Homework 5

Problem 5.1

a) All four methods have been implemented and are present inside the file "Fibanocii.cpp" and are present as seperate functions.

b) If I have increased the value of ~~n or and~~ n to over time and put the time limit to be 1.0 sec. The table is present as the "Table.pdf" file.

c) Other than closed form all methods return the same value of n as in closed form rounding error can be generated so for higher valos they dont come.

d) I have plotted down the graph and is present in the "plot.pdf" file.

Problem 5.2

a) Doing simple mathamatics. we can see that a brute force implementation will take $\Theta(n^2)$ to just multiply each element of Bit 1 to that of Bit 2. ~~and another~~ considering bitshifting will also take place for some value.

Bit → 1 55
Bit → 2 ×43
165
2200
+
$\Theta(n^2)$ 2365

and again another $\Theta(n^*)$ to add each element up to get the final output. So overall complexity would be $\Theta(n^{2*}) + \Theta(n^*)$ which would be $\Theta(n^2)$. So overall time complexity is $\Theta(n^2)$.

(b) Multiplying two large numbers.

Suppose $X = 1234$ then.

$X = 1234$

$= 12 * 10^{4/2} + 34$

$= 1200 + 34.$

$= 1234.$

From this we can derive that.

$X = a * 10^{n/2} + b$ where a = left half, b = right half and n = total digits.

So

~~Now for~~ ~~$X = 5678$~~ Now for another term

$= c * 10^n$ Y it would be $Y = c * 10^{n/2} + d$.

∴ Therefore $X * Y$

$= (a * 10^{n/2} + b)(e * 10^{n/2} + d)$

$\Rightarrow$ ~~$8ac$~~ which simplifies to

$= ac * 10^n + (ad + bc) * 10^{n/2} + bd.$

We will simplify (AD + BC) as it will be helpful later so.

$$(a+b)(c+d) - ac - bd$$
$$= ac + ad + bc + bd - ac - bd$$
$$= ad + bc.$$ useful to know.

Now implementing algorithm

```
float multiply (x, y)
n = max( bits in x, bits in y)
IF (n==1)
    return x * y
Else
  a = left half of x
  b = right half of x
  c = left half of y
  d = right half of y.

AC = multiply (a, c)
BD = multiply (b, d)
ADBC = multiply (a+b, c+d).

return (AC * 10^n + ((ADBC - AC - BD) * 10^(n/2)) + BD)
end.
```

c) Now the multiplication is just bit shifting in the formula so from that we can easily say our complexity is just

3. $T(n) = 3T(n/2) + \Theta(n)$. (divide & conquer)

e) According to master theorem,

$a = 3$

$b = 2$

So

~~$f(n) = \log n^{\log_2 3}$~~ $n^{\log_2 3}$

$= n^{1.58}$

Using case 1 of master theorem

~~$\Theta($~~ $f(n) = \Theta(n^{\log 1.58 - 0.58})$ for $\epsilon = 0.58$.

~~if~~

~~$\Theta(n)$~~ $= \Theta(n^{\log_2 3})$

$= \Theta(n^{1.58})$

which is better than brute force implementation of n^2.