NAME: J. SRILEKHA
DEPT: B-Tech [IT]
COURSE: DATA ANALYSIS OF ALGORITHM. [CSA0669]
ASSIGNMENT: 1.

(11) Solve the following recurrence relations:

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

$x(n) = x(n-1) + 5$ — (1)

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

$= x(n-2) + 5$ — (2)

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

$= x(n-3) + 5$ — (3)

Sub Eq (3) in (2)

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

$= x(n-3) + 10$ — (4)

Sub eq (4) in Eq (1)

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

$= x(n-3) + 15$

for some k,

$x(n) = x(n-k) + 5k$ — (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 Eq ③ in ②,

$x(n-1) = 3[3x[n-3]]$

$x(n-1) = 9x(n-3)$ — ④

Sub Eq 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

$n-k=1$

$k = n-1$

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

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

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

$= 3^n$

∴ The time complexity $= O(3^n)$

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

(Solve for $n = 2k$)

$x(n) = x(n/2) + c \longrightarrow (1)$

$x(n/2) = x(n/4) + c \longrightarrow (2)$

$x(n/4) = x(n/8) + c \longrightarrow (3)$

Sub (2) in (1);

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

$x(n) = x(n/4) + 2c \longrightarrow (4)$

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

Sub (3) in (4)

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

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

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

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

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

$x(n) = 1 + kc$

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

Time complexity $= O(\log n)$

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

$x(n) = x(n/3) + 1$ — 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

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

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

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

$= x(n/n) + k$

$= x(1) + k$

$= 1 + k$

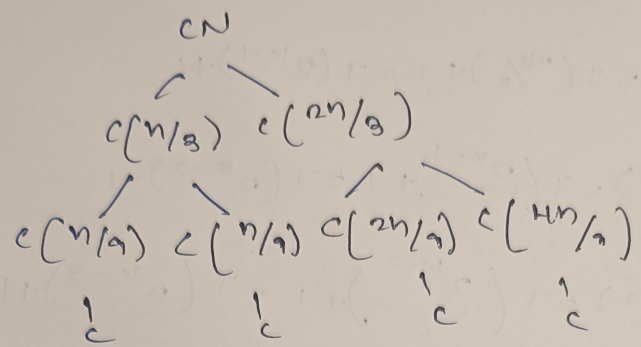
$x(n) = \log n$

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

② Evaluate following recurrences completely.

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

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



(3) consider following algorithm

```
min1[A[0.....n-1])
if n=1 return A[0]-1
Else temp-min1[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 computes minimum elements in an array A of size n

If $i<n$, $A[i]$ is smaller than all element then $A[i], i = i+1$ 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?

sub $n=2^k$

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

$n=k-1$

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

$n=k-2$

$T(2^{k-2}) = T\left(\frac{2^{k-2}}{2}\right)+1 = T(2^{k-3})+1$

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

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

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

since,

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

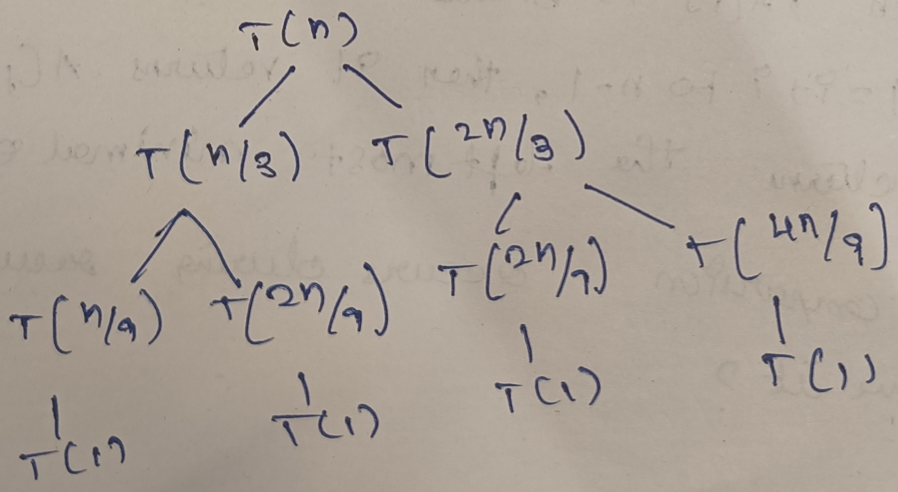
$T(2^k) = 1+k$

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

Time complexity $= O(\log n)$

ii) $T(n) = T(n/3) + T(2n/3) + cn$

use recursion tree method



So, $T(n) = T(n-1) + 1$ where $n > 1$ (one comparison every step except $n = 1$)

$T(1) = O$ (No compare when $n = 1$)

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

$= O + (n-1)$

$= n - 1$

Time complexity $= O(n)$

(11) Analyze order of growth.

(i) $F(n) = 2n^2 + 5$ and $g(n) = n$ use $\{g(n)\}$ solution

$F(n) \geq C \cdot g(n)$

$F(n) = 2n^2 + 5$

$C \cdot g(n) = 7n$

$n = 1$

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

$g(1) = 7$

$n = 2$

$F(2) = 2(2)^2 + 5$

$= 8 + 5 = 13$

$g(2) = 7 * 2 = 14$

$n = 3$

$F(3) = 2(3)^2 + 5$

$= 18 + 5$

$= 23$

$g(3) = 21$

$n = 1, 7 = 7$

$n = 2, 13 = 14$

$n = 3, 23 = 21$

$n \geq 3, F(n) \geq g(n) \cdot C$

$F(n)$ is always greater than or equal to where n value is greater or equal to 3

$\therefore F(n) = \Omega(g(n))$

$F(n)$ is grows more than $g(n)$ from below asymptotically.