CS202 Assignment 1.

Question 2

$x = x_L x_R = 2^{n/2} x_L + x_R$.
$y = y_L y_R = 2^{n/2} y_L + y_R$

Assumption: input binary numbers have even number of bits and same length.

$xy = (x_L x_R)(y_L y_R)$

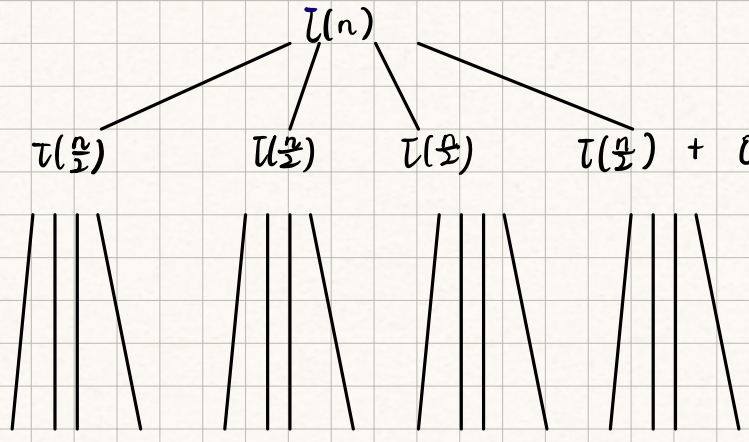
$= (2^{n/2} x_L + x_R)(2^{n/2} y_L + y_R)$

Multiplication of two n digits number $= 2^n \cdot x_L y_L + 2^{n/2} \cdot (x_L y_R + x_R y_L) + x_R y_R$

Multiplication of two n digits number $= 2^n \cdot x_L y_L + 2^{n/2} x_L y_R + 2^{n/2} x_R y_L + x_R y_R$

- Linear Time - $O(n)$
- 4 multiplications of $n/2$ digits number.
- Each term multiplying with 2^n is an easy operation - just add zero's behind.

Recurrence relation



Both multiplication (adding zero's) and addition are $O(n)$

$T(n) = aT(\frac{n}{b}) + f(n)$.
$a = 4$
$b = 2$
$f(n)$ is $O(n^d)$, $d = 1$.

$\therefore T(n) = 4T(\frac{n}{2}) + O(n)$.

Time complexity

$\therefore$ According to master theorem,
$a = 4 > 2^1 = b^d$, Hence, the complexity of this divide and conquer algorithm is $O(n^{\log_2 4}) = O(n^2)$.

Question 3

From question 2.

$$\begin{aligned} xy &= 2^n \cdot x_L y_L + 2^{n/2} \cdot x_L y_R + 2^{n/2} \cdot x_R y_L + x_R y_R \\ &= 2^n \cdot x_L y_L + 2^{n/2} \cdot (x_L y_R + x_R y_L) + x_R y_R \end{aligned}$$

this part can be written as:

already computed

$$(x_L y_R + x_R y_L) = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R .$$

Addition is linear time

Expansion

$$2^{n/2}(x_L y_R + x_R y_L) = 2^{n/2}\left[(x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R\right].$$

$$= 2^{n/2} \cdot (x_L + x_R)(y_L + y_R) - 2^{n/2} \cdot x_L y_L - 2^{n/2} \cdot x_R y_R$$

Substitution

$$xy = 2^n \cdot x_L y_L + 2^{n/2} \cdot (x_L y_R + x_R y_L) + x_R y_R$$

$$= 2^n \cdot x_L y_L + \left[2^{n/2}(x_L + y_R)(y_L + y_R) - 2^{n/2} x_L y_L - 2^{n/2} x_R y_R\right] + x_R y_R .$$

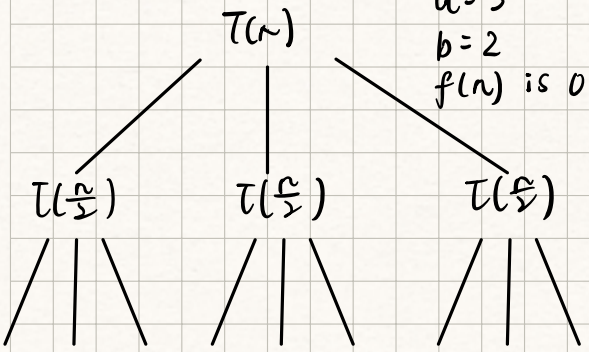
$$= (2^n - 2^{n/2})(x_L y_L) + 2^{n/2}(x_L + y_R)(y_L + y_R) + (1 - 2^{n/2})(x_R y_R)$$

$\frac{n}{2}$ digits number

Recurrence Relation

$$\therefore T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$

$a = 3$
$b = 2$
$f(n)$ is $O(n^d)$, $d = 1$.



Time complexity:

$$T(n) = a\left(\frac{n}{b}\right) + f(n)$$

$\begin{cases} a = 3 \\ b = 2. \\ f(n) \text{ is } O(n^d),\ d = 1. \end{cases}$

∴ According to master theorem,

$a = 3 > 2^1 = b^d$, Hence, the complexity of this divide and conquer algorithm is $O(N^{\log_2 3}) = O(N^{1.58})$