ASSIGNMENT => 2

Name :- L. Datta Kumar

Reg no :- 192324141

Subject :- Design and Analysis of Algorithm

Subject code :- CSA0670

Course :- CSE (AIDS)

Faculty Name :- Heion Kelvin

No. of Pages :- 3.

Max Marks :-

Marks Obtained :-

Signature

1. If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then $f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\})$. Prove the assertions.

sol:- By definition there exist constant c_1, n_1 such that for all $n \geq n_1$; $f(n) \leq c_1 g_1(n)$.

similarly there exist constant c_2, n_2 such that for all $n \geq n_2$; $f_2(n) \leq c_2 \cdot g_2(n)$.

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

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

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

$$f_1(n) + f_2(n) \leq c_1 \max\{g_1(n), g_2(n)\} + c_2 \max\{g_1(n), g_2(n)\}$$

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

Hence

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

2. Find the time complexity of the recurrence equation.

sol:- Let us consider such that recurrence for merge sort.

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

by using master's theorem

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

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

e.g.:- $T(n) = 2T(n/2) + n$.

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

by comparing of $f(n)$ with $n.\log_b a$

$\log_b a = \log_2 2 = 1$

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

$f(n)=n$

$n.\log_b a = n^1 = n$

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

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

---

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

sol: By applying of master theorem

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

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

sol: 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 $T(n) = 2.2.2.\ldots\ldots 2T(0) = 2^n.T(0)$.

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 $T(n)=2^n$

$T(n) = 2^n$.

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

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

$n^2+3n+5 \le cn^2$

for $c=2$ and $n_0=3$

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

for all $n \ge 3$

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

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

$g(n)=c.n^2$