1. If $f_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$ then $t_1(n)+t_2(n) \in O(\max\{g_1(n), g_2(n)\})$. prove the assertions

Solutions :-

By definition, there exist constants c_1, n, such that for all $n \geq n_1$; $t_1(n) < c_1 \cdot g_1(n)$

Similarly there exist constants c_1, n_1 such that for all $n \geq n_2$ $t_2(n) \leq c_2 \cdot g_2(n)$

Let $n_0 = \max(n_1, n_2)$ and $c = c_1 + c_2$ for all $n \geq n_0$:

$t_1(n) + t_2(n) \leq c_1 \cdot g_1(n) + c_2 \cdot g_2(n)$

By definition of maximum:

$g_1(n) \leq \max\{g_1(n), g_2(n)\}$

$g_2(n) \leq \max\{g_1(n), g_2(n)\}$

Thus,

$t_1(n) + t_2(n) \leq c_1 \max\{g_1(n), g_2(n)\} + c_2 \max\{g_1(n), g_2(n)\}$

$t_1(n) + t_2(n) \leq (c_1 + c_2) \max\{g_1(n), g_2(n)\}$

Hence

$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$

2. Big O Notation: Show that $f(n) = n^2 + 3n + 5$ is $O(n^2)$

Solution:-

To show $f(n) = n^2 + 3n + 5$ is $O(n^2)$.

$n^2 + 3n + 5 \leq c \cdot n^2$ $\quad$ $f(n) = n^2 + 3n + 5$

for $c = 2$ and $n_0 = 3$ $\quad$ $g(n) = cn^2$

$n^2 + 3n + 5 \leq 2n^2$

for all $n \geq 3$

$\therefore f(n) = n^2 + 3n + 5$ is $O(n^2)$

2. Find the time complexity of the recurrence equation

Let us consider such that recurrence for merge sort

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

By using master thoerem

$$T(n) = aT(n/b) + f(n)$$

where $a \geq 1$, $b \geq 1$ and $f(n)$ is positive function

Ex:- $T(n) = 2T(n/2) + n$

$a = 2$, $b = 2$ $f(n) = n$

By comparing of $f(n)$ with $n^{\log_b a}$

$\log_b a = \log_2 2 = 1$

Compare $f(n)$ with $n^{\log_b a}$

$f(n) = n$

$n^{\log_b a} = n^1 = n$

$\log_b a = 1$ $\quad T(n) = O(n^1 \log n) = O(n \log n)$

$T(n) = 2T(n/2) + n$ is $O(n \log n)$

3. $$T(n) = \begin{cases} 2T(n/2) + 1 & \text{if } n > 1 \\ 1 & \text{otherwise} \end{cases}$$

By Applying of masters thoerem

$T(n) = aT(n/b) + f(n)$ where $a \geq 1$, $b > 1$

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

Here $a = 2$, $b = 2$, $f(n) = 1$

If $f(n) = O(n^c)$ where $c < \log_b a$, then $T(n) = O(n^{\log_b a})$

If $f(n) = O(n^{\log_b a})$, then $T(n) = O(n^{\log_b a} \log n)$

If $f(n) = \Omega(n^c)$ where $c > \log_b a$ then $T(n) = O(f(n))$

lets calculate $\log_b a = \log_2 2 = 1$

$$f(n) = 1$$

$$n^{\log_b a} = n^1 = n$$

$f(n) = O(n^c)$ with $c < \log_b a$ (case I)

In this case $c = 0$ and $\log_b a = 1$

$c < 1$, so $T(n) = O(n^{\log_b a}) = O(n^1) = O(n)$

Time complexity of recurrence relation

$T(n) = 2T[n/2] + 1$ is $O(n)$

4. $T(n) = \begin{cases} 2T(n-1) & \text{if } n > 0 \\ 1 & \text{otherwise} \end{cases}$

Here, where $n = 0$

$$T(0) = 1$$

Recurrence relation analysis

for $n > 0$:

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

$T(n-1) \cdot 2T(n-2)$

$T(n-2) = 2T(n-3)$

$\vdots$

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

from this pattern

$T(n) = 2 \cdot 2 \cdot 2 \cdots 2 \cdot T(0) = 2^n \cdot T(0)$

since $T(0) = 1$, we have

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

The recurrence relation is

$T(n) = 2T(n-1)$ for $n > 0$ and $T(0) = 1$ is

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