SRN

PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

UE17CS311(CB)

END SEMESTER ASSESSMENT (ESA) B.Tech. V SEMESTER - Dec. 2019

Advanced Algorithms - UE17CS311(CB)

Time: 3 Hrs | Answer All Questions | Max Marks: 100

1a	<pre>Algorithm f(n)
	k ← 0
	i ← 1
	while (i < n)
	      j ← 1
	      while (j ≤ i)
	            k ← k + 1
	            j ← j + 1
	      i ← 2 * i
	return k</pre> Derive the return value of `f(n)` of the above algorithm. Use asymptotic notations; tighter O-notation or Θ-notation.
		06

1b	<pre>Algorithm f(n)
	if (n ≤ 1) return 1
	k ← f(n-1)
	k ← k + f(n-1)
	return k + 1</pre> Derive the number of times the function `f` would be invoked for `f(n)`. Use tighter O-notation or Θ-notation.
		06

1c	Consider a stack with a capacity to hold **'z'** number of elements at the maximum and the following four operations.
	
	`PUSH(S, x)`
	// Pushes element **x** onto the stack S if the stack is not full, in constant time
	
	`POP(S)`
	// Pops top element from the stack S if the stack is not empty, in constant time
	
	`MULTIPOP(S, k)`
	// Pops at most **k** elements from the stack
	`while not STACK-EMPTY(S) and k > 0 do`
	`POP(S), k ← k - 1`
	
	`Backup(S)`
	// Backs up all the elements in the stack, which is not more than **z** number of
	// elements in linear time. Backup(**S**) operation is not invoked by the user,
	// but is automatically invoked after every **z** number of other operations.
	
	Derive a constant amortized costs for each operation using the accounting method of amortized analysis. Explanation of each amortized cost is needed and avoid using other methods to find the amortized costs.
		08

SRN

2a	Derive a dynamic programming algorithm for the following problem. There is a row of **n** coins of values **c[1..n]**. The values are positive integers and not necessarily distinct. The goal is to pick up the maximum amount of money subject to the constraint that **no two coins adjacent** in the initial row can be picked up. Make sure the algorithm takes O(**n**) time and O(**1**) extra space.	06
2b	Derive an O(**n**) time complexity and O(1) space complexity algorithm to find the number of ways of tiling a hallway of dimension ***n x 3*** square feet (where **n ≥ 0**) using only the tiles of dimension ***1 x 3*** square feet. The tiles can be placed either horizontally or vertically without breaking.	06
2c	Several coins are placed in cells of an **m × n** board, no more than one coin per cell. A robot, located in the upper-left cell of the board, needs to collect as many coins as possible and bring them to the bottom-right cell. On each step, the robot can move either one cell to the right or one cell down from its current location. When the robot visits a cell with a coin, it always picks up that coin. Design an O(**mn**) time complexity algorithm to find the maximum number of coins the robot can collect.	08
3a	A **suffix trie** would take O(n^2) space. Explain the techniques used to reduce the space consumption to O(**n**) in a **suffix tree**.	06
3b	Find the prefix function π used in the Knuth-Morris-Pratt (KMP) algorithm for the pattern "**ababbabbabbababbabb**".	06

3c	Design a finite automaton that reads a text of length **n** character-by-character, going to an accepting state, only if the pattern of length **m** has just been seen and returns the offset of the pattern in the text. Write the following two algorithms.
	
	`ComputeTransitionFunction(Pattern, Alphabet)`
	returns the TransitionTable[state, symbol].
	
	`FiniteAutomatonMatcher(Text, TransitionTable, patternLength)`
	prints the offsets of all the occurrences of the pattern in the text.
		08

4a	Write an algorithm to find modular exponentiation a^b **mod n**. Find 7^{560} **mod 561** using the algorithm.	06
4b	Prove that if **a** and **b** are any integers, not both zero, then **GCD(a, b)** is the smallest positive element of the set {**ax + by**: **x**, **y** ∈ Z} of linear combinations of **a** and **b**.	06
4c	Write the Extended Euclid's algorithm to find the GCD of **a** and **b**, which also finds **x** and **y** where **ax + by** = GCD(**a**, **b**). Using the algorithm, find the **x** and **y** where the GCD of 462 and 840 is represented as a linear combination 462**x** + 840**y**.	08
5a	Explain the addition of two polynomials of degree-bound **n** in point-value representation. Mention the time complexity of the method.	06
5b	Explain a simple method of conversion from the coefficient representation to the point-value representation with an example polynomial $A(x) = a_0 + a_1x + a_2x^2 + a_3x^3$.	06
5c	Explain an O(n logn) method of multiplying two polynomials of degree-bound **n**.	08