Problem 5-1

=) a. The implimintation of the code can be seen in the Probs.cpp.

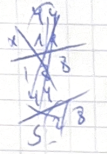
b. The value of n was increased 10 overtime and it can be seen in the Probstable-Pdf. and time was fixed at 1.0 second for each function.

c. Other methods return the same value of n but the closed form returns a different value for higher values due to rounding errors.

d. See the plot in as Ans5_Plot-Pdf

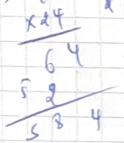
Problem 5-2

=) a.



bit 1 → 6

bit 2.



it can be seen that a brute force technique will take $\theta(n^2)$ to multiply each element of Bit 1 to that of Bit 2 without forgetting that bitshifting will also take place for some values. To add up each element would take $\theta(n)$ to add each element to for the final output. The final complexity would be $\theta(n^2) + \theta(n)$ which would be $\theta(n^2)$ since it is greater than $\theta(n)$. The total complexity is $\theta(n^2)$

b- Multiplying two large numbers.

=> Suppose $x = 4212$

=> $x = 42 \times 10^{4/2} + 12$

$= 4200 + 12$

$= 4212$

=> We can observe that:

$x = a \cdot 10^{n/2} + b$ where a = left half

b = right half

and n = total digits.

Considering the second term:

$y = c \cdot 10^{n/2} + d$

Therefore $x \cdot y =$

$(a \cdot 10^{n/2} + b) \cdot (c \cdot 10^{n/2} + d)$

$= a \cdot c \cdot 10^{n} + (ad + bc)\, 10^{n/2} + bd$

$=$

$(ad + bc)$ can also be represented as:

=> $(a+b)(c+d) - ac - bd$

=> $ac + ad + bc + bd - ac - bd$

$= ad + bc$

Th

The Algorithm

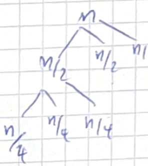
=)

```
Int Multiply (x, y)
    n = max (bits in x, bits in y)
        If (n == 1)
            return x 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 x 10^n + (Adbc - Ac - Bd) x 10^(n/2) ) + bd
```

c) Since the multiplication is just bit shifting in the formula. We can easily declare that our complexity is

$$T(n) = 3T(n/2) + \theta(n)$$

d.



$\rightarrow n$

$\rightarrow \frac{3}{2} n$

$\rightarrow \left(\frac{3}{2}\right)^2 n$

$\vdots$

$h = \log_2 n$

$$\sum_{h=0}^{h} \left(\frac{3}{2}\right)^h n$$

=>

$$\Rightarrow n \sum_{h=0}^{\log_2 n} \left(\frac{3}{2}\right)^h$$

$$n \left\{ \frac{1\left[1 - \left(\frac{3}{2}\right)^{\log_2 n}\right]}{1 - 3/2} \right.$$

$$\Rightarrow n \left\{ \frac{1 - n^{\log_2 3/2}}{-0.5} \right)$$

$$= n \left\{ \frac{n^{0.58} - 1}{0.5} \right]$$

$$\Rightarrow \frac{n^{1.58}}{0.5} - \frac{n}{0.5}$$

$$\Rightarrow 2n^{1.58} - 2n$$

$$\boxed{T(n) = O(n^{1.58})}$$

e. According to master theorem:

$a = 3$

$b = 2$

So $n^{\log_b a} = n^{\log_2 3}$

$= n^{1.58}$

$f(n) = \theta(n^{1.58 - 0.58})$ for $\epsilon = 0.58$

Case 3:

$\Rightarrow \theta(n^{\log_2 3})$

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

which is better than brute force implementation.

$\theta(n^2)$