1

Design

Analytical Assignment

S. MD. Waseem Rayan
192373004

1. Solve the following Recurrence Relations

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

$x(2) = x(2-1) + 5 = x(1) + 5 = 0 + 5 = 5$

$x(3) = x(3-1) + 5 = x(2) + 5 = 5 + 5 = 10$

$x(4) = x(4-1) + 5 = x(3) + 5 = 10 + 5 = 15$

So from above recurrence relation,

$x(n) = 5n$ for $n > 1$

for some k,

$x(n) = x(n-k) + 5k.$

$n - k = 1$

$n - 1 = k$

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

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

$O(n)$

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

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

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

$= 3^n$

$\therefore$ The Time Complexity $= O(3^n)$

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

(solve for $n=2^k$)

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

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

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

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

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

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

$x(n) = x(n/2^k) + kc \longrightarrow \boxed{k = \log n}$

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

$x(n) = x(\cancel{n}/\cancel{n}) + kc$

$x(n) = 1 + kc$

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

Time Complexity $= O(\log n)$

d) $x(n) = x(n/3) + 1$ for $n > 1$ $\quad x(1) = 1$

(solve for $n = 3k$)

$x(n) = x(n/3) + 1$ — 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)

$$x(n) = x(n/27) + 3 \rightarrow (5)$$

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

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

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

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

$$= x(1) + k$$

$$= 1 + k$$

$$x(n) = \log n$$

Time complexity $= O(\log n)$

2) Evaluate following recurrences completely

i) $T(n) = T(n/2) + 1$ where $n = \underline{2^k}$ for all $k \geq 0$

$T(n) = T(n/2) + 1$ $\quad n = \underline{2^k}$

sub $= 2^k$

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

$n = k-1$

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

$n = k-2$

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

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

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

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

since

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

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

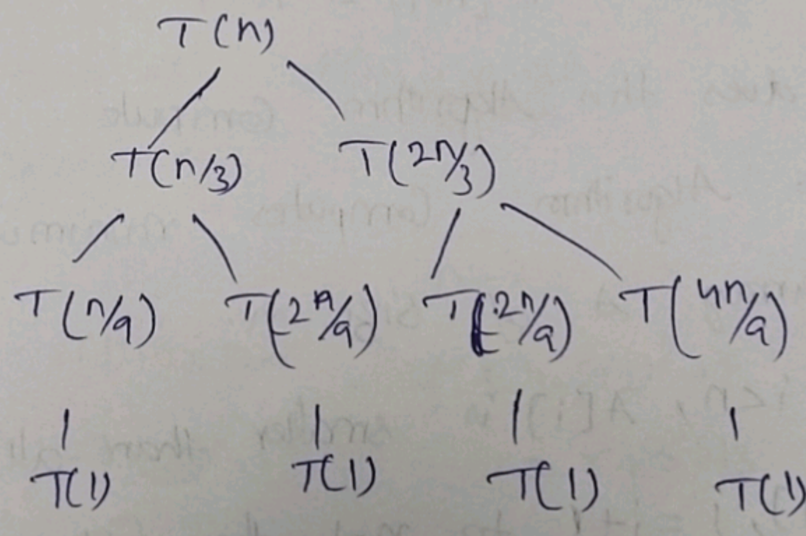
$T(n) = 1 + \log_2 n$

Time Complexity $= O(\log n)$

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

we use Recursion Tree method

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



C(n)

C(n/3) C(2n/3)

C(n/9) C(n/9) C(2n/9) C(4n/9)

c c c c

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 Compute

this Algorithm Computes minimum elements in an array A of size n.

if i<n, A[i] is smaller than all elements then, A[i], j = i+1 to n-1, then it returns A[j]

it also returns the leftmost minimum element

b) Mainly Comparison occurs during recursion and solve it?

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

$T(1) = 0$ [No Compares when n=1)

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

$= 0 + (n-1)$

$= n-1$

4) Analyse order of growth

1) $F(n) = 2n^2+5$ and $g(n) = 7n$. Use the $\Omega(g(n))$ notation

$F(n) = 2n^2+5$ $\quad F(n) \geq c \cdot g(n)$

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

n=1.

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

$g(1) = 7$

n=2

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

$= 8+5 = 13$

$g(2) = 7 \times 2 = 14$

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 c.g(n) when, 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.