Problem 3.2 Divide & Conquer & Solving Recurrences

a) a & b have 'n' bits each. To multiply a & b,
(i) we multiply each bit of a with right most bit of b. ~~Add it~~ Let the result be bitmul. Add it to sum = sum + bitmul.
(ii) Then shift 'b' right by 1, then repeat 1, until b becomes 0.

Demonstrating with an example,

```
           1 0 1 0
           1 1 1 0
        -----------
step 1     0 0 0 0   +
step 2   1 0 1 0 0   +
step 3 1 0 1 0 0 0   +
step 4 1 0 1 0 0 0 0
+
------------------------ ↓ Final sum
1 0 0 0 1 1 0 0
```

// Note that when we write the code in C, we can't store as data types, so we use array to store ~~the~~ a & b & then use same logic.

We observe that, when we multiply one bit of 'a' with one bit of 'b', the time is $\theta(n)$.
when we multiply 'n' bits of 'a' with 'n' bits of 'b', time is $\theta(n^2)$.

Similarly, adding 1 bit of 'a' with 1 bit of 'b' takes $\theta(n)$. So, adding 'n' bits of 'a' & 'b' takes $\theta(n^2)$.

We also have bit shifting 'n' times in the entire algorithm, this has time complexity $\theta(n)$.

So, total time complexity of this algorithm is

$$= \theta(n^2) + \theta(n^2) + \theta(n)$$
$$= \theta(2n^2 + n) \approx \theta(n^2)$$

b) Divide & conquer

For simplicity, we assume 'n' to be a power of 2

Demonstrating examples (trivial)

we can multiply it like this:

$\rightarrow (54 \cdot 97)$

$= (5 \times 10 + 4)(9 \times 10 + 7)$

$= 5 \times 9 \times 10^2 + 5 \times 7 \times 10 + 4 \times 9 \times 10 + 7 \times 4$

$= \ldots$

$\rightarrow (451 \cdot 608)$

$= (45 \times 10 + 1)(60 \times 10 + 8)$

$= ((4 \times 10 + 5) \times 10 + 1)((6 \times 10) \times 10 + 8)$

$= \ldots$

This shows us possible recursion patterns we could use. Doing it in binary (base 2),

$$a \cdot b = (a_1 2^{n/2} + a_2)(b_1 2^{n/2} + b_2)$$

$$= (a_1 b_1 2^n + a_1 b_2 2^{n/2} + a_2 b_1 2^{n/2} + a_2 b_2)$$

$$= \underset{(i)}{a_1 b_1} 2^n + 2^{\frac{n}{2}} (\underset{(ii)}{a_1 b_2} + \underset{(iii)}{a_2 b_1}) + \underset{(iv)}{a_2 b_2}$$

Here, $a1 = \frac{n}{2}$ left most bits of a

$a2 = \frac{n}{2}$ rightmost bits of a

$b1 = \frac{n}{2}$ leftmost bits of b

$b2 = \frac{n}{2}$ ~~left~~ rightmost bits of b

Then in (i) (ii), (iii) and (iv), we need to do recursive calls. This means 4 recursive calls. And, we have addition with linear time complexity.

So, this would give us time complexity $= 4T\left(\frac{n}{2}\right) + \theta(n)$

$\xleftrightarrow{\text{master method}} = \theta(n^2)$

But time is same, so we need an extra step.

We change the middle 2 terms to some other form to conclude to only one multiplication.

Something like:

$$a_1 b_2 + a_2 b_1 = (a_1 + a_2)(b_1 + b_2) - a_1 b_1 - a_2 b_2$$

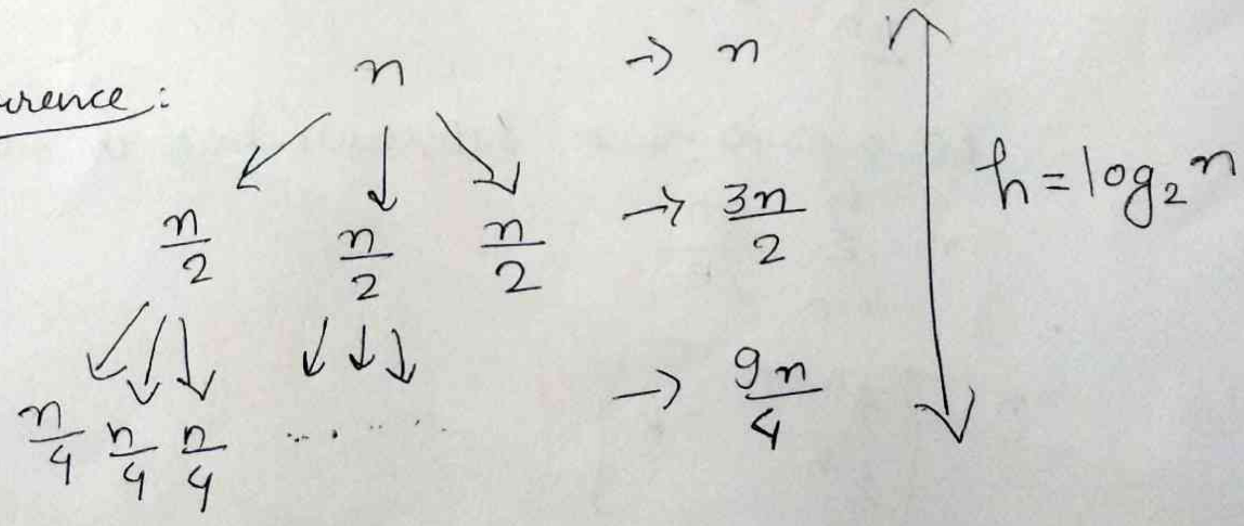
Replacing this in our main eq^n,

$$a \cdot b = \underset{(i)}{a_1 b_1} 2^n + 2^{\frac{n}{2}} \left[\underset{(ii)}{(a_1 + a_2)(b_1 + b_2)} - a_1 b_1 - a_2 b_2 \right] + \underset{(iii)}{a_2 b_2}$$

c) This means, $T(n) = \underbrace{3 T\left(\frac{n}{2}\right)}_{3 \text{ recursions}} + \theta(n)$

Master method $= \theta(n^{\log_2 3}) = \theta(n^{1.59})$

d) Recurrence:



$$= n + \frac{3n}{2} + \frac{9n}{4} + \frac{27n}{8} + \cdots$$

$$= \sum_{k=0}^{h} \frac{3^k \cdot n}{2^k} = n \cdot \sum_{k=0}^{h} \frac{3^k}{2^k} \quad -(i)$$

Analyze the geometric series: $\sum_{k=0}^{h} \frac{3^k}{2^k}$

Here,

total number of series elements $(n) = h - 0 + 1$
$= h + 1$

Ratio $(r) = \frac{3}{2}$

First term $(a) = 1$

So, geometric series $= \frac{a(1-r^n)}{(1-r)} = 1 \frac{\left(1 - \left(\frac{3}{2}\right)^{h+1}\right)}{\left(1 - \frac{3}{2}\right)}$

$$= \frac{1 - \frac{3^{h+1}}{2^{h+1}}}{-\frac{1}{2}} = 2\left[\frac{3^{h+1}}{2^{h+1}} - 1\right]$$

$$= \frac{3^{h+1}}{2^h} - 2$$

~~As n → ∞, h →~~ Substitute back in our ~~series~~ eqn (1)

$$n \cdot \sum_{k=0}^{h} \frac{3^k}{2^k}$$

$$= n \cdot \left[\frac{3^{h+1}}{2^h} - 2\right]$$

$$= n \cdot \left[\frac{3^{\log_2 n + 1}}{2^{\log_2 n}} - 2\right]$$

As $n \to \infty$, $= n\left[\frac{3^{\log_2 n}}{2^{\log_2 n}}\right] = n \cdot \left(\frac{3}{2}\right)^{\log_2 n}$

By log properties, $= n \cdot n^{\log_2 3/2}$

$a^{\log_{\cancel{y}} x} = x^{\log_{\cancel{y}} a}$ $\approx n^{1.58}$ $\therefore$ Time complexity $= \Theta(n^{1.58})$.

e) From recurrence we have,

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

Here, $a = 3$, $b = 2$

$\cancel{f(n) = O(n^{\log_2 3})}$ $\quad n^{\log_2 3} = n^{1.58}$

Here, $f(n) = n$

$= \cancel{n} \; \Theta(n^{1.58 - 0.58})$ where $\epsilon = 0.58$

Thus, $T(n) = \Theta(n^{1.58})$ by case 1 of Master Method