KONGU ENGINEERING COLLEGE, PERUNDURAI 638 060

CONTINUOUS ASSESSMENT TEST 1

(Answer Key)

(Regulations 2020)

Month and Year : March 2023	Roll Number:
Programme : B.E Branch : CSE Semester : IV	Date : 08.03.2023 Time : 2.30pm to 04.00pm
Course Code : 20CST44 Course Name : Design and Analysis of Algorithms	Duration : 1 ½ Hours Max. Marks : 50

PART - A (10 × 2 = 20 Marks)

1. **Apply Euclid's algorithm to find GCD(240,16).**
GCD(m,n) = GCD(n, m mod n)
GCD(240,16) = GCD(16,0) Hence GCD is 16.

2. **Assume the basic operation count for an algorithm C(n)= n(n+1)/2, How much time will the algorithm run if we increase the input size by 4 times?**
If we increase the input size by 4 times, the running time will increase by 4 times.

3. **Find the Order of growth of:**

$\sum_{i=1}^{n} (2i+10)$

$=2(1+2+3+\ldots.n) + 10(n-1+1)$

$=2n(n+1)/2 +10n = n(n+1)+10n = n^2+11n \in \Theta(n^2)$

4. **Compare the order of growth of $n^2(n+1)$ and n^3 using limits.**

$$\lim_{n\to\infty} \frac{n^2(n+1)}{n^3} = \lim_{n\to\infty} \frac{n^3+n^2}{n^3}$$

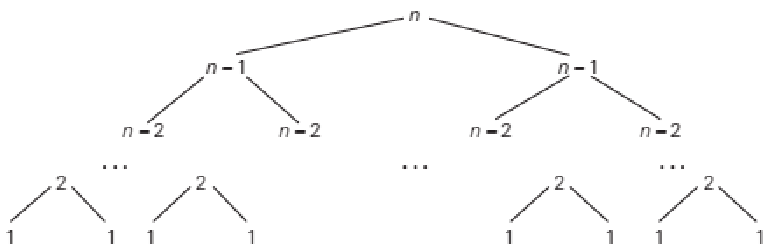
$$= \lim_{n\to\infty} \left(1+\frac{1}{n}\right)$$

$$= 1$$

Since positive constant, the functions have same order of growth.

$\therefore n^2(n+1) \in \Theta(n^3)$

5. **Ascertain the order of growth for the Tower of Hanoi puzzle by constructing tree of recursive calls made by the recursive algorithm.**



$C(n) = \sum_{l=0}^{n-1} 2^l$ (where l is the level in the tree

6. **Write any four desirable features of algorithm animation user interface.**
   - Easy, User Friendly
   - Understandable, Interactive

7. **Mention the best case, worst case and average case complexities of linear search.**
   - The algorithm makes the largest number of key comparisons among all possible inputs of size *n*: *Cworst(n)* = *n.*
   - *Cbest(n)* = 1 where the element is found at the first index.
   - Linear search makes **n/2** comparisons on an average where n is the number of elements.

8. **Define Convex Hull problem.**

The *convex-hull problem* asks to find the smallest convex polygon that would include all the points of a given set. The *convex hull* of a set *S* of points is the smallest convex set containing *S*.

9. **Write the pseudocode for Bubble sort algorithm using Brute Force technique.**

ALGORITHM *BubbleSort*($A[0..n-1]$)

```
//Sorts a given array by bubble sort
//Input: An array A[0..n − 1] of orderable elements
//Output: Array A[0..n − 1] sorted in nondecreasing order
for i ← 0 to n − 2 do
    for j ← 0 to n − 2 − i do
        if A[j + 1] < A[j] swap A[j] and A[j + 1]
```

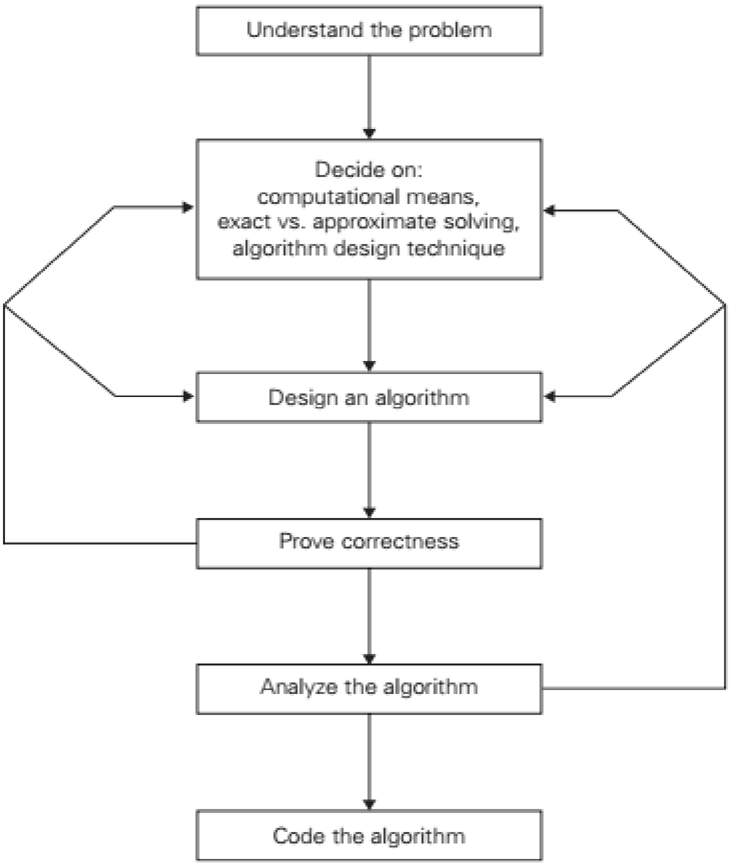
10. **Find the number of comparisons needed to search a string "Engineering" in the text "England Engineering College".**

England Engineering College
Engineering
Engineering
Engineering
Engineering
Engineering
Engineering
Engineering
Engineering
Engineering
Engineering

The number of comparisons performed = 22

Part – B (3 × 10 = 30 Marks) ANSWER ANY THREE QUESTIONS

11. **Summarize the sequence of steps involved in designing and analysing an algorithm.** (10)



12. **Design and analyse the time complexity of recursive and non-recursive algorithms to compute the factorial of the given number ‘n’.** (10)

Non-recursive algorithm: (2m)

ALGORITHM *Fact(n)*
//Determines the factorial of given element
//Input: An element n
//Output: the factorial of given element
fact← 1
for *i* ← 1 **to** *n* **do**
fact ← fact*i
return *fact*

Analysis: (3m)

* Input size is n
* Basic operation is multiplication.

$$M(n) = \sum_{i=1}^{n} 1$$
$$= n - 1 + 1 = n$$
$$M(n) = n \in \Theta(n)$$

Recursive algorithm: (2m)

ALGORITHM $F(n)$
//Computes $n!$ recursively
//Input: A nonnegative integer n
//Output: The value of $n!$
if $n = 0$ **return** 1
else return $F(n-1) * n$

Analysis: (3m)

Set up the recurrence relation and initial condition for the algorithm’s number of multiplications *M(n)*:
$M(n) = M(n - 1) + 1$ for $n > 0$,
$M(0) = 0$.

$M(n) = M(n - 1) + 1$
$= [M(n - 2) + 1] + 1 = M(n - 2) + 2$ — substitute $M(n - 1) = M(n - 2) + 1$
$= [M(n - 3) + 1] + 2 = M(n - 3) + 3.$ — substitute $M(n - 2) = M(n - 3) + 1$

$M(n) = M(n - 1) + 1 = . . . = M(n - i) + i = . . . = M(n - n) + n = n.$

13. **Write the recursive algorithm for finding n^{th} Fibonacci term. Find its time efficiency.** (10)

We consider the ***Fibonacci numbers***, a famous sequence
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, . . .
that can be defined by the simple recurrence
$F(n) = F(n - 1) + F(n - 2)$ for $n > 1$
and two initial conditions
$F(0) = 0, F(1) = 1$ **(2m)**

We can take advantage of a theorem that describes solutions to a ***homogeneous second-order linear recurrence with constant coefficients***
$ax(n) + bx(n - 1) + cx(n - 2) = 0$

$$F(n) = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n),$$

(2m)

ALGORITHM *F (n)*
//Computes the *n*th Fibonacci number recursively by using its definition
//Input: A nonnegative integer *n*
//Output: The *n*th Fibonacci number
if $n \le 1$ **return** *n*
else return $F(n - 1) + F(n - 2)$ **(2m)**

		we get the following recurrence for $A(n)$: $A(n) = A(n-1) + A(n-2) + 1$ for $n > 1$, $A(0) = 0, A(1) = 0$. We can reduce our inhomogeneous recurrence to a homogeneous one by rewriting it as $[A(n) + 1] - [A(n-1) + 1] - [A(n-2) + 1] = 0$ and substituting $B(n) = A(n) + 1$: $B(n) - B(n-1) - B(n-2) = 0$, $B(0) = 1, B(1) = 1$. **(2m)** But it can actually be avoided by noting that $B(n)$ is, in fact, the same recurrence as $F(n)$ except that it starts with two 1's and thus runs one step ahead of $F(n)$. So $B(n) = F(n+1)$, and $A(n) = B(n) - 1 = F(n+1) - 1 = \frac{1}{\sqrt{5}}(\phi^{n+1} - \hat{\phi}^{n+1}) - 1.$ **(2m)**	

14.		**Design selection sort algorithm and apply this algorithm to sort the given list in ascending order. Also, find its time complexity.**
		3, 7, 1, 8, 23, 12, 17, 14, 28, 45, 32
		
		ALGORITHM *SelectionSort*$(A[0..n-1])$
		//Sorts a given array by selection sort
		//Input: An array $A[0..n-1]$ of orderable elements
		//Output: Array $A[0..n-1]$ sorted in nondecreasing order
		for $i \leftarrow 0$ **to** $n-2$ **do**
		$min \leftarrow i$
		for $j \leftarrow i+1$ **to** $n-1$ **do**
		if $A[j] < A[min]$ $min \leftarrow j$
		swap $A[i]$ and $A[min]$
		(3m)
		Problem:
		| 3 7 **1** 8 23 12 17 14 28 45 32
		1| 7 **3** 8 23 12 17 14 28 45 32
		1 3| **7** 8 23 12 17 14 28 45 32
		1 3 7 | **8** 23 12 17 14 28 45 32
		1 3 7 8 | 23 **12** 17 14 28 45 32
		1 3 7 8 12 | 23 17 **14** 28 45 32
		1 3 7 8 12 14 | **17** 23 28 45 32
		1 3 7 8 12 14 17 | **23** 28 45 32
		1 3 7 8 12 14 17 23 | **28** 45 32
		1 3 7 8 12 14 17 23 28 | 45 **32**
		1 3 7 8 12 14 17 23 28 32 | 45 **(3m)**
		The analysis of selection sort is straightforward. The input size is given by the number of elements n; the basic operation is the key comparison $A[j] < A[min]$. The number of times it is executed depends only on the array size and is given by the following sum:
		$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2}[(n-1) - (i+1) + 1] = \sum_{i=0}^{n-2}(n-1-i).$
		$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2}(n-1-i) = \frac{(n-1)n}{2}.$
		Thus, selection sort is a $\Theta(n^2)$ algorithm on all inputs. **(4m)**
			(10)