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

Sol:

By definition, 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

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

2. find the time complexity the recurrence equation.

Let us consider such that recurrence for merge sort

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

by using master theorem

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

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

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

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

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

$\log_b^a = \log_2^2 = 1$

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

$f(n) = n$

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

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

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

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

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

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 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 $n>0$

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

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

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

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

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

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

Ans:

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

$n^2 + 3n + 5 < c \cdot n^2$

for $c = 2$ and $na = 3$ $\quad f(n) = n^2 + 3n + 5$

$n^2 + 3n + 5 \leq 2n^2$ $\quad g(n) = cn^2$