National University of Computer and Emerging Sciences, Lahore Campus

Course:	Design & Analysis of Algorithms	Course Code:	CS2009
Program:	BS (Computer Science)	Semester:	Spring 2023
Duration:	20 Minutes	Total Marks:	15
Paper Date:	21-Feb-2023	Weight	2.5
Section:	J	Page(s):	2
Exam:	Quiz 1	Reg. No.	

Instruction/Notes:

Question 1: [5 marks]

For the following functions f(n) and g(n), indicate whether f(n) = O(g(n)), f(n) = Ω(g(n)), or both, i.e. f(n) = θ(g(n)). Justify your answer.

$f(n) = n^2 \log n$

$f(n) = \theta(g(n))$

$g(n) = 2^{1/2} n^2 \log(2^{1/2} n)$

$g(n) = 2^{1/2} n^2((\log 2^{1/2}) + (\log n))$

$g(n) = \underbrace{2^{1/2}}_{\text{constant}} (n^2 \underbrace{\log 2^{1/2}}_{\text{constant}} + n^2 \log n)$

$g(n) = c_1 (n^2 c_2 + n^2 \log n)$

→ Analyzing g(n) gives $n^2 \log n$ as dominant factor i.e. there exist constants c_m and c_n for which;

$c_m (n^2 \log n) \leq 2^{1/2} n^2 \log(2^{1/2} n) \leq c_n (n^2 \log n)$

Question 2: [5 Marks]

Find big-theta of the function $f(n) = n/18 - 19n^{1/2} + 20$, give the constants c_1, c_2, n_0

let $g(n) = n$

Then we can find constants by,

$c_1 n \leq \frac{n}{18} - 19n^{1/2} + 20 \leq c_2 n$

$c_1 \leq \frac{1}{18} - \frac{19}{n^{1/2}} + \frac{20}{n} \leq c_2$

Let $n_0 = 1$

$c_1 \leq \frac{1}{18} - 19 + 20 \leq c_2$

$c_1 \leq \frac{19}{18} \leq c_2$

So, $n_0 = 1$

$c_1 = 1$

$c_2 = 2$

Question 3: [5 marks]

Write down the Running Time equation (T(n)) of the following algorithm and analyze its time complexity. Show complete steps. If you make any assumptions, state them clearly.

```
int algo(input, n)
{
If (n<=0)                      — O(1)
        {return 0}             — O(1)
X = algo(A, n/2)               — T(n/2)
Y = algo(A, n/4)               — T(n/4)
Z = A[(n/2) + (n/4) +1]          O(1)
return (X+Y+Z)                   O(1)
}
```

$$T(n) = T(n/2) + T(n/4) + c \quad \text{— (i)}$$

$$T(n/2) = T(n/4) + T(n/8) + c$$

$$T(n/4) = T(n/8) + T(n/16) + c$$

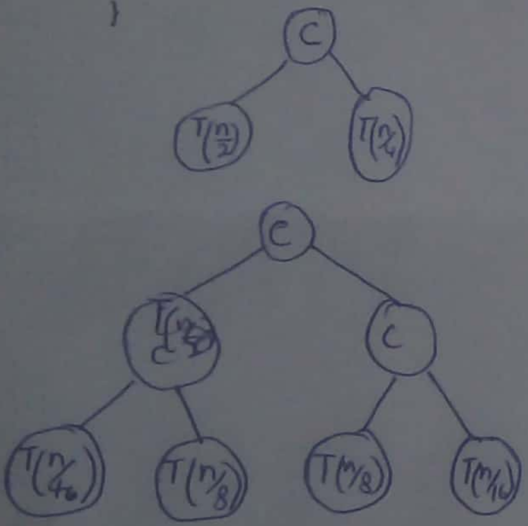
putting back in (i)

$$T(n) = T(n/4) + 2(T(n/8)) + T(n/16) + 2c + c$$

$$T(n) = c + 2c + \dots + 2^{\log_2 n} c$$

$$T(n) = c\left(\frac{1-2^{\log_2 n}}{1-2}\right) = c\left(\frac{1-n}{-1}\right) = (n-1)c$$

or $O(n)$



...

$$T(n) = c + 2c + 4c + \dots + 2^k c$$

Here $k \approx \log_2 n$

$$T(n) = 2^0 c + 2^1 c + 2^2 c + \dots + 2^{\log_2 n} c$$

$$T(n) = \left(\sum_{i=0}^{\log n} 2^i\right) c$$

$$T(n) = c\left(\frac{2^{\log_2 n} - 1}{2-1}\right) = (2^{\log_2 n} - 1)c = (n^{\log_2 2} - 1)c = n$$

$$T(n) = O(n)$$