1) Given

$$T(n) = \begin{cases} 2 & \text{if } n = 2 \\ 2T(n/2) + n & \text{where } n = 2^k, k > 1 \end{cases}$$

The above function can also be written as following

$$T(2^k) = \begin{cases} 2 & \text{if } k = 1 \\ 2T(2^k/2) + 2^k & \text{where } k > 1 \end{cases}$$

We have to verify if $T(n) = n \lg n$ ~~$T(2^k)$~~ $=$ which is same as $T(2^k) = 2^k \lg 2^k$ — (1)

<u>Base condition:</u> $(k = 1)$

<u>L.H.S</u> $T(2^1) = 2$ (By function definition)

<u>R.H.S</u> $2^1 \lg 2^1 = 2$

L.H.S = R.H.S $\therefore$ Base condition works for the equation (1)

Let's assume (1) is valid for an arbitarary value p,

$$\Rightarrow T(2^p) = 2^p \lg 2^p \quad \text{— (2)}$$

Now let's verify if $p+1$ is valid

$T(2^{p+1}) = 2^{p+1} \lg 2^{p+1}$

~~LH~~ $\cancel{= 2^p \times 2 \times (p+1) \times 1}$

R.H.S

$= 2^{p+1} \times \lg (2^p \times 2) = 2^p \times 2 (\lg 2^p + \lg 2)$

$= 2 \times 2^p (\lg 2^p) + 2 \times 2^p$

$= 2 \times T(2^p) + 2 \times 2^p$ (From - 2)

L.H.S

$T(2^{p+1}) = 2T(2^p) + 2^{p+1}$ (From definiti

$\Longrightarrow$

L.H.S = R.H.S

$\therefore$ Our assumption is true that for every k

$T(2^k) = 2^k \lg 2^k$

Since $n = 2^k$

$T(n) = n \lg n$

2) Part A

① $n^3 + 3^n$

~~n^3 & 3^n ∞~~ $n^3 \neq 3^n \Rightarrow$ One goes faster than other

~~if $(n^3 > 3^n)$~~

~~$\lim_{n \to \infty} \frac{n^3 + 3^n}{n^3} =$~~

Since $3^n > n^3$ for all large n

$O(n^3 + 3^n) = O(3^n)$ is the simplest form.

②

~~3n lg~~ $3n \log(5n) =$ ~~3n lg~~

$= 3n \log(5 \times n) =$ ~~3n~~ $3n \log 5 + 3n \log n$

for all large n $3n \log n > 3n \log 5$

$\therefore O(3n \log 5n) = O(3n \log n)$

$3n\log n$ & $n \log n$ grow at similar rate

as $\lim_{n \to \infty} \frac{3n \log n}{n \log n} = 3$ (Constant)

$\therefore O(3n \log n) = O(n \log n)$ is the simplest form

(3) $100 \times 2^n + 3^n$

$$\lim_{n \to \infty} \left(\frac{100 \cdot 2^n}{3^n} + \frac{3^n}{3^n} \right) = \cancel{\text{constant}}\ c \ (\text{constant})$$

$\therefore$ 3^n & $100 \times 2^n + 3^n$ grow at same rate

$\therefore$ $O(100 \times 2^n + 3^n) = O(3^n)$ is the simplest form

(4) $80n\log n + 5n^3 + \sqrt{n}$

$$\lim_{n \to \infty} \frac{80n\log n + 5n^3 + \sqrt{n}}{n^3} = 5 \ (\text{constant})$$

$\therefore$ n^3 & $80n\log n + 5n^3 + \sqrt{n}$ grow at same rat.

$\therefore$ $O(80n\log n + 5n^3 + \sqrt{n}) = O(n^3)$

(5) $1^3 + 2^3 + \cdots n^3 = \frac{(n^2(n+1)^2)}{4}$

$$\frac{n^2(n^2+2n+1)}{4} = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$$

$$\lim_{n \to \infty} \frac{\frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}}{n^4} = c \ (\text{constant})$$

$\therefore$ $O(1^3 + 2^3 + \cdots n^3) = O(n^4)$ is the simplest)

Part B

$$f(n) = 1 + 2 + \cdots 2^n$$

$$\cancel{(a^n + a^{n-1} \cdots a + 1)(a-1) = a^{n+1} - 1}$$

$$\cancel{f(n+1) = 1 + 2 + \cdots 2^n + 2^{n+1}} = (f(n) + 2^{n+1})$$

$$\cancel{f(n+1) - f(n) = 2^{n+1}}$$

$$
\begin{aligned}
2 \times f(n) &= \cancel{2} + \cancel{4} + \cdots \cancel{2^n} + 2^{n+1} \\
- \quad f(n) &= 1 + \cancel{2} + \cancel{4} \cdots \cancel{2^n} \\
\hline
f(n) &= 2^{n+1} - 1
\end{aligned}
$$

Base case: $f(1) = 1 + 2^1 = 3$

$2^{1+1} - 1 = 4 - 1 = 3$

$\therefore$ Base case is true

Let's assume $\boxed{f(K) = 2^{K+1} - 1}$ — (1) for arbitrary K.

for $(K+1)$ $\quad f(K+1) = 2^{K+2} - 1$

LHS

$$f(K+1) = 1 + 2 + \cdots 2^K + 2^{K+1}$$

$$f(K+1) = f(K) + 2^{K+1} = 2^{K+1} - 1 + 2^{K+1}$$

$$= 2 \times 2^{K+1} - 1 = 2^{K+2} - 1$$

LHS = RHS

$\therefore$ Our assumption is true $f(K) = 2^{K+1} - 1$

As $f(k) = 2^{k+1} - 1 \Rightarrow f(k) \in \Theta(2^{k+1})$

~~$O(f(k)) = O(2^{k+1})$~~

$f(k) \in O(2^{k+1})$

3) 1) A bit can store 2 states of information.

So to store a positive integer n we need x bits at least let's assume

$2^x \geq n \Rightarrow \log_2(2^x) \geq \log_2 n$

$x \lg 2 \geq \lg n \Rightarrow x \geq \lg n$

We ~~need~~ only need ~~as many~~ the smallest value of x

So we choose one closest to lg n.

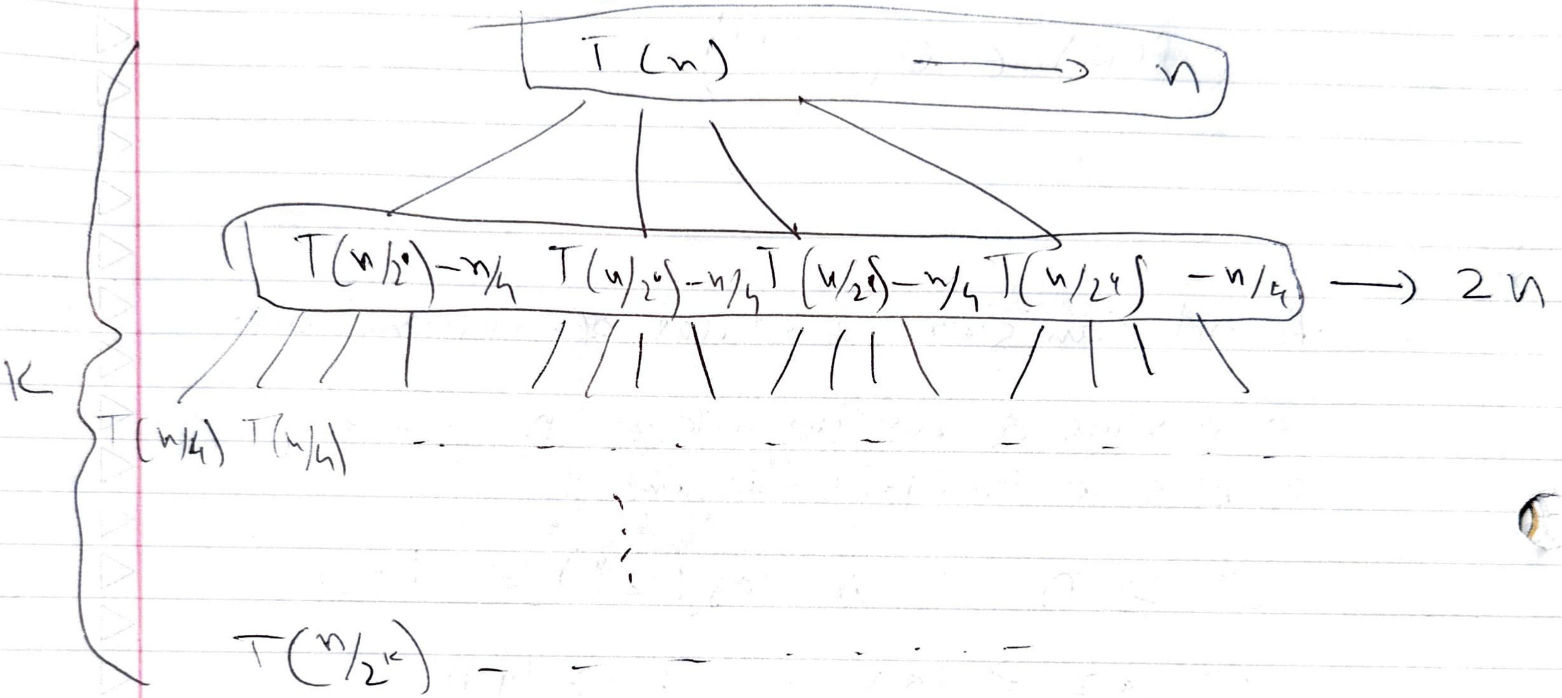
~~#~~ ∴ Number of bits would $\in O(\lg n)$

2) In this case we are dividing n by 2 in every iteration. By defination log is repeated division

∴ the loop would iterate $\log_2 n$ times.

∴ "Hello" would be printed $\log_2 n$ times.

4) $T(n) = \begin{cases} 1 & n = 1 \\ 4(T(n/2)) + n & n = 2^K, K > 1 \end{cases}$



Every level the effort doubles ~~to~~ from ~~a~~ previous.

So total effort $= n + 2n + \cdots \cancel{(k+1)}\ 2^K n$

$\cancel{= n(1+2+\cdots k+1)\ n\left(\frac{K(K-1)}{2}\right)}$

$\cancel{= n\left(\frac{(k+1)(k+2)}{2}\right)}$

$= n(1 + 2 + \cdots 2^K)$

$= \boxed{n(2^{K+1} - 1)} = n 2^K \times 2 - n$

$= 2n^2 - n$ (From function definition)

$= O(n^2)$

$T(n) = O(n^2)$

To verify by substitution let's substitute in function definition

$$T(n) = 4T(n/2) + n \quad (n = 2^k \; \& \; k \geq 1)$$

$$T(n) = 4(O((n/2)^2)) + n$$

We can write $O(n^2)$ as $cn^2 + d$

$$T(n) = 4(cn^2 + d) + 4$$

$$= 4(n^2 + d + 4$$

$$= O(n^2)$$

$\therefore$ we proved our finding with substitution method

(5)

The time it takes to insert n^{th} element in a list ~~can be~~ depends on the element & list of elements before it.
The upper bound of inserting would be $n-1$ in the case when we have largest element.

$$T(n) = T(n-1) + O(n)$$

$$\Rightarrow T(n) = T(1) + O(2) + O(3) + \cdots O(n)$$

$T(1)$ is constant $\Rightarrow T(n) = c(1 + 2 + 3 + \cdots n)$
(c is a constant)

$$T(n) = c\,\frac{n(n+1)}{2} = O(n^2)$$

6) a) $T(n) = 2T(n/2) + n^4$

$$O(n^{\log_b a}) = O(n^{\log_2 2})$$

$f(n) = n^4 = O(n^4) = O(n^{1+\varepsilon})$

Here $\varepsilon = 3$ (3rd case)

$af(n/b) < cf(n)$ to satisfy regularity.

$2f(n/2) < cf(n)$

$2\frac{n^4}{16} < cn^4$ is true for $c > \frac{1}{8}$.

$\therefore$ This satisfies regularity condition.

$\boxed{T(n) = O(n^4)}$

b) $T(n) = T(7n/10) + n$

$$O(n^{\log_b a}) = O(n^{\log_{10/7} 1}) = O(n^0)$$

$f(n) = n = O(n) = O(n^{0+\varepsilon})$

$\varepsilon = 1$ (3rd case)

$af(n/b) < cf(n)$ to satisfy regularity

$f(7n/10) < cf(n)$

$\frac{7n}{10} < cn$ is true for $c > 7/10$

$\therefore$ This satisfies regularity condition

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

c) $T(n) = 16\,T(n/4) + n^2$

$f(n) = \Theta(n^2) = \cancel{\Theta(n^{\log_4 16})}$

$g(n) = n^{\log_4 16} = n^2$

$f(n) = \Theta(n^2) = g(n)$ (second condition)

$\therefore T(n) = n^2 \log_4 n$

d) $T(n) = 2T(n/4) + \sqrt{n}$

$\boxed{\text{Since } f(n) = \sqrt{n} \text{ is not a polynomial we can't apply master's theorem.}}$

By intuition we know that $n^{\log_8 9} = \sqrt{n}$

$\therefore T(n) = \sqrt{n} \log_4 n$

e) $T(n) = \sqrt{2}\,T(n/2) + \log n$

Since subproblems can't be irrational the recurrence relation is invalid which ~~makes the ma~~ means we can't apply master's theorem.

f) $T(n) = 64\,T(n/8) - n^2 \log n$

By definition $f(n)$ should be ~~a pos~~ an asymptomatically positive function. Since $-n^2 \log n$ is not asymptomatically positive we can't apply master's theorem.

g) $T(n) = 2T(n/4) + n^{0.51}$

Since $f(n) = n^{0.51}$ it's not a polynomial.
So we can't apply master's theorem.

By intuition $n^{\log_4 2} < n^{0.51}$
$T(n) = O(n^{0.51})$

h) $T(n) = 16T(n/4) + n!$

$f(n) = n!$ which can't be bounded by any polynomial function.
$\therefore$ Master's theorem won't apply.

i) $T(n) = \frac{1}{2}T(n/2) + 1/n$
$f(n) = 1/n$ is not an asymptomatically positive function and ~~a is~~ number of subproblems can't be $1/2$.
$\therefore$ Master's Theorem doesn't apply.

j) $T(n) = 2^n T(n/2) + n^n$

~~T(n)~~ $f(n)$ is an exponential function & not polynomial.
$\therefore$ Master's theorem doesn't apply.