CSE 3131: ALGORITHM DESIGN 1

ASSIGNMENT 1:

Submission due date: 12/11/2022

- *Assignment scores/markings depend on neatness and clarity.*
- *Write your answers with enough detail about your approach and concepts used, so that the grader will be able to understand it easily. You should ALWAYS prove the correctness of your algorithms either directly or by referring to a proof in the book.*
- *The marking would be out of 100.*
- *You are allowed to use only those concepts which are covered in the lecture class till date.*
- *Plagiarized assignments will be given a zero mark.*

---

*CO1: to apply **knowledge of computing and mathematics** to algorithm design;*

(i) to argue/prove **correctness of algorithms (both recursive and iterative)** using inductive proofs and invariants;

*(ii) to analyze worst-case **running times of algorithms (both recursive and iterative)** using asymptotic analysis;*

Method of Induction:

Counter Examples:

Proofs of Correctness:

Asymptotic Notations:

Time and Space Complexity:

Solving Recurrences:

Sl.No.	***Question***	***PO***	***Level***
	Method of Induction: Prove the correctness of the statements given in question number 1 to 10 using mathematical induction.		
1.	Prove the following statement using method of Induction. For $\forall n \geq 1$ and n, k $\in R^+$, $\left\lceil \frac{n}{k} \right\rceil = \left\lfloor \frac{n+k-1}{k} \right\rfloor$	PO1	L2, L3, L4
2.	Prove the following statement using method of Induction. For every non-negative integer n, $\sum_{i=0}^{n} 3^i = \frac{3^{n+1}-1}{2}$	PO1	L2, L3, L4
3.	Prove the following statement using method of Induction. Every non-negative integer can be written as the sum of distinct powers of 2. For example: $23 = 2^4 + 2^2 + 2^1 + 2^0$.	PO1	L2, L3, L4

4.	A Binary tree s full if every node has either two children (internal node) or no children (leaf node). Prove that in any full binary tree, the number of leaves is exactly one more than the number of internal nodes.	PO1	L2, L3, L4
5.	Prove by mathematical induction that a decimal number is divisible by 3 if and only if the sum of its digits is divisible by 3.	PO1	L2, L3, L4
6.	For two nonnegative integers a and b, their Greatest Common Divisor (GCD) is the largest integer c such that c divides both a, b (both a/c, b/c are integers). Euclid's algorithm described below can compute the GCD efficiently. *GCD*(a, b) If b == 0 then return a Else return *GCD*(b, a % b) End If Prove that Euclid's algorithm computes GCD(a, b) in time $O(\log(a+b))$ assuming computing $a\%b$ takes 1 unit of time.	PO1	L2, L3, L4
7.	The Fibonacci numbers 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, . . . are recursively defined as follows: $F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2 \end{cases}$ Prove the following statements using mathematical induction. i. For all non-negative integers n, F_n is even if and only if n is divisible by 3. ii. $\sum_{i=0}^{n} F_i = F_{n+2} - 1$ iii. ${F_n}^2 - F_{n+1}F_{n-1} = (-1)^{n+1}$ iv. If n is an integer multiple of m, then F_n is an integer multiple of F_m.	PO1	L1, L2, L3, L4
8.	Prove by mathematical induction that the sum of the cubes of three successive natural numbers is divisible by 9.	PO1	L2, L3, L4
9.	For every non-negative integer n, $\sum_{i=0}^{n}(2i+1)^2 = \frac{(n+1)(2n+1)(2n+3)}{3}$	PO1	L2,

			L3, L4
10.	Prove that, $\|A \times B\| = \|A\| \times \|B\|$ for all finite sets *A* and *B*.	PO1	L2, L3, L4
	Counter Examples: Prove or disprove the statements given in question no. 11 to 20 using counter example.		
11.	For every $n \in \mathbb{N}$, $n^2 - n + 11$ is prime.	PO1	L2, L3
12.	For every $x, y \in \mathbb{N}$, $2^{2x} + 3^{2y+1}$ is prime.	PO1	L2, L3
13.	For every $m \in$ I, there exists an $n \in \mathbb{N}$, such that $\left\|\frac{1}{m} - \frac{1}{n}\right\| > \frac{1}{2}$.	PO1	L2, L3
14.	If *n* is an integer and n^2 is divisible by 4, then *n* is divisible by 4.	PO1	L2, L3
15.	$(a+b)^2 = a^2 + b^2$ is not an algebraic identity $\forall a, b \in R$.	PO1	L2, L3
16.	For $\forall a \in \mathbb{N}$, $\frac{1}{x+2} = \frac{1}{x} + \frac{1}{2}$ is not an identity.	PO1	L2, L3
17.	If $pq = x$, then $p = \frac{x}{q}$ $\forall p, q, x \in R$.	PO1	L2, L3
18.	Show using counter example that, "a + b can be less than min(a, b) $\forall a, b \in R$".	PO1	L2, L3
19.	For all real number *x*, if x^2 is rational then *x* is rational.	PO1	L2, L3
20.	For all real number *x*, y, $\lceil x + y \rceil = \lceil x \rceil + \lceil y \rceil$.	PO1	L2, L3
	Proofs of Correctness:		
21.	Check the correctness of following algorithm that performs conversion from decimal to binary.	PO1, PO2	L2, L3, L4

	`` Algorithm Decimal_to_Binary Input: n, a positive integer Output: b, an array of bits, the bin repr. of n, starting with the least significant bits t:=n; k:=0; while (t>0) do b[k]:=t mod 2; t:=t div 2; k:=k+1; end ```		
22.	Check the correctness of the following algorithm. *percolateMax*(*A*, *n*) /* *A* is an array of *n* integers */ *i* = 0 while *i* < *n* - 1 do if *A*[*i*] > *A*[*i*+1] swap(*A*[*i*], *A*[*i*+1]) *i* = *i* + 1 return *A*[*n*-1]	PO1, PO2	L2, L3, L4
23.	Check the correctness of following algorithm (code) that performs in-place reversal of an array of size *n*. ``` //inputs: array A of size n void reverse_array(int *A, int n): int i = (n - 1) / 2; int j = n / 2; int tmp; while (i >= 0 && j <= (n - 1)) tmp = A[i]; A[i] = A[j]; A[j] = tmp; i--; j++; ```	PO1, PO2	L2, L3, L4
24.	Check the correctness of following algorithm (code) that performs iterative binary search on an array of size *n* to find the element *target*.	PO1, PO2	L2, L3, L4

	`` def binary_search(A, target) lo = 0 hi = len(A) - 1 while lo <= hi: mid = (lo + hi) / 2 if A[mid] == target: return mid elif A[mid] < target: lo = mid + 1 else: hi = mid - 1 ```		
25.	``` float power(float x, int a) // input condition: a >= 0 { if(a == 0) return 1.0; else if(a%2) // a is odd return x*power(x,a-1); else return power(x*x,a/2); } ``` i. Prove the correctness of the above function to compute x^a using mathematical induction. ii. Write the iterative version of the program and prove the correctness using loop invariant.	PO1, PO2	L2, L3, L4
	Asymptotic Notations:		
26.	**For each of the following pairs of functions, determine whether $f(n) = O(g(n))$ or $g(n) = O(f(n))$.** a. $f(n) = n(n-1)/2$ and $g(n) = 6n$ b. $f(n) = n + 2\sqrt{n}$ and $g(n) = n^2$ c. $f(n) = n + \log n$ and $g(n) = n\sqrt{n}$ d. $f(n) = n \log n$ and $g(n) = n\sqrt{n}/2$ e. $f(n) = 2(\log n)^2$ and $g(n) = \log n + 1$	PO1	L1, L2, L3
27.	**State TRUE or FALSE justifying your answer with proper reason.** a. $2n^2 + 1 = O(n^2)$ b. $n^2(1 + \sqrt{n}) = O(n^2)$	PO1	L1, L2, L3

	c. $n^2(1+\sqrt{n}) = O(n^2 \log n)$ d. $3n^2 + \sqrt{n} = O(n + n\sqrt{n} + \sqrt{n})$ e. $\sqrt{n} \log n = O(n)$ f. $lg\ n \in O(n)$ g. $n \in O(n\ lg\ n)$ h. $n\ lg\ n \in O(n^2)$ i. $2^n \in \Omega(6^{\ln n})$ j. $lg^3 n\ \in\ o(n^{0.5})$		
28.	**For each of the following pair of functions $f(n)$ and $g(n)$, determine whether $f(n) = O(g(n))$ or $f(n) = \Theta(g(n))$ or $f(n) = \Omega(g(n))$.**	PO1	L1, L2, L3
	a. $f(n) = \sqrt{n}, g(n) = \log(n+3)$ b. $f(n) = n\sqrt{n}, g(n) = n^2 - n$ c. $f(n) = 2^n - n^2, g(n) = n^4 + n^2$ d. $f(n) = n^2 + 3n + 4, g(n) = 6n^2$ e. $f(n) = n + n\sqrt{n}, g(n) = 4n \log(n^2+1)$		L1, L2, L3
29.	**Prove the following statements:**	PO1	L1, L2, L3
	a. If $f_1(n) = \Omega(g_1(n))$ and $f_2(n) = \Omega(g_2(n))$ then $f_1(n) + f_2(n) = \Omega(g_1(n) + g_2(n))$		L1, L2, L3
	b. If $f_1(n) = \Omega(g_1(n))$ and $f_2(n) = \Omega(g_2(n))$ then $f_1(n) + f_2(n) = \Omega(\min(g_1(n) + g_2(n)))$		L1, L2, L3
	c. If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$		L1, L2, L3
	d. If $f(n) = O(g(n))$, then $f(n)^k = O(g(n)^k)$.		L1, L2, L3
	e. If $f_1(n) = \theta(g_1(n)), f_2(n) = \theta(g_2(n)), \dots, f_k(n) = \theta(g_k(n))$ where k $\in I^+$ then prove that $\sum_{i=1}^{k} f_i(n) = \theta\left(\max_{1 \le j \le k}(g_j(n))\right)$.		L1, L2, L3
30.	Given $f(n) = (n+a)^b$ for any real constants *a* and *b*, where *b* > 0. Prove that $f(n) \in \theta(n^b)$.	PO1	L1, L2, L3
31.	Prove that, $a^n \in o(b^n)$ for *b* > *a* > 0.	PO1	L1, L2, L3

32.	Prove that $lg\ n = O(\sqrt{n})$, however $\sqrt{n} \neq O(lg\ n)$.	PO1	L1, L2, L3
	Time and Space Complexity:		
33.	Let *A* [1 .. 60]= 10, 11, ... , 70. How many comparisons are performed by algorithm Binary_search when searching for the elements present in the first, middle and last position of the array? Formulate and explain by taking a suitable example.	PO1, PO2	L1, L2, L3
34.	Suppose you are choosing between the following three algorithms: • Algorithm *A* solves problem by dividing them into five sub-problems of half the size, recursively solving each sub-problem and then combining the solution in linear times. • Algorithm *B* solves problem size *n* by recursively solving two sub-problems of size *n* by recursively solving two sub-problems of size *n*-1 and then combining the solutions in constant time. • Algorithm *C* solves problems of size *n* by dividing them into nine sub-problems of size *n*/3, recursively solving each sub-problem and then combining the solutions in O(n^2) times. What are the running times of each of these algorithm (in big-O notation), and which would you choose?	PO1, PO2	L1, L2, L3
35.	**function(int *n*)** { if(*n*==1) return 1; else function(*n*/3); function(*n*/3); function(*n*/3); for(*i* = 1; *i* <= *n*; *i*++) *x* = *x* + 1; } Find the time and space complexity of the given algorithm.	PO1	L1, L2, L3
36.	**Void function(int *n*){** *Temp* = 1; Repeat For 1=1 to n *temp* = *temp* + 1 *n*=*n*/2; Until *n*<=1	PO1	L1, L2, L3

	} Find the time and space complexity of the given algorithm.		
37.	**int function(int *n*){** if(*n* <= 2) return 1; else return(function(floor(sqrt(*n*)))+1); } Find the time and space complexity of the given algorithm.	PO1	L1, L2, L3
38.	**Void function(int n)** { if(*n* == 1) return 1; else for(*i* = 1; *i* <= 8; *i*++) function(*n*/2); for(*i* = 1; i<= n^3; *i* ++) count=count+1; } Find the time and space complexity of the given algorithm.	PO1	L1, L2, L3
39.	The following pseudocode performs linear search on an array of size *n* to find the presence of an element *el*. LinearSearch(A, n, el) 1. for *i* = 1 to *n* do 2. If *A*[*i*] = *el* then 3. return *i* 4. return *NIL* Write the recursive version of the LinearSearch algorithm and compare the time and space complexities with the iterative version.	PO1	L1, L2, L3
	Solving Recurrences:		
40.	**Solve the following recurrence using any of the suitable methods. If no solution is possible, justify using proper reasoning.** a. $T(n) = \begin{cases} 1 & if\ n = 2 \\ 1 & if\ n = 4 \\ T\left(\frac{n}{2}\right) + 2T\left(\frac{n}{4}\right) + \theta(n^2) & if\ n > 4 \end{cases}$ Where n is	PO1	L1, L2, L3

assumed to be a power of 2

b. $T(n) = n^{\frac{1}{3}} T\left(n^{\frac{2}{3}}\right) + \theta(n)$

c. $T(n) = \sqrt{n} T(\sqrt{n}) + \log n$

d. $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + \theta(n)$

e. $T(n) = 3T(n/4) + cn^2$

f. $T(n) = \begin{cases} 8 & for\ n = 1 \\ 3T(n-1) - 15 & for\ n > 1 \end{cases}$

g. $T(n) = \begin{cases} 5 & for\ n = 1 \\ 2T(n-1) + 3n + 1 & for\ n > 1 \end{cases}$

h. $T(n) = \frac{n}{n-5} T(n-1) + 1$

i. $T(n) = T(\log n) + \log n$

j. $T(n) = T\left(n^{\frac{1}{4}}\right) + 1$

k. $T(n) = n + 7\sqrt{n} \cdot T(\sqrt{n})$

l. $T(n) = T\left(\frac{3n}{4}\right) + \frac{1}{\sqrt{n}}$

m. $T(n) = \begin{cases} 1 & for\ n = 1 \\ 3T\left(\frac{n}{3} + 5\right) + \frac{n}{2} & for\ n > 1 \end{cases}$

n. $T(n) = \begin{cases} 1 & for\ n = 1 \\ 2T\left(\frac{n}{2}\right) + \frac{n}{\log\log n} & for\ n > 1 \end{cases}$

o. $(n) = \begin{cases} 1 & for\ n = 0 \\ T(n-2) + 2\log n & for\ n > 0 \end{cases}$

p. $T(n) = 2T\left(\frac{n}{2}\right) + n^2(1 + \sin n)$

Submission and Grading:

Submit the hard copy of your assignment by the due date, i.e. 12.11.2022.

Use A4 sized papers only for writing the assignment answers. Keep a front page containing your credentals, assignment number and date.

Part of your assignment grade comes from its "external correctness." This is based on correct output on various sample inputs.

The rest of your assignment's score comes from "internal correctness." Internal correctness includes:

1. Use of methods to minimize the number of steps.
2. Appropriate use of rules, axioms, and suitable diagrams to enhance readability of your responses.