USN | | | | | | | | | | |

Solution for Internal Assessment Test 1 – March 2019

Sub:	Design & Analysis of Algorithms					Sub Code:	17CS43	Branch:	CSE
Date:	06/03/2019	Duration:	90 min's	Max Marks:	50	Sem / Sec:	4/A,B ,C& D		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1	Write an algorithm to find maximum element in an array of n elements. Give the mathematical analysis of this non-recursive algorithm.	[10]	**CO1, CO2, CO3**	**L2**

```
MaxElement (A[0..n-1])
Maxval = A[0]
For i = 1 to n-1 do
    if A[i] > Maxval
        Maxval = A[i]
return Maxval
```

The innermost comparison is the basic operation. If C(n) denotes the number of times this basic operation is executed, then C(n) = sigma (i from 1 to n-1) 1 = n-1 belongs to theta(n)

		MARKS	CO	RBT
2	Explain divide and conquer technique. Write a recursive algorithm for finding the maximum and minimum element from the list.	[10]	**CO2, CO3**	**L1**

1. The problem's instance is divided into several smaller instances of the same problem
2. The smaller instances are solved (Typically recursively)
3. If necessary, the smaller instance solutions are combined to get a solution of the overall problem

Recursively, T(n) = aT(n/b) + f(n), where T(n) is the running time, n can be divided into b instances of size n/b, with a of them needing to be solved and f(n) is the time taken to divide the problem and combine the solutions.

MaxMin(i, j, max, min)
// a[1: n] is a global array; i and j are integers 1<= i <= j <= n. max and min get set to largest and smallest values in a[i : j] respectively

```
Begin
  If (i = j) then max = min  = a[i]
  Else  if (i = j – 1) then
    Begin
      If (a[i] < a[j]) then
          max = a[j]; min = a[i];
      Else
          max = a[i]; min = a[j];
    End
  Else
    Begin
      mid = floor((i + j)/2)
```

```
    MaxMin(i, mid, max, min);
    Maxmin(mid+1, j, max1, min1)
    If (max < max1) then max = max1;
    If (min > min1) then min = min1;
  End
End
```

Initially, the above is called with MaxMin(1, n, x, y)

Q. No.	Question	Marks	CO	Level
3	Define three asymptotic notations and from the following equalities prove if it is incorrect or correct using the definitions of asymptotic notations i) $6n^2-8n=\Theta(n^2)$ ii) $12n^2+8=O(n)$ iii) $3n^2\log n=\Theta(n^2)$	[10]	**CO2**	**L3**

A function t(n) is said to be in O(g(n)) if there exist some positive number c and some non-negative integer n_0 such that t(n) <= cg(n) for all n >= n_0

A function t(n) is said to be in Omega(g(n)) if there exist some positive number c and some non-negative integer n_0 such that t(n) >= cg(n) for all n >= n_0

A function t(n) is said to be in Theta(g(n)) if there exist some positive numbers c_1 and c_2 and some non-negative integer n_0 such that c_2g(n) <= t(n) <= c_1g(n) for all n >= n_0

i) is true, ii) and iii) are false

Q. No.	Question	Marks	CO	Level
4	Design a recursive algorithm for solving tower of Hanoi problem and give the general plan of analyzing that algorithm.	[10]	**CO2, CO3**	**L2**

```
Tower (n, s, d)
// move n disks from peg s to peg d
// disks are numbered from 1 to n, 1 occupying the highest position, n the lowest
If (n = 1) move disk 1 from s to d
Else
  Tower (n-1, s, i)
  Move disk n from s to d
  Tower (n-1, i, d)
```

Initially the above may be called with s = 1, i = 2, d = 3

Solve the recurrence relation $T(n) = 2T(n-1) + 1$ which leads to $T(n) = 2^n - 1$

Q. No.	Question	Marks	CO	Level
5	Design an algorithm for binary search, give an example. Show that the worst case efficiency of binary search is $\Theta(\log n)$.	[10]	**CO3, CO4**	**L3**

```
BinarySearch (A[0..n-1], K)

while (l <= r) do
    m = floor((l+r)/2)
    if (K = A[m]) return m
    else if (K < A[m]) r = m-1
    else l = m+1
return -1
```

Recurrence relation is $C_{worst}(n) = C_{worst}(floor(n/2)) + 1$, $n > 1$, $C_{worst}(1) = 1$

Solving the above gives Cworst(n) = floor($\log_2 n$) + 1 which implies that the efficiency of Binary Search is Theta(log n)

6	Write an algorithm for merge sort. Analyze its efficiency. Mergesort (A[0..n-1]) if (n > 1) copy A[0..floor(n/2) – 1] to B[0..floor(n/2) – 1] copy A[floor(n/2)..n – 1] to C[0..ceiling(n/2) – 1] Mergesort(B[0..floor(n/2) – 1]) Mergesort(C[0..floor(n/2) – 1]) Merge(B, C, A) $C(n) = 2C(n/2) + C_{merge}(n)$, n > 1, C(1) = 0 In the worst case, $C_{merge}(n) = n – 1$ and so $C_{worst}(n) = 2C_{worst}(n/2) + n – 1$, n > 1, $C_{worst}(1) = 0$. Solving this gives $C_{worst}(n) = n\log n – n + 1$	[10]	**CO2, CO3, CO4**	**L4**
7(a)	Define an algorithm. Discuss the criteria of an algorithm with an example. An algortihm is a sequence of unambiguous instructions to solve a problem. Criteria: The unambiguity requirement is essential The range of inputs for which the algorithm works have to be specified The algorithm terminates after a finite number of steps The instructions may be effective so that they may be carried out Zero or more inputs have to be given One or more outputs have to be produced	[5]	**CO1**	**L1**
(b)	Prove that: If $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$ then $t_1(n)+ t_2(n) \in O(\max\{g_1(n), g_2(n)\})$ Given in Levitin book	[5]	**CO1**	**L3**
8	Explain about Master's theorem. Solve the following using substitutions and Master's theorem i) $T(n)=2T(n/2)+n$,$T(1)=2$ ii) $T(n)= 9T(n/3)+4n^6$, $T(1)= 1$ If f(n) belongs to $O(n^d)$ with d >= 0 in the recurrence equation T(n) = aT(n/b) + f(n), then T(n) belongs to $O(n^d)$, if $a < b^d$ T(n) belongs to $O(n^d \log n)$ if $a = b^d$ T(n) belongs to $O(n^{\log_b a})$ if $a > b^d$	[10]	**CO1**	**L3**

.
T(n) = O, Omega or Theta (nlogn)
T(n) = O, Omega or Theta (n^6)