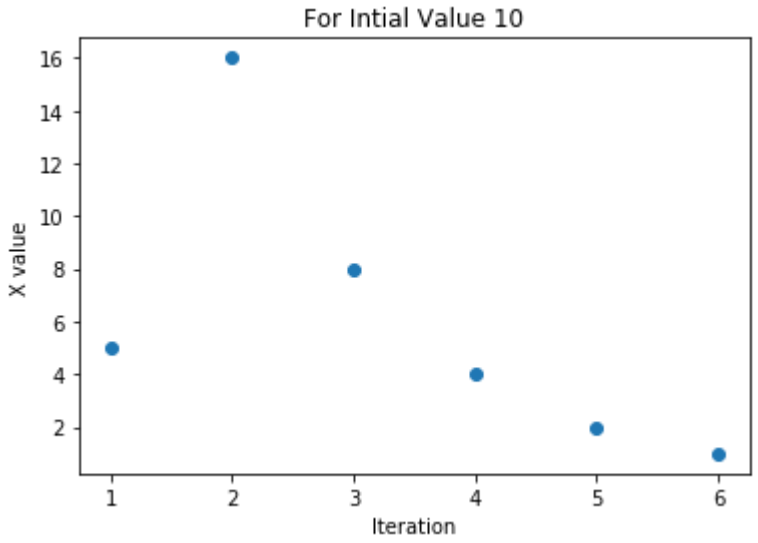
Convergence to 1

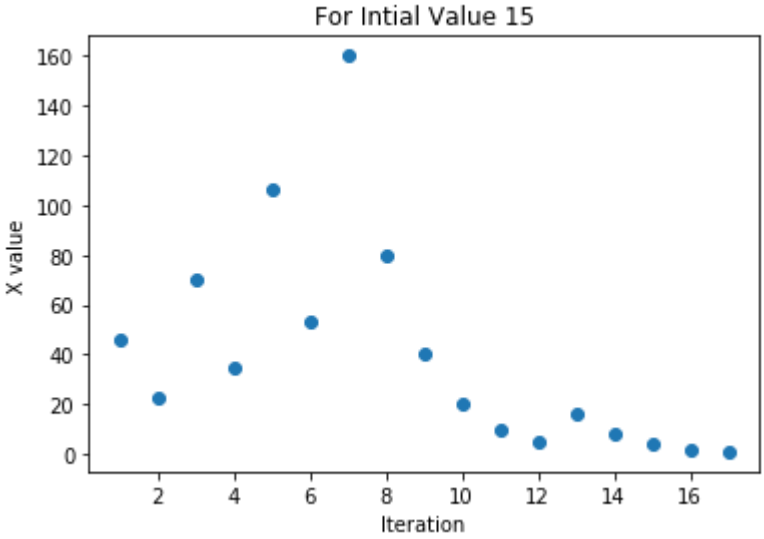
Considering the original collartz algorithm, in order for the algorithm to terminate, the value of x must equal 1. The convergence to 1 changes depending on waht intial value is given.


For Intial Value 3
X value
Iteration
16
14
12
10
8
6
4
2
1
2
3
4
5
6
7


For intial Value 4
X value
Iteration
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0


For Intial Value 5
X value
Iteration
16
14
12
10
8
6
4
2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0





As you can see, the convergence to 1 is not always uniform. Considering the input value 5, the values of x converge to 1 very uniformly, however, consider initial value 3, 10 or 15 where the value of x seems random, and then hits a power of 2 and begins to converge uniformly. Therefore, the termination of this algorithm for any given intial value is undetermined.