National University of Computer & Emerging Sciences, Karachi
Fall-2017 Department of Computer Science
Final Exam
18th December 2017, 9:00 am– 12:00 noon

Course Code: CS302	**Course Name:** Design and Analysis of Algorithm
Instructor Name / Names: Dr. Muhammad Atif Tahir, Subhash Sagar and Zeshan Khan	
Student Roll No:	**Section:**

Instructions:

- Return the question paper.
- Read each question completely before answering it. There are **9 questions.**
- In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper.

Time: 180 minutes. **Max Marks**: 50 points

Question 1: **(6 points)**

a) Given $[A_{nxn}][X_{nx1}]=[C_{nx1}]$ and A = LU, show that LU Decomposition leads to **[3 points]**

i. $[L_{nxn}]\ [Z_{nx1}] = [C_{nx1}]$
ii. $[U_{nxn}]\ [X_{nx1}] = [Z_{nx1}]$

where **L="Lower Triangular Matrix"** and **U="Upper Triangular Matrix".**

b) Whether the given matrix has LU decomposition or not? **[3 points]**

$$A = \begin{bmatrix} 5 & 2 & 3 \\ 5 & 2 & -3 \\ 10 & 2 & 4 \end{bmatrix}$$

Question 2: **(5 points)**

Given Two Strings S_i and S_j. Design and analyze an algorithm to find whether one string S_i is the rotation of another string S_j. Strings S_i and S_j are rotations of each other if $S_i = uv$ and $S_j = vu$ for some strings u and v. For example, ALGORITHM is a rotation of RITHMALGO. The time complexity of the designed algorithm should be ≤ O (n).

Question 3: **(6 points)**

Floyd-Warshall can be used to determine whether or not a graph has *transitive closure*, i.e. whether or not there are paths between all vertices using the following procedure:

- Assign all edges in the graph to have weight = 1
- Run Floyd-Warshall
- Check if all $d_{ij} < n$ i.e. all values in matrix D is less than all values in matrix n.

Use the above algorithm to determine whether the graph in ***Figure 1*** has *transitive closure* or not.

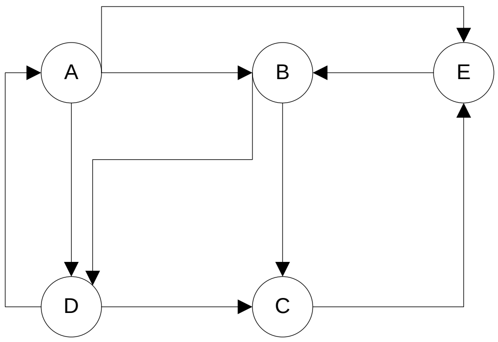

Figure 1

Question 4: **(5 points)**

Complete the following table.

Algorithm	**Worst Case**	**Write below whether the algorithm belongs to Dynamic Programming / Greedy Algorithms / None of them**
0/1 Knapsack using Dynamic Programming		
0/1 Knapsack using Brute Force		
Breadth First Search		
Depth First Search		
Kruskal's algorithm for finding MST		
Prim's algorithm for finding MST		
Shortest path by Dijkstra		
Shortest path by Bellman Ford		
Matrix Chain Multiplication using Dynamic Programming		
Matrix Chain Multiplication using Brute Force		

Question 5: **(6 points)**

Answer the following questions

(a) Explain the difference between greedy algorithms and dynamic programming **[2 Points]**
(b) Explain what P, NP and NP-Complete problems are? What is meant by "is P = NP"? **[4 Points]**

Question 6: **(5 points)**

Consider each of the following words as a set of letters: {arid, dash, drain, heard, lost, nose, shun, slate, snare, thread}. Show which set cover GREEDY-SET-COVER produces, when we break ties in favor of the word that appears first in the dictionary

GREEDY-SET-COVER$(X, \mathcal{F})$

$U = X$
$\mathcal{C} = \emptyset$
while $U \neq \emptyset$
  select an $S \in \mathcal{F}$ that maximizes $|S \cap U|$
  $U = U - S$
  $\mathcal{C} = \mathcal{C} \cup \{S\}$
return $\mathcal{C}$

Question 7: **(6 points)**

For each of the following questions, circle either **T** (True) or **F** (False) and justify using some examples e.g. assuming a function?

T **F** For all positive f(n), $f(n) + o(f(n)) = \Theta(f(n))$.

T **F** For all positive f(n), g(n) and h(n), if $f(n) = O(g(n))$ and $f(n) = \Omega(h(n))$, then $g(n) + h(n) = \Omega(f(n))$.

T **F** If $f(n)= O(g(n))$ and $f(n)= \Omega(g(n))$, then we have $(f(n))^2 = \Theta((g(n))^2)$

T **F** If $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, then we have $f(n) = g(n)$

Question 8: **(5 points)**

Proved that the weight of the Minimum Spanning Tree (MST) is NOT always less than Optimal, the weight of the Travel Salesman Problem (TSP) solution T if all edges in the graph have positive weights except one edge with negative weight.

Question 9: **(6 points)**

Suppose that we have a given set of five matrices A1, A2, A3, A4 and A5 with the following dimensions:

Matrix	A1	A2	A3	A4	A5
Dimension	5 X 4	4 X 3	3 X 5	5 X 4	4 X 3

Use the Matrix Chain Multiplication algorithm to discover the optimal parenthesization of the matrices.