DAA Assignment 4 - Group 17
Finding the number of steps it takes to reduce a given number (n) to 1 on applying $n/2$ if it is even and $(3 * n + 1)$ if it is odd.

Pratham Prakash Gupta (IHM2017003), Priyal Gupta (IHM2017004), Aditya Gupta (IHM2017005), Rohit Ranjan (IHM2017001)

February 2, 2018

1 Abstract

Here, we have tried to design an algorithm to find the count of steps required to reduce a number (n) to 1 depending on the conditions given in the question. There is no specific time complexity of this question, but the best case time complexity comes out to be O($\log_2 m$), where $n = 2^m$.

2 Keywords

Hailstone sequence : A hailstone sequence is a sequence which converges to 1 on successively dividing it by 2 if the number is even and multiplying it by 3 and adding 1 to it if the number is odd.

3 Introduction and Literature Survey

The Hailstone sequence of numbers can be generated from a starting positive integer, n by:
1. If n is 1 then the sequence ends.
2. If n is even, then the next n of sequence is $n/2$.
3. If n is odd, then the next n of sequence is $(3 * n + 1)$

The (unproven) Collatz conjecture is that the hailstone sequence for any starting number always terminates. The conjecture is also known as $3 * n + 1$ conjecture.

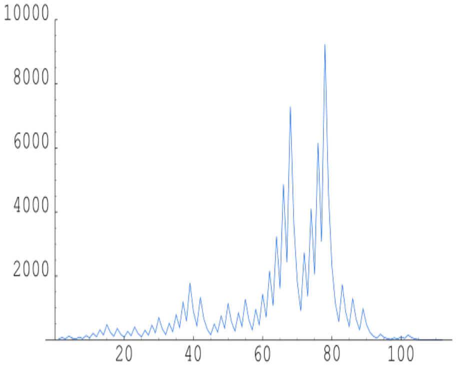
The hailstone sequence is also known as hailstone numbers (because the values are usually subject to multiple descents and ascents like hailstones in a cloud).
This sequence is also known as the Collatz sequence.

Example of a Hailstone sequence :
For n = 12, the sequence is 12, 6, 3, 10, 5, 16, 8, 4, 2, 1

4 Idea

We solved this problem using Recursion. We created a function - fun(n, count) which takes an input value n

Figure 1: Graph of Input number on x-axis vs Number to steps taken to reduce the number to 1 on y-axis.



and count. If n becomes equal to 1 at any steps, it return the count of steps till that time. Else if the number is even, it return fun(n/2, count) and if the number is odd, it returns fun(3*n+1, count).
The count of steps does not follows any specific series, as the number of times we divide or multiply the number is not fixed.

5 Algorithm Design

5.1 Parameters :

n: The given number which is to be reduced to be 1.
count: The number of steps required to reduce the given number to 1.

Algorithm 1
fun(n , count)

```
if n=1 then
    return count
else if n%2 = 0 then
    count = count +1
    return fun (n/2, count)
else
    count = count +1
    return fun (3*n + 1, count)
```

5.2 Pseudo Code :

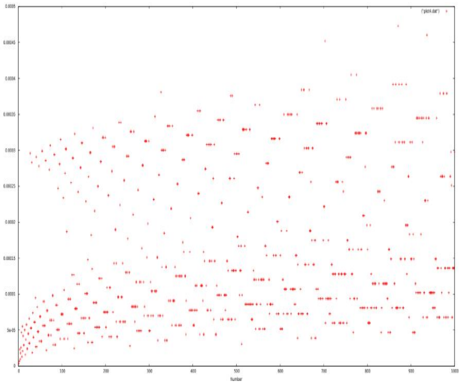
6 Analysis and Discussion

6.1 Time Complexity

Time complexity is the computational complexity that describes the amount of time it takes to run an algorithm. Time complexity is commonly estimated by counting the number of elementary operations performed by the algorithm, supposing that each elementary operation takes a fixed amount of time to perform.
Thus, the amount of time taken and the number of elementary operations performed by the algorithm are taken to differ by at most a constant factor. But in this case the time taken by the algorithm to complete is not fixed because depending on the value of n, the number of operations required to reduce it can be different.
So no general time complexity can be given for this algorithm.

Best Case - The best case occurs when the given number n is of the form 2 raised to power m. Then the time complexity in such a case is equal to O($\log_2 m$), where $n = 2^m$

Figure 2: Graph of Input number on x-axis vs Number to steps taken to reduce the number to 1 on y-axis as plotted on matplotlib



Worst Case - The worst case complexity can not be defined as it depends on the value of n.
But the worst case will occur when the number takes very large number of operations to be reduced to 1.

6.2 Space Complexity

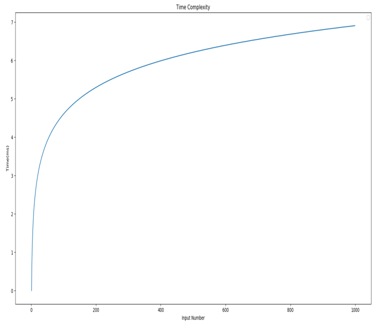
The space complexity of an algorithm is the maximum amount of space used at any one time, thus the space algorithm for this code is O(1).

7 Experimental Setup

Language Used :- C++
Plotting tool :- matplotlib
Report Making Tool :- TextPortable(Latex)

8 Conclusion

Figure 3: Time complexity for best case scenario.



Though it is very easy to write a function to converge the number to 1, it is very complex to find the time complexity as the number of steps for any n is not defined.
We has successfully calculated the number of steps it takes for any n to get reduced to 1.

9 References

1. https://en.wikipedia.org/wiki/Collatz$_conjecture$
2. https://en.wikipedia.org/wiki/File:Collatz5.svg
3. https://rosettacode.org/wiki/Hailstone$_sequence$