K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

2nd SESSIONAL TEST QUESTION PAPER 2018–19 Even SEMESTER

SET – A/~~B~~

USN										

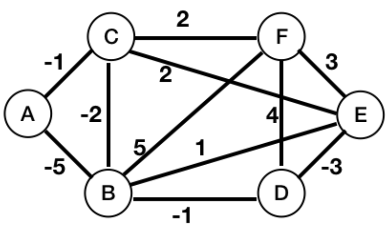
Degree : **B.E** | **Semester** : **IV**
Branch : **Computer Science & Engineering** | **Subject Code** : **17CS43**
Subject Title : **Design and Analysis of Algorithms** | **Date** : **2019-04-16**
Duration : **90 Minutes** | **Max Marks** : **30**

Note: Answer ONE full question from each part.

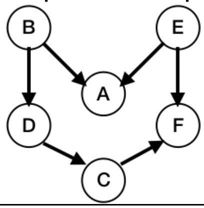
Q No.	Question	Marks
	PART-A	
1(a)	Use Divide and conquer approach and **construct** an algorithm to find k^{th} smallest element in the given sequence of n elements. Hint: Use the idea of quicksort except that only one sub-problem has to be solved. Identify which subsequence contains the k^{th} smallest element and use that subsequence to find the solution.	**5**
Sch & Ans	Sch:3 marks for identifying sub-problem and 2 marks for algo Ans: Take the first element as pivot and using quicksort identify its correct position in sorted order. Let this position be p. Now we have 3 choices. $p<k$: In this k^{th} element is in 2nd partition, and its we need to search $(k-p)^{th}$ smallest element in 2nd partition. $p=k$: This implies that we have found k^{th} smallest element $p>k$: This implies that k^{th} smallest element is in first partition and we need to find k^{th} smallest element if first partition. Thus, after the partition of the input using pivot, we are working with only 1 sub-problem. This would give a solution in $O(n)$ time.	
(b)	**Make use of** fractional knapsack algorithm to **build** a solution for the instance $n=7$, knapsack size $m=15$, profits $p_1=10$, $p_2=5$ $p_3=15$, $p_4=7$, $p_5=6$, $p_6=18$, $p_7=3$, and weights as $w_1=2$, $w_2=3$, $w_3=5$, $w_4=7$, $w_5=1$, $w_6=4$, $w_7=1$.	**5**
Sch & Ans	Sch:1 mark for sorting weights as per profit/weights, 4 marks for generating the solution Ans: Ordering weights in non-increasing order of profit are $p_5/w_5=6$, $p_1/w_1=5$, $p_6/w_6=4.5$, $p_3/w_3=3$, $p_7/w_7=3$, $p_2/w_2=1.5$, $p_4/w_4=1$ Thus picking up weights till it becomes 15 would be as follows $W_5=1, w_1=2, w_6=4, w_3=5, w_7=1, w_2=2$ (Take partial weight of w_2) Total profit: $6+10+18+15+3+2*5/3=3.33=55.33$	
(c)	Consider that a country has coins in the denominations of Rs 14, 12, 5, and 1. **Show** an example value to prove that greedy algorithm always does not generate change with minimum number of coins.	**5**
Sch & Ans	Sch:2 marks for identifying proper example value and 1 mark for generating change as per algo, 2 marks for optimal solution. Ans: Consider the change value to be given as 24.	

	As per greedy algorithm, the change given would require 3 coins Rs 14: 1 coin Rs 5 : 2 coins The optimal solution for coin change would require 2 coins Rs 12: 2 coins	
	OR	
2(a)	**Analyze** time complexity for QuickSort in worst case. Hint: Define and explain the recurrence relation for worst case and solve the same.	**5**
Sch & Ans	Sch: 2 marks for identifying pivot position for worst case, 1 mark for writing recurrence relation and 2 marks for solving recurrence equation For worst case performance, portioning of input array of size k should split as skewed as possible i.e. one partition should have only 0 element, and other partition should have k-1 elements. Thus, the recurrence relation becomes T(n) = T(n-1)+O(n) (after partition we need to merge n elements) Solving the recurrence relation gives T(n) = O(n^2)	
(b)	Consider *Dijkstra's* algorithm to find the single source shortest path for a weighted graph. **Identify** the changes that you need to make in this algorithm to find a shortest path between given two vertices i.e. **develop** the algorithm for single-pair-shortest-paths problem.	**5**

Sch & Ans	Sch: 2 marks for identifying changes and 3 marks for the algorithm.
	Ans: The major change is in the for loop where we iterate it |V|-1 times, Iterate this till destination node is discovered,
	The key part of algorithm would be as below
	for i=0 to |V|-1 do
	    u = DeleteMin(Q) //time implememtation based
	    V_T= V_T={u}
	    If u == destination vertex
	        Break; //single source single dest path is found.
	    for every vertex wЄV-V_T adjacent to w, do
	        if d_u+weight(u,w)< d_w, then
	            d_w←d_u+weight(u,w)
	            p_w←u
	            Decrease(Q,w,d_w)//time implementation based
	        fi
	    end //for wЄV-V_T
	end //for i=0	

(c)	**Apply** the algorithm developed in Q2(b) to find the shortest path from node F to node B. **Compare** the path thus found using this algorithm with actual shortest path and **explain** the difference. 	**5**
Sch	Sch: 2 marks for computing path from F to B, 2 marks for identifying optimal path, and 1	

& Ans	marks difference explanation Ans: The path from F to B using Dijkstra's algorithm F→D→B with a total path length 3. The optimal path from F to B is F→C→A→B with a total path length of -4. The difference arises because Dijkstra's algorithm does not work with negative weights for edges.	
	PART-B	
3(a)	**Illustrate** the product P(x)*Q(x) computation manually using divide and conquer multiplication approach for following polynomials. $P(x) = 0+1x+2x^2+3x^3+\ldots+7x^7$ $Q(x) = 8+7x+6x^2+5x^3+\ldots+1x^7$	**5**
Sch & Ans	Sch: 2 marks for defining the approach and 3 marks for solving Ans: Write P(x) and Q(x) as $P(x)=P_1(x)+x^4P_2(x)$ $Q(x)=Q_1(x)+x^4Q_2(x)$ $C_1=P_1Q_1$ $C_2=P_2Q_2$ $C_3=[(P_1+P_2)*(Q_1+Q_2) - C_1 - C_2]$ Compute P(x)*Q(x) as C_1+x7 C_2+x^4 C_3, and proceed in this manner	
(b)	**Construct** a Minimum Cost Spanning Tree using Prim's algorithm for the graph shown in Q2(c) starting from node C as the root of the spanning tree. Please note that some edges have negative weights associated with them	**5**
Sch & Ans	Sch: 1 mark each for adding each node other than node C Ans: The nodes and edges will be added in below order C: B: {C,B} A: {B,A} D: {B,D} E: {D,E} F: {C,F}	
(c)	**Analyze** the time complexity of *Prim's* algorithm using both *Adjacency Martrix* and *Adjacency List* representation.	**5**
Sch & Ans	Sch: 3 marks for complexity analysis using Adjacency Matrix and 2 marks for using Adjacency List Ans: For prim's algorithm, for loop runs $\|V\|$ times, and in each iteration we find next nodes having minimum cost edge. Each edge would contribute to change of node weight only 1 time. Thus, total number of times, weight of node is reduced is $O\|E\|$. When using Adjacency matrix, maintain the nodes in an unsorted array and finding a minimum weight node will take $O(\|V\|)$ time. Since there are $\|V\|-1$ iterations, thus total time complexity using Adjacency matrix is $O(\|V\|^2)$ When using Adjacency list, we maintain the node weights in a priority queue. Thus find min takes $O(1)$ time, but once find min node is removed and weights of other nodes are adjusted, each weight adjustment may take $O(\lg \|V\|)$ time. Since this weight adjustment can be done for each edge, the total time complexity is $O(\|E\|*\lg \|V\|)$.	

	OR	
4(a)	For the graph below, Identify and **List** all possible topological sorting orders.  	**5**
Sch & Ans	Sch: 1 mark each for first 5 topological ordering list. Ans Order 1: B E A D C F Order 2: B E D A C F Order 3: B E D C A F Order 4: B E D C F A Order 5: E B A D C F Order 6: E B D A C F Order 7: E B D C A F Order 8: E B D C F A Order 9: B D C E A F Order 10: B D E C A F Order 11: B D E A C F	
(b)	**Analyze** the time complexity of Kruskal's algorithm using *Union-Find* with fixed cost of Union operation as `O(1)` and varying cost of Find operation.	**5**
Sch & Ans	Sch: 2 marks for cost analysis using Union() operation, 2 marks for cost analysis using Find() operation and 1 marks total cost analysis Ans: Cycle checking is done for each edge i.e. top loop runs `|E|` times. Union operation is done |V|-1 times and thus total cost of union operation is `O(|V|)` Find operation requires traversing the tree and depth of tree may be `O(lg |V|)`. Thus, total find cost could be `O(|V|*lg |V|)` Thus, total time complexity is `O(|E|+|V|*lg |V|)` in addition to sorting all edge by their weights.	
(c)	Consider the graph as shown in Q2(c), and **develop** Union Find trees to check for cycle formation when considering each edge for Minimum Cost Spanning Tree.	**5**
Sch & Ans	Sch: 1 mark for taking each node and making it a union with other set except the first node. Ans: Following edges will be added to spanning tree (A, B)=-5, B→A(2), C→C(1), D→D(1), E→E(1), F→F(1) (D, E)=-3, B→A(2), C→C(1), E→D(2), F→F(1) (B, C)=-1, {B,C}→A(3), E→D(2), F→F(1) (B, D)=-1, {{{E→D}→B},C}→A(5), F→F(1) (C, F)=2, {{{E→D}→B},{F→C}}→A(6)	

Signature of the Faculty **Signature of the HOD**