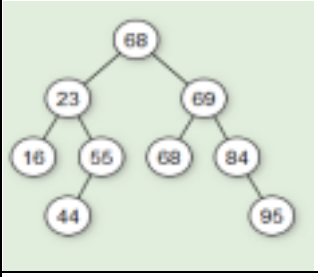
USN | | | | | | | | | | |

Internal Assessment Test 1 – June 2024

Sub:	ANALYSIS & DESIGN OF ALGORITHMS					Sub Code:	BCS401	Branch:	AIML/AIDS
Date:	06.05.24	Duration:	90 minutes	Max Marks:	50	Sem/Sec:	IV -A, B, C		OBE

		Answer any FIVE FULL Questions	MARKS	CO	RBT
1.	a)	What is an algorithm? Write Euclid's algorithm to find the GCD of two numbers.	5M	CO1	L1
	b)	Evaluate the following: I. $f(n) = 10n^2 + 4n + 2$, prove that $f(n) = O(n^2)$ II. $f(n) = 100n + 5$, prove that $f(n) = \Omega(n)$	5M	CO1	L3
2	a)	Define the Mathematical Analysis of Recursive Algorithms with the help of an example.	5M	CO1	L2
	b)	Define Towers of Hanoi problem and describe the time complexity.	5M	CO1	L2
3	a)	Write an algorithm to sort an array using bubble sort and analyze the same for time complexity. Express using asymptotic notations.	10M	CO1	L2
4	a)	Compare Merge Sort and Binary Search algorithms with respect to their time complexities.	10M	CO2	L2
5	a)	Write the Inorder, Postorder & Preorder traversals of the following tree: 	5M	CO2	L2
	b)	Write the algorithm for Selection sort.	5M	CO1	L1
6	a)	Partition the array [10, 2, 4, 14, 5, 6, 11, 15, 3, 20] using the Quick Sort algorithm.	5M	CO2	L3
	b)	What is the Recurrence relation of the Binary Search algorithm? What is the analysis of the Best and Worst cases?	5M	CO2	L2

CI **CCI** **HOD**

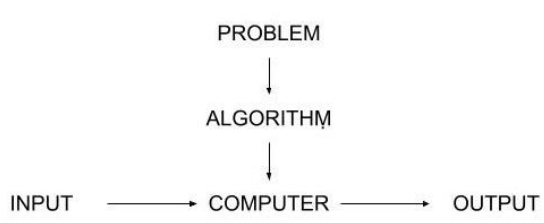
----------All the Best----------

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

Internal Assessment Test 1 – June 2024

Sub:	ANALYSIS & DESIGN OF ALGORITHMS					Sub Code:	BCS401	Branch:	AIML/AIDS
Date:	06.05.24	Duration:	90 minutes	Max Marks:	50	Sem/Sec:	IV -A, B, C		OBE

		Answer any FIVE FULL Questions	MARKS	CO	RBT

1.	a)	What is an algorithm? Write Euclid's algorithm to find the GCD of two numbers.
		
		Ans: An algorithm is defined as a finite sequence of unambiguous instructions followed to accomplish a given task. It is a step by step procedure to solve a given problem in a finite number of steps by accepting a set of inputs and producing the desired result.
		
		 
		
		Algorithm Euclid(m, n)
		//Computes gcd(m,n)
		//Input: m and n, positive integers.
		//Output: GCD of m and n.
		while n != 0 do
		r <- m mod n
		m <- n
		n <- r
		return m
			5M	CO1	L1

	b)	Evaluate the following: I. $f(n) = 10n^2 + 4n + 2$, prove that $f(n) = O(n^2)$ II. $f(n) = 100n + 5$, prove that $f(n) = \Omega(n)$ Ans: Given $f(n) = 10n^2 + 4n + 2$,	5M	CO1	L3

		Replace $4n$ and 2 with n^2. $$cg(n) = 10n^2 + n^2 + n^2$$ $$= 12n^2$$ $f(n) \in O(n)$ only if $f(n) \leq cg(n)$ for all $n \geq n_0$ $10n^2 + 4n + 2 \leq 12n^2$ for all $n \geq 3$ where $c = 12$. Therefore, by definition $f(n) = O(n^2)$ Ans: Given $f(n) = 100n + 5$, Replace 5 with n. $$cg(n) = 100n + n$$ $$= 101n$$ $f(n) \in \Omega(n)$ only if $f(n) \geq cg(n)$ for all $n \geq n_0$ $100n + 5 \geq 101n$ for all cannot be satisfied for any $n \geq n_0$ Therefore, by definition $f(n) \notin \Omega(n)$			
2	a)	Define the Mathematical Analysis of Recursive Algorithms with the help of an example. Ans: Recursion is a method of solving the problem where the solution to a problem depends on solutions to smaller instances of the same problem. A recursive function is a function that calls itself during execution.	5M	CO1	L2

Example :

```
int fact (int n)
{
    if (n == 0)
        return 1;
    else
        return (n * fact(n-1));
}
```

Here consider n = 1,

if n = 1, then 1! = 1 * fact(0)

0! = 1

Factorial of n = 0 returns 1, therefore 1! = 1 * 1.

= 1

The Base case is when n = 0.

$n! = 1$ if $n = 0$.

Consider n = 5,

$5! = 5 * 4!$

$4! = 4 * 3!$

$3! = 3 * 2!$

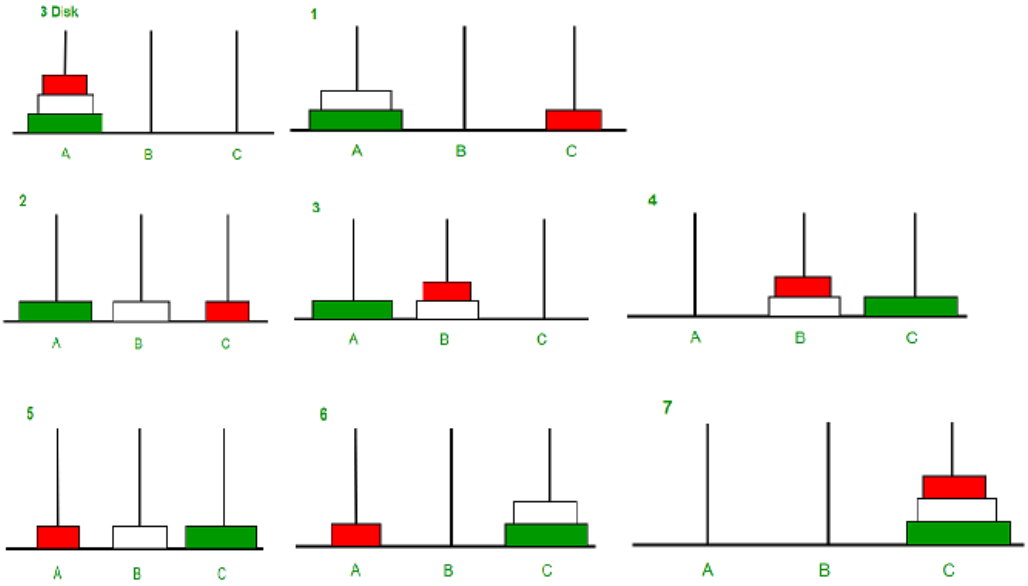
$2! = 2 * 1!$

$1! = 1 * 0!$

Therefore $n! = n * (n-1)!$ when $n > 0$

$$F(n) = \begin{cases} 1 & \text{if } n = 0 \\ n * F(n-1) & \text{otherwise} \end{cases}$$

	b)	Define Towers of Hanoi problem and describe the time complexity. Ans: In the Tower of Hanoi problem, there are 3 poles A, B & C. There are n disks of different sizes and they are placed in such a way that the smaller disk is placed on the disk of larger size. The smallest disk will be on top and the largest disk will be at the bottom. The two other poles B & C are empty. The disks need to be transferred from pole A to pole C using pole B as an auxiliary pole. The following rules need to be applied: - Only one disk can be moved at a time from one pole to another.	5M	CO1	L2

		- A larger disk cannot be placed on top of a smaller disk. - Only one pole can be used as an intermediate at a time.  $C(n) = \begin{cases} 1 & \text{if } n=1 \\ C(n-1) + 1 + C(n-1) & \end{cases}$ ① This represents the ~~no of disks~~ movements of disks from Source to temp. ② This represents the movement of disk from source to destination ③ This represents movement of disks from temp to destination.			

3	a)	Write an algorithm to sort an array using bubble sort and analyze the same for time complexity. Express using asymptotic notations.
		
		Ans:
		
		Algorithm BubbleSort(A[], n)
		
		//Purpose: Sorting of array
		//Input: A - Array of elements, n - number of elements in array
		//Output: Sorted Array
			10M	CO1	L2

		for j<- 1 to (n-1) do *for i<- 0 to n-j-1 do* *if(A[i] > A[i+1])* *temp <- A[i]* *A[i] <- A[i+1]* *A[i+1] <- temp* *end if* *end for* *end for* In Bubble Sort, there are three different scenarios, each with a unique time complexity: 1. Best case scenario: The best case scenario occurs when the input list is already sorted. In this case, Bubble Sort performs (n-1) comparisons and zero swaps, leading to a time complexity of $O(n)$. 2. Average case scenario: The average case scenario happens with randomly arranged data. The number of swaps and comparisons is roughly half the total number of pairs, leading to a time complexity of $O(n^2)$. 3. Worst case scenario: The worst case scenario occurs when the input list is sorted in the exact opposite order. In this case, every pair of adjacent elements is swapped, leading to a time complexity of $O(n^2)$. Comparisons in the first pass: n-1 Comparisons in the second pass: n-2 Comparisons in the third pass: n-3 ... Comparisons in the last pass: 1 So, the total number of comparisons = (n-1)+(n-2)+(n-3)+...+1 = n*(n-1)/2 which is equivalent to $O(n^2)$. $C(n) \in O(n^2)$.			
4	a)	Compare Merge Sort and Binary Search algorithms with respect to their time complexities. $C(n) = \begin{cases} 0 & \text{if } n=1 \\ 2C(n/2)+n & \text{otherwise.} \end{cases}$	10M	CO2	L2

The time complexity of Merge sort algorithm is given by

$$T_1(n) = c_1(n/2) + c_1(n/2) + n$$

$$= 2c_1(n/2) + n$$

$$= 2\left[2c_1(n/4) + n/2\right] + n$$

$$= 2^2 c_1(n/2^2) + 2n$$

$$= 2^2 c_1(n/2^2) + 2n$$

$$= 2^2\left[2c_1(n/2^3) + n/2^2\right] + 2n$$

$$= 2^3 c_1(n/2^3) + 3n$$

$$= \vdots$$

$$= 2^i c_1(n/2^i) + n.i$$

Let $2^i = n$

$$c_1(n) = n c_1(n/n) + ni$$

$$= n c_1(1) + ni = n.i$$

Taking log,

$$i \log 2 = \log n$$

$$i = \log_2 n$$

$$T_1(n) = n \log n \qquad T_1(n) \in \Theta(n \log n)$$

The time complexity of Binary Search is given by

$$C_2(n) = \begin{cases} 1 & \text{if } n=1 \\ C(n/2) + 1 & \text{otherwise.} \end{cases}$$

$C_2(n) = C_2(n/2) + 1$

$= C_2(n/2) + 1 + 1$

$= C_2(n/2^2) + 2$

$= C_2(n/2^3) + 3$

$= C_2(n/2^4) + 4$

$= i + C(n/2^i)$

Let $2^i = n$;

$C_2(n) = i + C_2(n/n)$

$= i + C(1)$

$= i + 0$

$C_2(n) = i$

Taking log,

$\therefore \log 2^i = \log n$

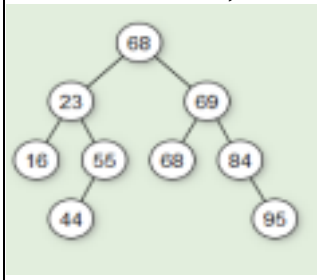
$\therefore i \log 2 = \log n$

$\therefore \log_2 n$

$C_2(n) = \log n$ $\quad$ $C_2(n) \in O(\log n)$

* Comparing the time complexities of Mergesort & Binary Search, the time complexity of Mergesort is greater than that of Binary Search

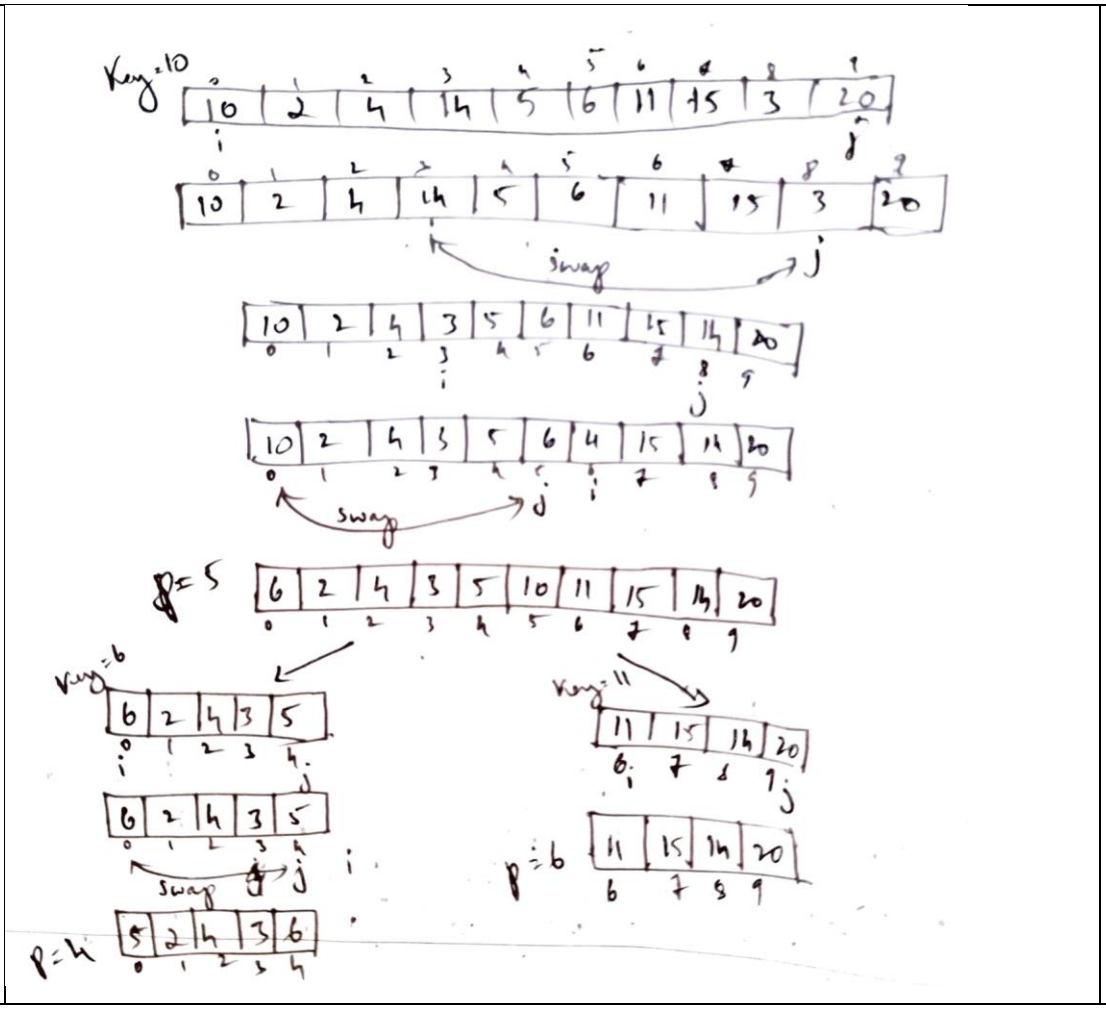
$C_1(n) > C_2(n)$ (or) $n \log n > \log n$

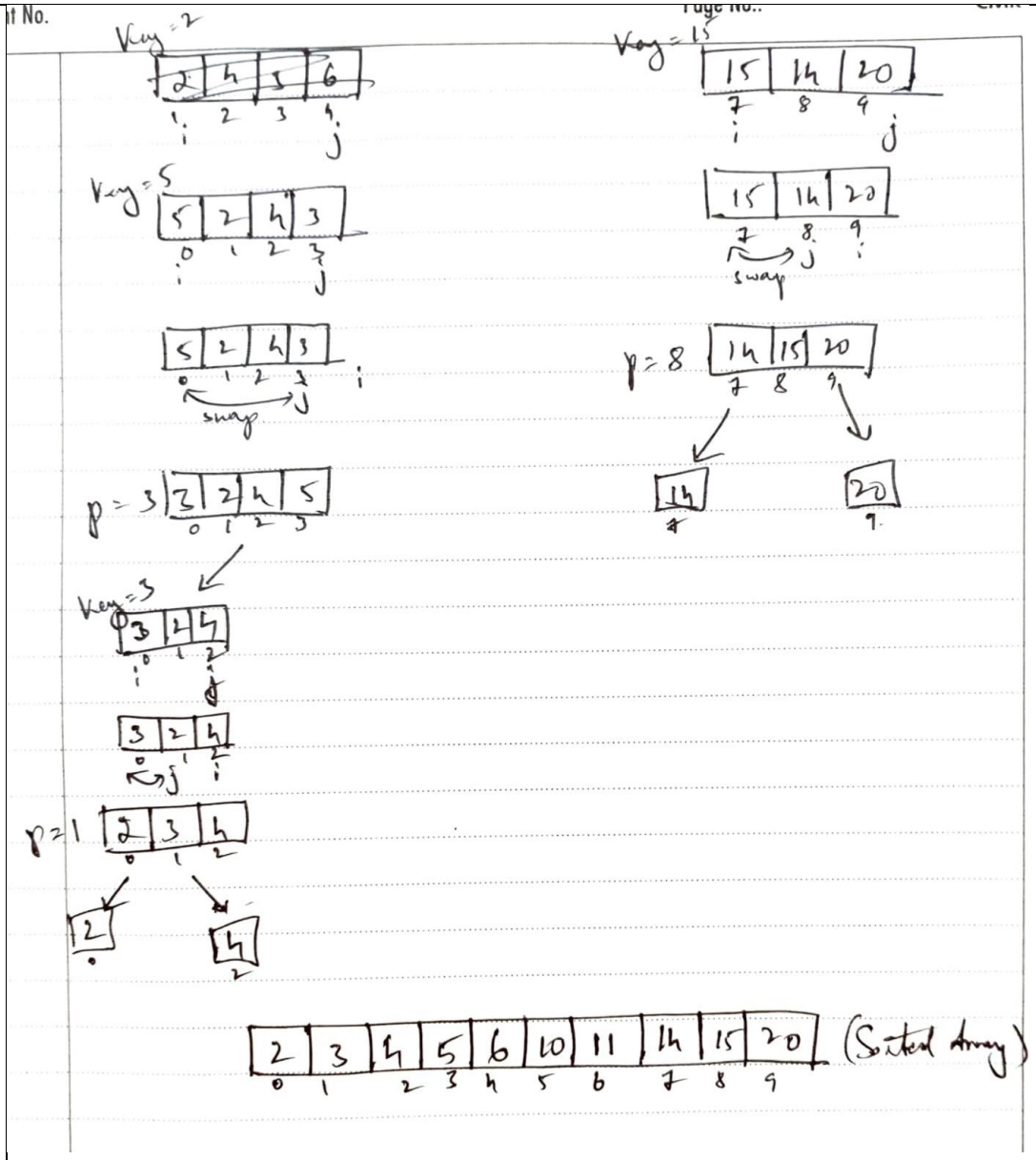
5	a)	Write the Inorder, Postorder & Preorder traversals of the following tree: 	5M	CO2	L2

		5a) Inorder: (VLR) => 68, 23, 16, 55, 44, 69, 68, 84, 95 Postorder (LRV): 16, 44, 55, 23, 68, 95, 84, 69, 68. Inorder (LVR): 16, 23, 44, 55, 68, 68, 69, 84, 95			

	b)	Write the algorithm for Selection sort.
		
		Ans:
		Algorithm SelectionSort(A[], n)
		
		//Purpose: To sort the given elements in ascending order.
		//Input: A - Array of elements, n - size of the no. of elements in the array.
		//Output: Sorted array.
		
		for i <- 0 to n-2 do
		pos <- i
		for j <- i+1 to n-1 do
		if(A[j] < A[pos])
		pos <- j
		end for
		temp <- A[pos]
		A[pos] <- A[i]
		A[i] <- temp
		end for
			5M	CO1	L1

6	a)	Partition the array [10, 2, 4, 14, 5, 6, 11, 15, 3, 20] using the Quick Sort algorithm. Ans:	5M	CO2	L3


Key=10
swap
p=5
Key=6
Key=11
swap
p=4
p=6

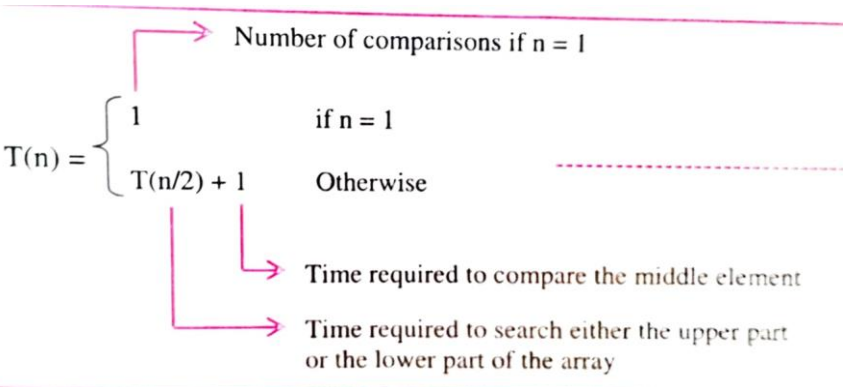


b) What is the Recurrence relation of the Binary Search algorithm? What is the analysis of the Best and Worst cases?

5M | CO2 | L2

Ans:

The recurrence relation of binary search algorithm is as follows:



Analysis of Best Case:

The best case occurs when the element to be searched for is present in the middle of the array. In such a case, the total no. of comparisons will be 1.

		Analysis of Worst Case: The worst case occurs when maximum no. of comparisons are done to search the element.			

CI **CCI** **HOD**

----------All the Best----------