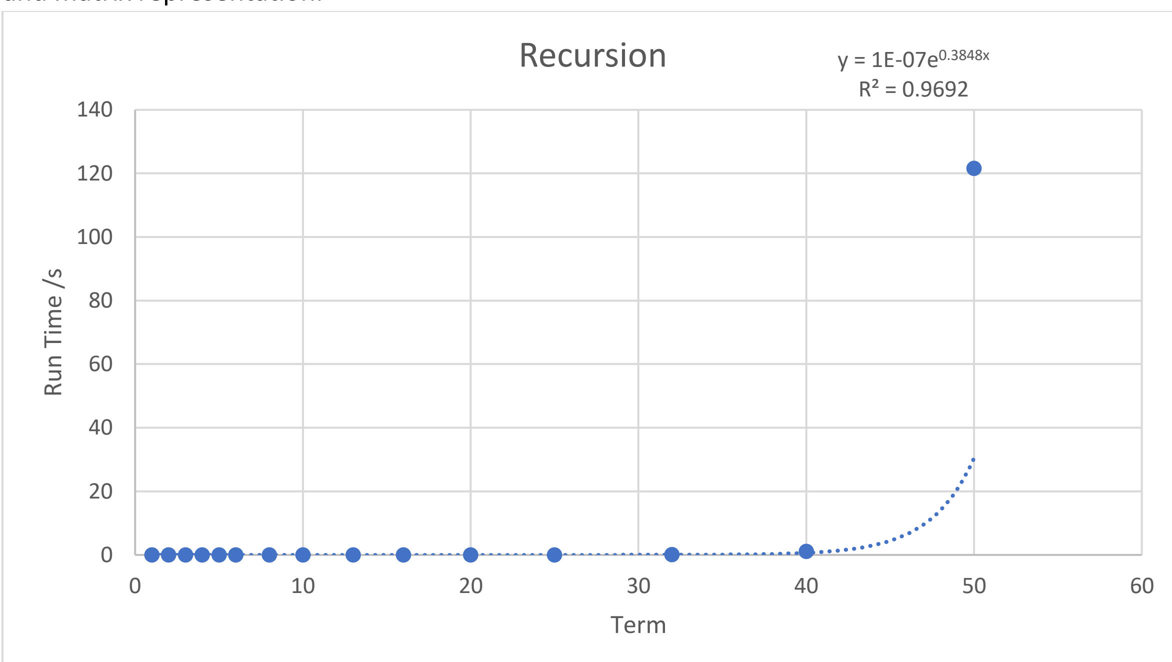
Problem 1: Fibonacci Numbers

a) Find attached C file.
b) Find attached spreadsheet.
c) For the same n all methods apart from the closed method give the same Fibonacci number. For recursion, bottom up and matrix representation base cases are defined and the n^{th} term is computed using these base cases. Whereas, closed form uses a value

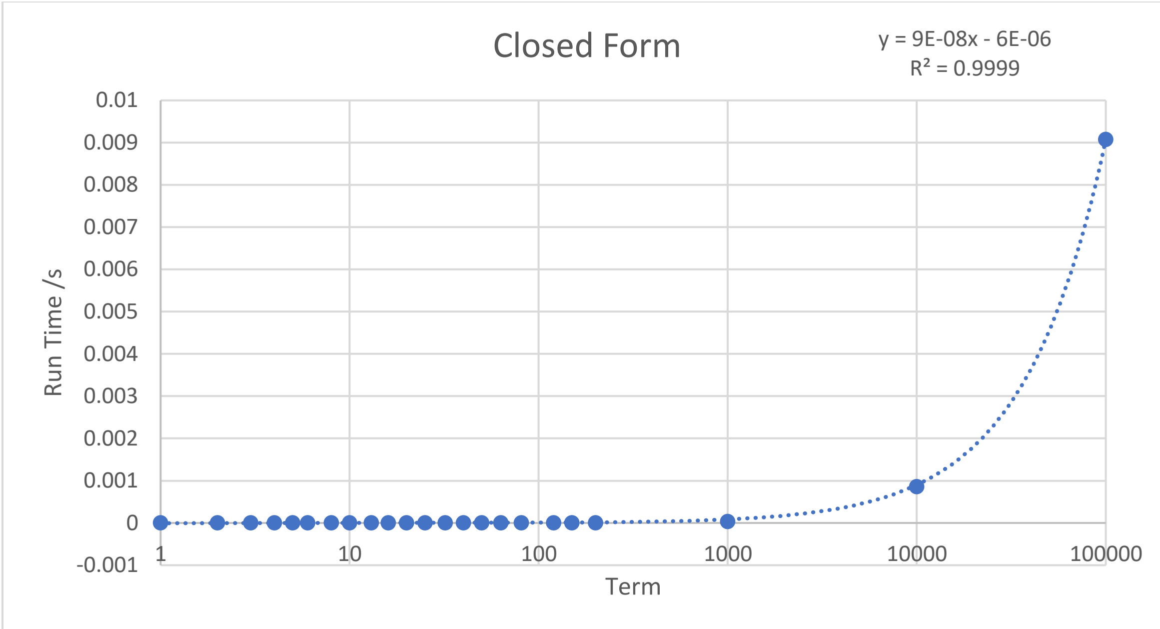
$$\Phi = \frac{1+\sqrt{5}}{2}$$

This value produces good approximations of n up until a very large n.

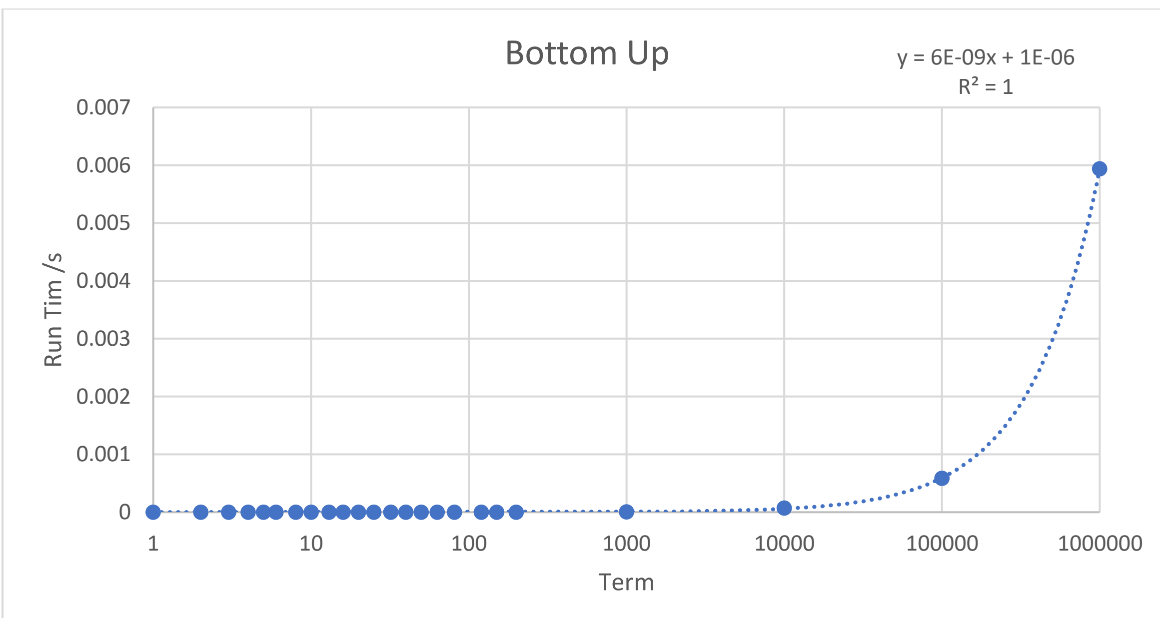
d) Below are the charts for the respective algorithms. After the 1 million terms the program received a segmentation fault. Therefore, n only goes until 1 million for bottom up, closed form and matrix representation.



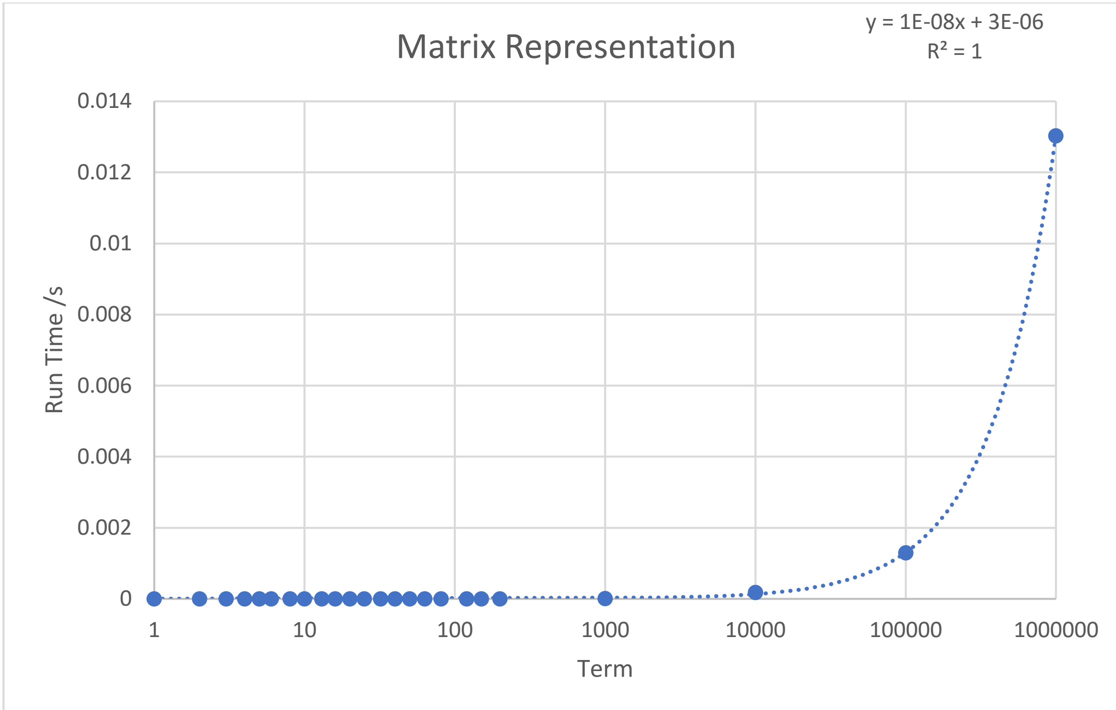
As shown from the beset fit trendline, the recursion method has exponential computation time.
$f(n) = 1 \times 10^{-7} e^{0.3848n}$



As shown above, closed form has linear time.
$f(n) = 9 \times 10^{-8} n - 6 \times 10^{-6}$.



Bottom also demonstrates linear computation time.
$f(n) = 6 \times 10^{-9} n + 1 \times 10^{-6}$.



The results for matrix representation here yielded linear time.
$f(n) = 1 \times 10^{-8}n + 3 \times 10^{-6}$.

Problem 2

a) ~~For~~ brute-force implementation of multiplication of two integers a & b with size n (bits).

The brute-force method compares & computes the product of every bit.
∴ the number of comparisons is $n \times n$
giving a time complexity of $O(n^2)$.

b) Using Karatsuba's Algorithm:

diving the main problem into ~~act~~ smaller subproblems.

$$\left.\begin{aligned} \text{let } x &= 10^{n/2}a + b \\ y &= 10^{n/2}c + d \end{aligned}\right\} \text{subproblem}$$

$$xy = 10^{n} \times ac + 10^{n/2}(ad + bc) + bd$$

Assuming n is a power of 2, we divide the integers in half (a, b, c, d) & use the formula to calculate the product without using brute-force.

i.e. let $x = 7539$; $y = 4899$

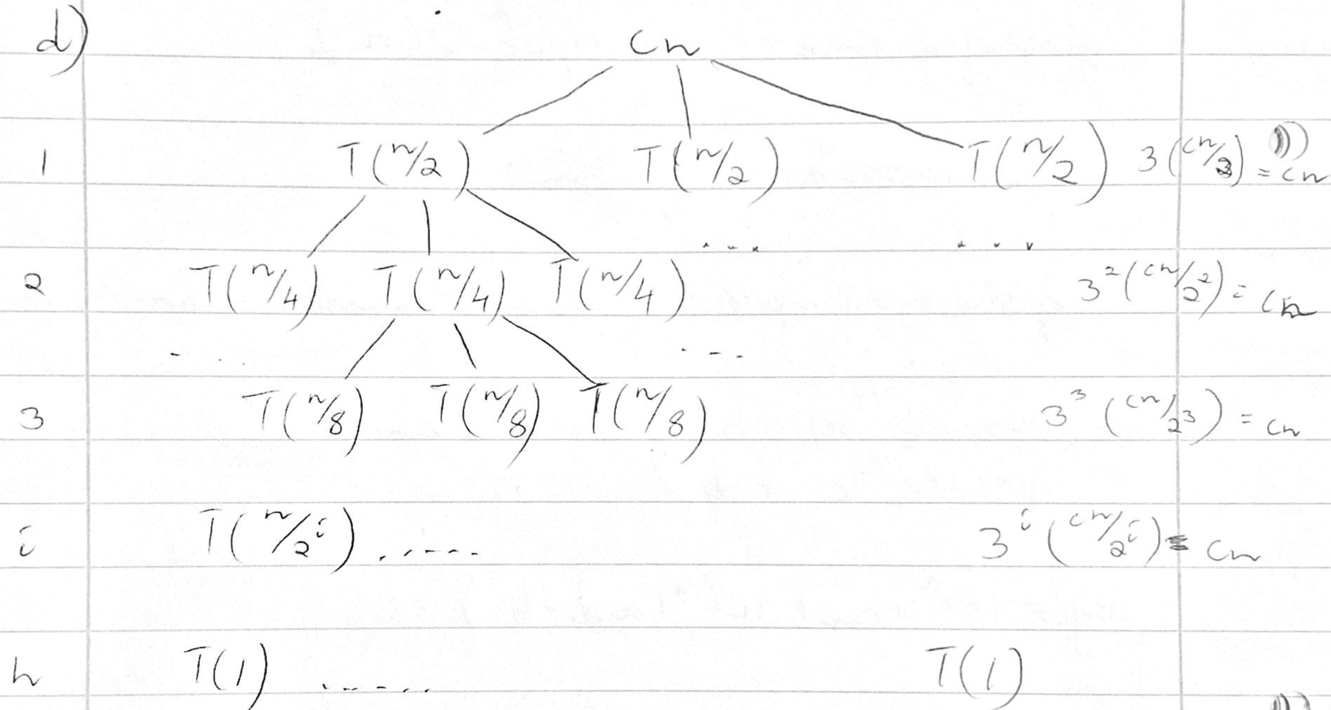
$a = 75$, $b = 39$, $c = 48$, $d = 99$

c) Let $n = 2^h$ for some value of h. The algorithm recurses 3 times on the $n/2$ bit number.
For addition & subtraction, we have $O(n)$

$\therefore \quad T(n) = 3T(n/2) + O(n)$

$\therefore \quad \Theta(n^{\log_2 3})$

d)



at $h \Rightarrow n/2^h = 1 \Rightarrow 2^h = n \Rightarrow h = \log_2 n$

$$T(n) = 2^h \times T(1) + \sum_{i=0}^{h-1} cn$$

$$\begin{aligned} T(n) &= 3^{\log_2 n} T(1) + cn\log_2 n \\ &= n^{\log_2 3} T(1) + cn \log_2 n \\ &= \Theta(n^{\log_2 3}) \end{aligned}$$

~~d)~~ 7

e) $T(n) = 3T(n/2) + O(n)$

$a = 3, \ b = 2$

$\therefore \ \Theta(n^{\log_2 3})$