1. Find the value of $T(2)$ for the recurrence relation $T(n) = 3T(n-1) + 12n$, given that $T(0) = 5$.

A: Rewriting the Recurrence Relation:

$$T(n) = \begin{cases} 3T(n-1) + 12n, & n > 0 \\ 5, & n = 0 \end{cases}$$

As, we know, $T(0) = 5$, let's try to make $n-1 = 0$, $\therefore \boxed{n = 1}$.

$$\therefore T(1) = 3T(1-1) + 12*1$$
$$= 3 * T(0) + 12$$
$$= 3 * 5 + 12$$
$$\therefore \boxed{T(1) = 27} \longrightarrow (1)$$

Let's solve for $T(2)$: directly substituting the value of $n = 2$.

$$T(2) = 3 * T(2-1) + 12*2$$
$$\boxed{T(2) = 3 * T(1) + 24} \longrightarrow (2)$$

Substituting eq (1) in (2)

we get,

$$T(2) = 3 * 27 + 24$$

$$\therefore \boxed{\text{The value of } T(2) = 105}$$

//

2. Given a Recurrence Relation, solve it using the substitution method.

a. $T(n) = T(n-1) + c$.

A: $T(n) = T(n-1) + c$ is given.

As, we all know that this recurrence relation is of Factorial, so rewriting this would give,

$$T(n) = \begin{cases} c, & n \leq 1 \\ T(n-1) + c, & n > 1 \end{cases}$$

where $c = 1$, a constant time.

Solving it by substitution method,

$$\boxed{T(n) = T(n-1) + c} \longrightarrow (1)$$

solving for $T(n-1)$,

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

$$\boxed{T(n-1) = T(n-2) + c} \longrightarrow (2)$$

Substituting (2) in (1)

$$T(n) = [T(n-2) + c] + c$$

$$\boxed{T(n) = T(n-2) + 2c} \longrightarrow (3)$$

Solving for $T(n-2)$,

$$T(n-2) = T((n-2)-1) + c$$

$$\boxed{T(n-2) = T(n-3) + c} \longrightarrow (4)$$

Substituting (4) in (3)

$$T(n) = [T(n-3) + c] + 2c$$

$$\therefore \boxed{T(n) = T(n-3) + 3c}$$

Continuing the substitution for k times

$$T(n) = T(n-k) + kc$$

We know that,
when $n \leq 1$, Time complexity is 1. $\Rightarrow T(0) = 1, T(1) = 1$

$$n - k = 0$$
$$\therefore \boxed{n = k}$$

$$T(n) = T(n-n) + nc$$
$$= T(0) + nc$$

$$\boxed{T(n) = 1 + nc}$$

Writing in terms of Big O.

$\boxed{T(n) = O(n \cdot c)} \Rightarrow$ as we need to ignore the constants

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

it's a Linear Time Complexity.

b. $T(n) = 2T(n/2) + n$

A: Rewriting the given Recurrence Relation, by adding a base case,

$$T(n) = \begin{cases} c, & n \leq 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + n, & n > 1 \end{cases}$$

Solving for $\boxed{T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n}$ → (1) by substitution,

$$T\left(\frac{n}{2}\right) = 2 \cdot T\left(\frac{n}{2*2}\right) + \frac{n}{2}$$

$$\therefore \boxed{T\left(\frac{n}{2}\right) = 2 \cdot T\left(\frac{n}{4}\right) + \frac{n}{2}} \rightarrow (2)$$

Substituting (2) in (1)

$$T(n) = 2\left[2 \cdot T\left(\frac{n}{4}\right) + \frac{n}{2}\right] + n$$

$$\boxed{T(n) = 4T\left(\frac{n}{4}\right) + 2n} \rightarrow (3)$$

Solving for $T\left(\frac{n}{4}\right)$

$$T\left(\frac{n}{4}\right) = 2 \cdot T\left(\frac{n}{4*2}\right) + \frac{n}{4}$$

$$\boxed{T\left(\frac{n}{4}\right) = 2 \cdot T\left(\frac{n}{8}\right) + \frac{n}{4}} \rightarrow (4)$$

Substituting (4) in (3)

$$T(n) = 4\left[2 \cdot T\left(\frac{n}{8}\right) + \frac{n}{4}\right] + 2n$$

$$\therefore \boxed{T(n) = 8T\left(\frac{n}{8}\right) + 3n} \rightarrow (5)$$

Solving for $\frac{n}{8}$,

$$T\left(\frac{n}{8}\right) = 2\left(T\left(\frac{n}{8*2}\right)\right) + \frac{n}{8}$$

$$\boxed{T\left(\frac{n}{8}\right) = 2 \cdot T\left(\frac{n}{16}\right) + \frac{n}{8}} \rightarrow (6)$$

Substituting (6) in (5)

$$T(n) = 8 \cdot \left[2\,T\left(\frac{n}{16}\right) + \frac{n}{8}\right] + 3n$$

$$\boxed{T(n) = 16\,T\left(\frac{n}{16}\right) + 4n}$$

Substituting for k times, we get,

$$\boxed{T(n) = 2^k \cdot T\left(\frac{n}{2^k}\right) + kn.} \rightarrow (7)$$

When, $\frac{n}{2^k} = 1$, $n = 2^k$.

taking $\log_2$ on Both sides.

$$\log_2 n = k \cdot \log_2 2^1$$

$$\therefore \boxed{k = \log_2 n}$$

Substituting the value of k in eqn (7)

$$T(n) = 2^{\log_2 n} \cdot T\left(\frac{n}{2^{\log_2 n}}\right) + \log_2 n \cdot n.$$

$$= n^{\log_2 2} \cdot T\left(\frac{n}{n^{\log_2 2}}\right) + n \cdot \log_2 n$$

$$= n \cdot T(1) + n \cdot \log_2 n$$

$$= n \cdot 1 + n \cdot \log_2 n, \quad \text{because base case says } \boxed{T(1) = 1}$$

$$\boxed{T(n) = n + n \cdot \log_2 n}$$

We get,

$$\boxed{T(n) = n + n.\log_2 n}$$

writing in terms of Big O notation,

$$\boxed{\text{Time complexity} = O(n.\log_2 n)},$$

because for Big O time complexity, all we care about is a dominating term.

$n \log n$ dominates n

$$\therefore \boxed{T(n) = O(n.\log_2 n)}$$ //

C. $T(n) = 2T\left(\frac{n}{2}\right) + c$

A: Rewriting the given Recurrence Relation, by adding a base case condition.

$$T(n) = \begin{cases} c, & n \leq 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + c, & n > 1 \end{cases}$$

Solving for $\boxed{T(n) = 2\,T(n/2) + c}$ → (1) by substitution method.

Replacing n with $n/2$ in eqn (1), we get

$$T\left(\frac{n}{2}\right) = 2 \cdot T\left(\frac{n}{2*2}\right) + c$$

$$\boxed{T(n/2) = 2 \cdot T(n/4) + c} \rightarrow (2)$$

Substituting (2) in (1)

$$T(n) = 2\left(2 \cdot T\left(\frac{n}{4}\right) + c\right) + c$$

$$\therefore \boxed{T(n) = 4 \cdot T\left(\frac{n}{4}\right) + 3c} \rightarrow (3)$$

Replacing n with $n/4$ in eqn (1), we get

$$T(n/4) = 2 \cdot T\left(\frac{n}{4*2}\right) + c$$

$$\boxed{T(n/4) = 2 \cdot T\left(\frac{n}{8}\right) + c} \rightarrow (4)$$

Substituting (4) in (3)

$$T(n) = 4\left(2 \cdot T\left(\frac{n}{8}\right) + c\right) + 3c$$

$$T(n) = 8 \cdot T\left(\frac{n}{8}\right) + 4c + 3c$$

$$\therefore \boxed{T(n) = 8 \cdot T\left(\frac{n}{8}\right) + 7c} \rightarrow (5)$$

Replacing n with $n/8$ in eqn (1), we get

$$T\left(\frac{n}{8}\right) = 2\,T\left(\frac{n}{8*2}\right) + c$$

$$\boxed{T\left(\frac{n}{8}\right) = 2 \cdot T\left(\frac{n}{16}\right) + c} \rightarrow (6)$$

Substituting (6) in (5),

$$T(n) = 8 \cdot \left(2 \cdot T\left(\frac{n}{16}\right) + c\right) + 7c$$

$$\boxed{T(n) = 16\,T\left(\frac{n}{16}\right) + 15c}$$

Substituting for k times, we get

$$\boxed{T(n) = 2^k \cdot T\left(\frac{n}{2^k}\right) + (2^k - 1)\,c} \rightarrow (7)$$

We know that $T(1)$ = constant, 1

$$\therefore \frac{n}{2^k} = 1 \Rightarrow n = 2^k.$$

taking $\log_2$ on B.S.

$$\log_2 n = k \cdot \log_2 2,$$

$$\therefore \boxed{k = \log_2 n}$$

Substituting the value of k in eq (7),

$$T(n) = 2^{\log_2 n} \cdot T\left(\frac{n}{2^{\log_2 n}}\right) + (2^{\log_2 n} - 1)\,c$$

$$= n^{\log_2 2} \cdot T\left(\frac{n}{n^{\log_2 2}}\right) + (n^{\log_2 2} - 1)\,c$$

$$= n \cdot T(1) + (n-1)\,c$$

$$\boxed{T(n) = n + (n-1)c}$$

writing in terms of Big O,

$$\boxed{\text{Time Complexity} = O(n)}$$ //

Because, both terms: $(n-1)c$ is a constant

$\therefore O(n)$ //

d. $T(n) = T\left(\frac{n}{2}\right) + c$

A: Rewriting the given Recurrence Relation, by adding a base case condition,

$$T(n) = \begin{cases} c, & n \leq 1 \\ T\left(\frac{n}{2}\right) + c, & n > 1 \end{cases}$$

Solving for $\boxed{T(n) = T\left(\frac{n}{2}\right) + c}$ → (1) by substitution.

Replacing n by $n/2$ in eqn (1) we get

$\boxed{T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + c}$ → (2)

Substituting eqn (2) in (1)

$$T(n) = \left(T\left(\frac{n}{4}\right) + c\right) + c$$

$\therefore \boxed{T(n) = T\left(\frac{n}{4}\right) + 2c}$ → (3)

Replacing n by $n/4$ in eqn (1) we get

$\boxed{T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + c}$ → (4)

Substituting (4) in (3)

$$T(n) = \left(T\left(\frac{n}{8}\right) + c\right) + 2c$$

$\therefore \boxed{T(n) = T\left(\frac{n}{8}\right) + 3c}$ → (5)

Replacing n by $n/8$ in eqn (1), we get

$T(n/8) = T\left(\frac{n}{16}\right) + c$ → (6)

substituting (6) in (5)

$\boxed{T(n) = T\left(\frac{n}{16}\right) + 4c}$

Substituting for k times,

$$T(n) = T\left(\frac{n}{2^k}\right) + kc \longrightarrow (1)$$

W.K.T, $T(1)$ = constant time complexity

$\therefore \frac{n}{2^k} = 1 \implies n = 2^k$

taking $\log_2$ on B.S

$$\log_2 n = k \cdot \log_2 2$$

$$\therefore \boxed{k = \log_2 n}$$

Substituting the value of k in eq (1)

$$T(n) = T\left(\frac{n}{2^{\log_2 n}}\right) + \log_2 n\, c$$

$$= T\left(\frac{n}{n^{\log_2 2}}\right) + \log_2 n\, c$$

$$= T(1) + \log_2 n\, c$$

$$= a + \log_2 n\, c$$

ignoring the constants and writing the time complexity in terms of Big O notation.

$$\boxed{T(n) = O(\log_2 n)}$$

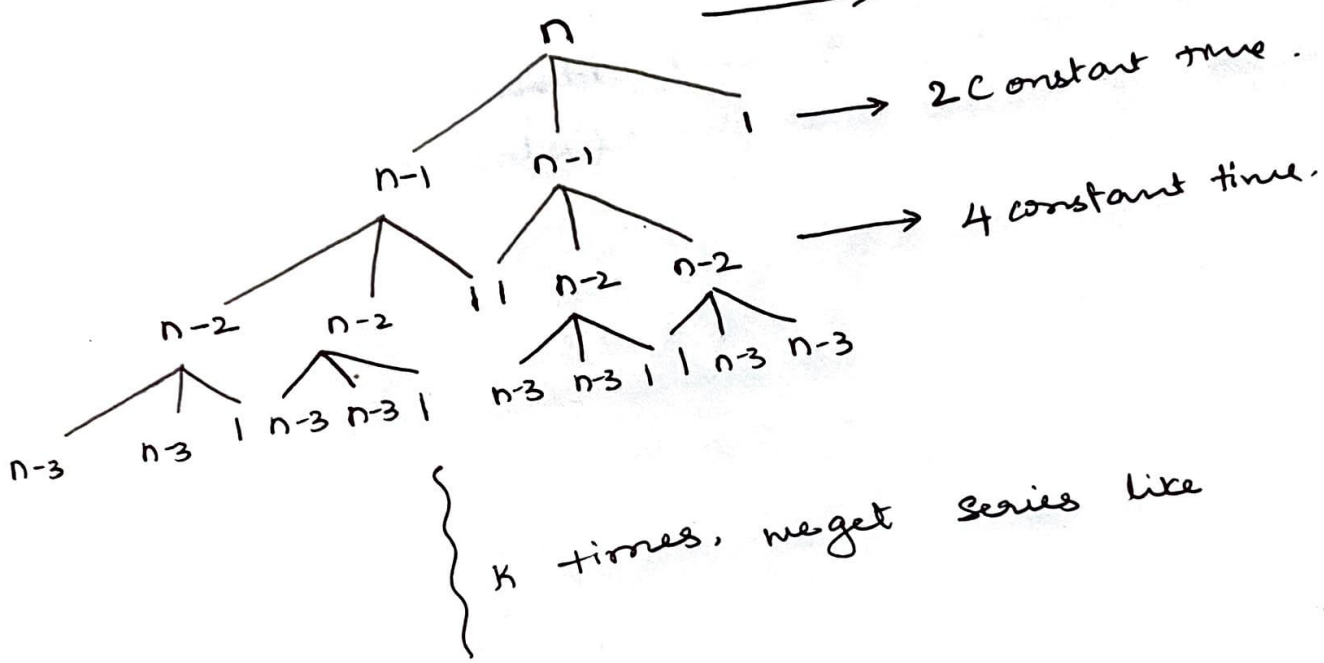
3. Given a Recurrence Relation, solve it using the recursive tree approach.

a. $T(n) = 2T(n-1) + 1$

A: Rewriting the recurrence relation by adding a base case,

$$T(n) = \begin{cases} 1, & n = 0 \\ 2T(n-1) + 1, & n > 0 \end{cases}$$

Solving for $T(n) = 2T(n-1) + 1$ using Recursive Tree approach.



$$= 1 + 2 + 4 + 8 + 16 + \cdots +$$

$$\boxed{T(n) = 2^0 + 2^1 + 2^2 + 2^3 + 2^4 + \cdots + 2^k} \; T(n-k) + kc.$$

this is a GP series.

∴ Sum of GP series, $\dfrac{a(r^{k+1} - 1)}{r - 1}$

where $a = 1$, $r = 2$

$T(n) = 2^{k+1} - 1 \Rightarrow T(n) = (2^{k+1} - 1)\, T(n-k) + kc$

We know that, when $n=0$, $T(0)=1$

Assuming $n-k=0$

$\therefore n=k.$

$\therefore T(n) = 2^{n+1} - 1$

Writing in terms of Big O time complexity we get.

$T(n) = 2^{n+1} - 1 + nc.$

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

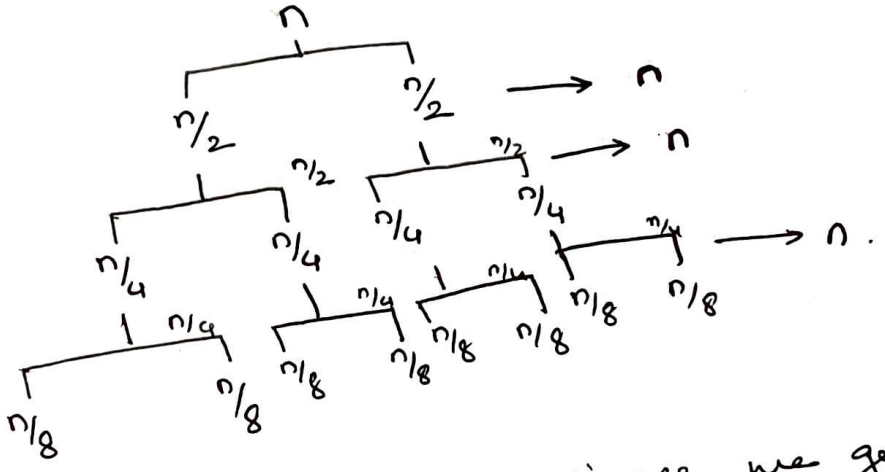
we always should take the dominating term, when calculating the time complexity.

b. $T(n) = 2T\left(\frac{n}{2}\right) + n$

A: Rewriting the Recurrence Relation, by adding a base case condition,

$$T(n) = \begin{cases} 1 & , n = 0 \\ 2T(n/2) + n & , n > 0 \end{cases}$$

Solving $T(n) = 2T(n/2) + n$ using recursive tree approach.

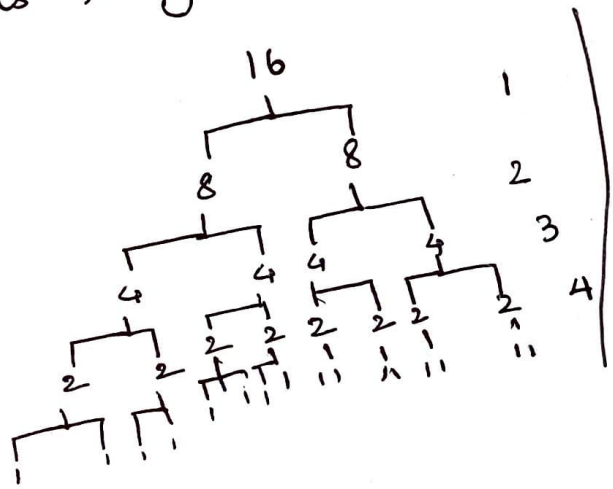


} dividing for k times we get

we get series like the number of height

The complexity to divide the task into sub task is height of the tree.

when $n = 16$,



$\log_2 16 = 4$

this is interms of logarithmic.

$\therefore$ $c.n.\log n$

writing the relation in terms of Big O notation, we get the time complexity as

$$\boxed{T(n) = O(n.\log n)}$$ //