

1) If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then $f_1(n) + f_2(n) \in O(\max\{g_1(n), g_2(n)\})$ prove that assertion

we need to show that $f_1(n) + f_2(n) \in O(\max\{g_1(n), g_2(n)\})$ this means there exist a positive constant c and n_0 such that $f_1(n) + f_2(n) \leq c$

$$f_1(n) \leq c_1 g_1(n) \text{ for all } n \geq n_1$$

$$f_2(n) \leq c_2 g_2(n) \text{ for all } n \geq n_2$$

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

$$f_1(n) + f_2(n) \leq c_1 g_1(n) + c_2 g_2(n)$$

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

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

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

thus,

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

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

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

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

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

$$f_1(n) + f_2(n) \leq (c_1 + c_2) \max\{g_1(n), g_2(n)\} \text{ for all } n \geq n_0$$

By the definition of Big-O Notation

$$f_1(n) + f_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

$$c = c_1 + c_2$$

$f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then

$$f_1(n) + f_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

Thus the assertion proved

2. Find 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's theorem

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

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

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

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

By comparing $O(f(n))$ with $n \log_b a$

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

compare $f(n)$ with $n \log_b a$

$$f(n) = n$$

$$n \log_b a = n \cdot 1 = n$$

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

In our case

$$\log_b a = 1$$

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

The time complexity of recurrence relation is

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

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

By applying of master's theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \text{ where } \begin{matrix} a \geq 1 \\ b \geq 1 \end{matrix}$$

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

$$\text{Here } a=2, b=2, f(n)=1$$

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

Let's 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 $\log_b a = 1$

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

Time complexity of recurrence relation

$$T(n) = 2T(n/2) + 1 \text{ is } O(n)$$

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

Here, where $n = 0$

$$T(0) = 1$$

Recurrence relation Analysis

for $n > 0$

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

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

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

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

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

Even this problem

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

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

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

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

$f(n) = O(g(n))$ 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 $f(n) \leq c \cdot n^2$

$$f(n) =$$

Let choose $c = 2$

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

$$\begin{aligned} f(n) = n^2 + 3n + 5 &\leq n^2 + 3n^2 + 5n^2 \\ &= 9n^2 \end{aligned}$$

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

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