Name: S. Sabitha.

Reg No: 192321156

ASSIGNMENT - 01.

Course: Design & Analysis of Algorithm

Course Code: CSA0669.

1. Solve the following recurrence relations:

a) $x(n) = x(n-1) + 5$ for $n>1$ $x(1)=0$.

$$x(n) = x(n-1) + 5 \rightarrow (1)$$

$$x(n-1) = x(n-1-1) + 5$$

$$= x(n-2) + 5 \rightarrow (2)$$

$$x(n-2) = x(n-2-1) + 5$$

$$= x(n-3) + 5 \rightarrow (3)$$

sub eqn (3) in (2)

$$x(n-1) = x(n-3) + 5 + 5$$

$$= x(n-3) + 10 \rightarrow (4)$$

sub eqn (4) in eqn (1)

$$x(n) = x(n-3) + 10 + 5$$

$$= x(n-3) + 15$$

for some k.

$$x(n) = x(n-k) + 5k \text{ — } (5)$$

$$n-k = 1$$

$$n-1 = k$$

eqn (5)

$$x(n) = x(1) + 5(n-1)$$

$$x(n) = 0 + 5n - 5$$

$$O(n).$$

b). $x(n) = 3x(n-1)$ for $n>1$, $x(1) = 4$.

$x(n) = 3x(n-1)$ — 1

$x(n-1) = 3x(n-1-1) = 3x(n-2)$ — 2

$x(n-2) = 3x(n-3)$ — 3

Sub eqn 3 in 2,

$x(n-1) = 3\{3x[n-3]\}$

$x(n-1) = 9x(n-3)$ — 4

Sub eqn 4 in 1

$x(n) = 3[9x(n-3)]$

$x(n) = 27x(n-3)$

at some k,

$x(n) = 3^k x(n-k)$ — 5

$n-k = 1$

$k = n-1$

eqn 5 $\Rightarrow$ $x(n) = 3^{n-1} x(1)$

$= 3^{n-1} \cdot 4$

$= 3^n \cdot 3^{-1} \cdot 4$

$= 3^n$.

$\therefore$ The time complexity $= O(3^n)$

c). $x(n) = x(n/2) + n$ for $n>1$ $x(1) = 1$

(solve for $n = 2^k$).

$x(n) = x(n/2) + c$ — 1

$x(n/2) = x(n/4) + c$ — 2

$x(n/4) = x(n/8) + c$ — 3

sub ② in ①:

$x(n) = x(n/4) + c + c$

$x(n) = x(n/4) + 2c$ — ④

$= x(n/2^2) + 2c$

sub ⑤ in ④

$x(n) = x(n/8) + c + 2c$

$x(n) = x(n/2^3) + 3c$

$x(n) = x(n/2^k) + kc$

$n = 2^k$; $x(1) = 1$

$x(n) = x(n/n) + kc$

$x(n) = 1 + kc$

$x(n) = 1 + \log n + c$.

Time complexity $= O(\log n)$.

d). $x(n) = x(n/3) + 1$ for $n > 1$ $x(1) = 1$ (solve for $n = 3^k$)

$x(n) = x(n/3) + 1 \rightarrow$ ①

$x(n/3) = x(n/9) + 1$ — ②

$x(n/9) = x(n/27 + 1$ — ③.

sub ② in ①

$x(n) = x(n/9) + 2$ — ④

sub ③ in ④

$x(n) = x(n/27) + 3$ — ⑤

$= x(n/3^3) + 3$

$x(n) = n(n/3^k) + k$.

$x(n) = x(n/3^k) + c.$

$= x(n/n) + k$

$= x(1) + k.$

$= 1 + k.$

$x(n) = \log n$

$\therefore$ Time complexity $= O(\log n)$,

2. Evaluate following recurrences. completely.

B. $T(n) = T(n/2) + 1$ where $= 2k$ for all $k \geq 0$

$T(n) = T(n/2 + 1)$ $n = 2k.$

Sub $n = 2k$

$T(2^k) = T(2^k/2) + 1 = T(2^{k-1}) + 1.$

$n = k-1$

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

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

$n = k-2$

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

$= T(2^{k-3}) + 1.$

$(2^1) + T(2^0) + 1.$

$n^* = 2^k \Rightarrow k = \log_2 n.$

$T(2^k) = T(2^{k-1}) + 1 = T(2^{k-2}) + 1 + 1 \cdots$

Since,

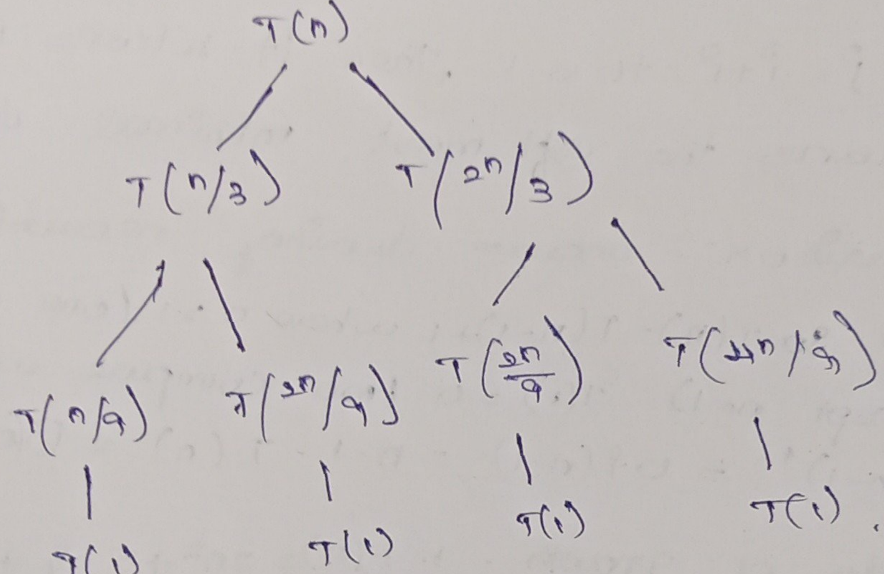
$2^0 = 1$, $T(2^0) = T(1)$

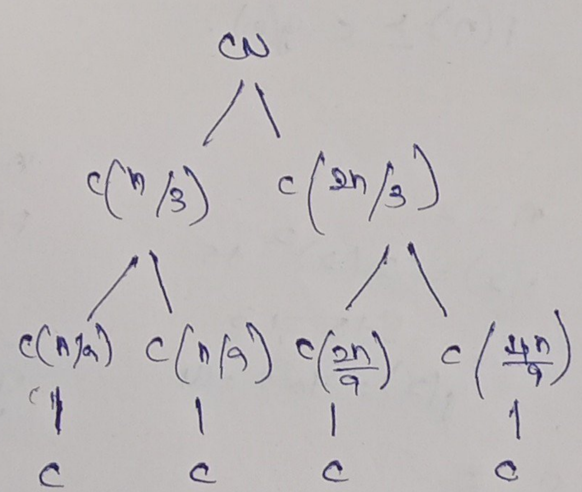
$T(2^k) = 1 + k$.

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

Time complexity = $O(\log n)$

we recursion tree method.





3. consider following algorithm.

min 1 [A [0 .... n-1])

if n = 1 return A[0] - 1

else temp . min 1[A [0 . . . . n-2])
if temp < A [n-1] return temp
else
return A[n-1] - n-1

a). what does this algorithm computes?
This algorithm computa minimum elements
in an array A of Size of n.
If $i < n$, A[i] is Smaller than all elements
then . A[i], j = i+i to n-1, then it returns A[i].
It also returns the left most minimal element.

b). Mainly comparison occurs during recursion
and solve it ? So $T(n) = T(n-1) + 1$ where $n > 1$ (one comparison
Every step except $n=1$). $T(1) = 0$ (no compare when $n = 1$)
$T(n) = T(1) + (n-1)^{*} = 0 + (n-1) = n-1$ $T(n) = O(n)$

4. Analyze Order of growth. 1. $F(n) = 2n^2 + 5$ & $g(n) = 7n$.
$F(n) \geq c \cdot g(n)$.
$F(n) = 2n^2 + 5$
$c \cdot g(n) = 70$

$n = 1$
$F(1) = 2(1)^2 + 5 = 7$
$g(1) = 7$

$n = 2$
$F(2) = 2(2)^2 + 5$
$= 8 + 5 = 13$
$g(2) = 7 \times 2 = 14$

$n = 3$
$F(3) = 2(3)^2 + 5$
$= 18 . 15$
$= 23$.
$g(3) = 21$.

$n \geq 3$, $F(n) \geq g(n) \cdot c$
F(n) is always greater than or Equal
to where n value is greater or equal to 3.
$\therefore F(n) = c\,(g(n))$.