Problem 2:

a) Brute force multiplication in real life (3rd grade multiplicat algorithm has running time of $\Theta(n^2)$. In our case, a and b have n-bits each. Brute-force algorithm is ~~pretty~~ straight forward. Multiply each bit ~~with each~~ from a with each bit of b. Each time shift the ~~previous~~ next one for one place to the left relative to the first one. Consider the example:

$$
\begin{array}{r}
\overset{\to 11}{1011} \cdot \overset{\to 13}{1101} \\
\hline
1011 \\
00000 \\
101100 \\
1011000 \\
\hline
10001111
\end{array}
\longrightarrow 143 \Rightarrow 11 \cdot 13 = 143 \checkmark
$$

After doing that, we add all our results.

First multiplying each bit of a with each bit of b takes n^2 operations $\therefore T_1(n) = \Theta(n^2)$. Adding ~~n bit numbers~~ n bit numbers n times also takes n^2 operations $\therefore T_2(n) = \Theta(n^2)$ (Since each time we shift the number to the left one time, that can be considered constant).

The total time is $T(n) = T_1(n) + T_2(n) = \Theta(n^2) + \Theta(n^2)$

$$\Rightarrow T(n) = \Theta(2n^2) \Rightarrow \boxed{T(n) = \Theta(n^2)}$$

b) Let us do a little magic with numbers in base 10. Say, we want to ~~add~~ multiply 32 and 21.

$32 \cdot 21 = 672$, but

$32 = 10 \cdot 3 + 2$ and $21 = 10 \cdot 2 + 1 \Rightarrow 32 \cdot 21 = (3 \cdot 10 + 2)(2 \cdot 10 + 1)$

$\Rightarrow \quad 6\cdot 10^2 + 7\cdot 10 + 2 = 672$

Another example: $2347 \cdot 3732 = 8759004$

Also: $23\cdot 10^2 + 47 = 2347$

$37\cdot 10^2 + 32 = 3732$

$2347 \cdot 3732 = (23\cdot 10^2 + 47)(37\cdot 10^2 + 32) = 851\cdot 10^4 + 2475\cdot 10^2 + 1504 =$

$= 8759004$ ✓

We can separate those terms recursively one more time.

We can do this algorith also for numbers in base 2.

$$a\cdot b = (a_l \cdot 2^{\frac{n}{2}} + a_r)(b_l \cdot 2^{\frac{n}{2}} \cdot b_r) =$$

$$= \overset{1}{a_l b_l} \cdot 2^n + 2^{\frac{n}{2}}(\overset{2}{a_l b_r} + \overset{3}{a_r b_l}) + \overset{4}{a_r b_r}$$

~~a_l is~~

$b_l/a_l \rightarrow \frac{n}{2}$ left most bits of b/a

$b_r/a_r \rightarrow \frac{n}{2}$ rightmost bits of b/a

In the equation we still have four recursive calls (labeled $1 \rightarrow 4$)

$$\Rightarrow T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n) \xrightarrow{\text{master method}} T(n) = \Theta(n^2)$$

However: $a_l b_r + b_l a_r = (a_l + a_r)(b_l + b_r) - a_l b_l - a_r b_r$

$$\Rightarrow a\cdot b = \overset{1}{a_l b_l} \cdot 2^n + 2^{\frac{n}{2}}\left[\overset{3}{(a_l + a_r)(b_l + b_r)} - a_l b_l - a_r b_r\right] + \overset{2}{a_r b_r}$$

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

$f(n) = \Theta(n)$

$n^{\log_b a} = n^{\log_2 3} \approx n^{1,58}$

~~Since~~

Considering $\Theta(n^{\log_b a + \varepsilon})$ for $\varepsilon \simeq 0.58$ we get:

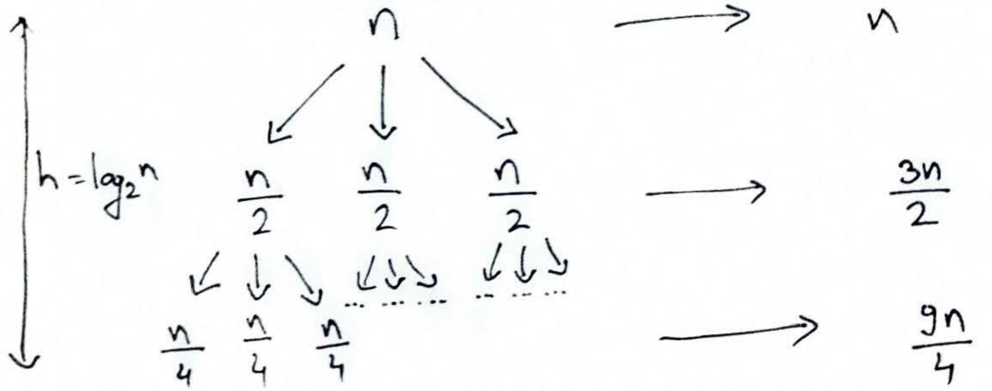
$T(n) = \Theta(n^{1,58})$

---

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

---

d)



$\Rightarrow \quad n + \frac{3n}{2} + \frac{9n}{4} + \frac{27n}{8} + \cdots = \sum_{k=0}^{h} \frac{3^k n}{2^k} = \frac{\frac{1}{2}(3^h - 1)}{2^{h+1} - 1} n$

$\Rightarrow \quad \frac{(3^h - 1)}{2^{h+2} - 2} n = \frac{3^h - 1}{2(2^{h+1} - 1)} n \approx n^{1,58}$

---

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

$f(n) = n$

$n^{\log_b a} = n^{\log_2 3} \approx n^{1,58}$

So using case 3, $\varepsilon > 0 \Leftrightarrow \varepsilon = 0,58$ and $2\frac{n}{3} \leq cn$ for $c < 0$, i.e., $c = \frac{2}{3} \Rightarrow T(n) = \Theta(n^{1,58})$ $\square$