① If $t_1(n) \in O(g_1(n))$ 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 that assertions.

Sol) we need to show that $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$. This means there exists a positive constant C and n_0 such that $t_1(n) + t_2(n) \le C$

$$t_1(n) \le c_1 g_1(n) \text{ for all } n \ge n_1$$

$$t_2(n) \le c_2 g_2(n) \text{ for all } n \ge n_2$$

Let $n_0 = \max\{n_1, n_2\}$ for all $n \ge n_0$

Consider $t_1(n) + t_2(n)$ for all $n \ge n_0$

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

we need to relate $g_1(n)$ and $g_2(n)$ to $\max\{g_1(n), g_2(n)\}$:

$g_1(n) \le \max\{g_1(n), g_2(n)\}$ and

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

Thus,

$$c_1 g_1(n) \le c_1 \max\{g_1(n), g_2(n)\}$$

$$c_2 g_2(n) \le c_2 \max\{g_1(n), g_2(n)\}$$

$$c_1 g_1(n) + c_2 g_2(n) \le c_1 \max\{g_1(n), g_2(n)\} + c_2 \max\{g_1(n), g_2(n)\}$$

$$c_1 g_1(n) + c_2 g_2(n) \le (c_1 + c_2) \max\{g_1(n), g_2(n)\}$$

$t_1(n)+t_2(n) \leq (c_1+c_2)\max\{g_1(n)\ g_2(n)\}$ for all $n \geq n_0$

By the defination of Big-O Notation

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

$$c = c_1+c_2$$

$t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, then

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

Thus, the assertion is proved.

② Find the Time complexity of the recurrence equation

Let us consider such that recurrence for merge sort.

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

By using master Thereom

$$T(n) = aT\left(\frac{n}{b}\right)+f(n)$$

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

<u>Ex</u>: $T(n) = 2T\left(\frac{n}{2}\right)+n$

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

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

$\log_b a = \log_2 2 = 1$

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

$f(n)=n$

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

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

In our case.

$\log_b a = 1$

$T(n) = O(n^1 \log n) = O(n \log n)$

Then time complexity of recurrence relation is

$T(n) = 2T(\frac{n}{2}) + n$ is $O(n \log n)$

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

Sol) By Appling OF master thereom

$T(n) = aT(\frac{n}{b}) + f(n)$ where $a \geq 1$
$b > 1$

$T(n) = 2T(\frac{n}{2}) + 1$

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

By comparision of $f(n)$ and $n^{\log_b a}$

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 $\log_b a$:

$\log_b a = \log_2 2 = 1$

$f(n) = 1$

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

$f(n) = O(n^c)$ with $c < \log_b a$ (case 1)

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 $n > 0$:

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

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

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

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

$\vdots$

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

From this pattern

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

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

$T(n) = 2^n$

(5) Big O Notation. show that $f(n) = n2 + 3n + 5$ is $O(n^2)$

Sol) $f(n) = O(g(n))$ means $c > 0$ and $n_0 \geq 0$

$f(n) \leq c \cdot g(n)$ for all $n \geq n_0$

given is $F(n) = n^2 + 3n + 5$

$c > 0$, $n_0, \geq 0$ such that $f(n) \leq c \cdot n^2$

$f(n) = n^2 + 3n + 5$

lets choose $c = 2$

$f(n) \leq 2 \cdot n^2$

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

$= 9n^2$

So, $c = 9$, $n_0 = 1$ $f(n) \leq 9n^2$ for all $n \geq 1$

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