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".**

Solution:

If solving a set of linear equations: $[A][X] = [C]$

If $[A] = [L][U]$ then $[L][U][X] = [C]$ [0.5]

Multiply both sides by $[L]^{-1}$

Which gives: $[L]^{-1}[L][U][X] = [L]^{-1}[C]$ [0.5]

Remember $[L]^{-1}[L] = [I]$

Which leads to $[I][U][X] = [L]^{-1}[C]$ [0.5]

Now, if $[I][U] = [U]$ then $[U][X] = [L]^{-1}[C]$ [0.5]

Now, let $[L]^{-1}[C] = [Z]$ [0.5]

Which ends with and [0.5]

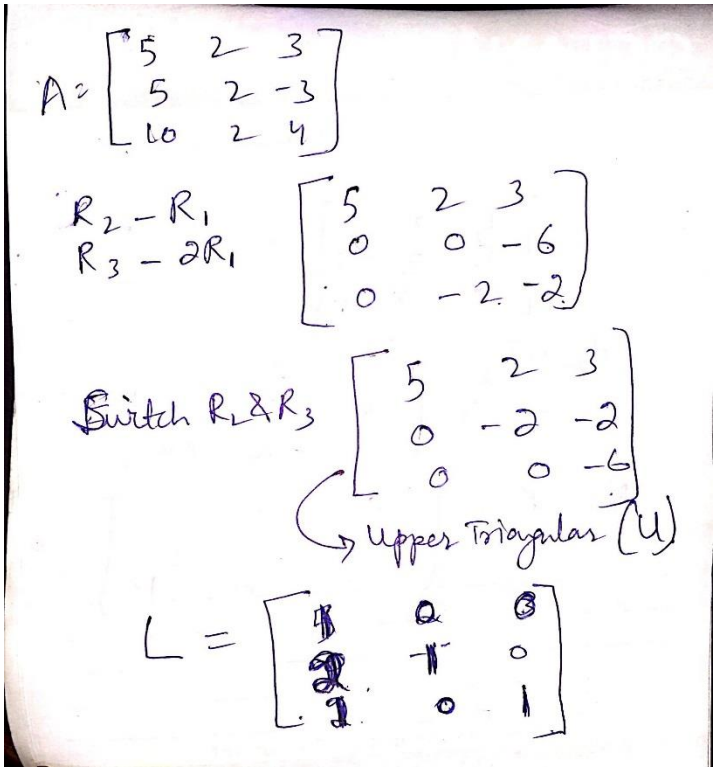
1) $[L][Z] = [C]$
2) $[U][X] = [Z]$

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

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

Yes the decomposition is possible, [0.5 for Yes] [2.5 for clear steps which can vary]

Here due to row switch, lower triangular matrix rows are bit different. And there may be other solutions as well like if we interchange the 1st and 3rd row of original matrix and then solve accordingly.



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). [Approach can vary, some will use functions, some take input directly into main method etc]

(a) There are different ways to program this,

```
(b) public class MyClass {
(c)     public static void main(String args[]) {
(d)         String input = "ALGORITHM";
(e)         String rotated = "RITHMALGO";
(f)
(g)         int index = rotated.indexOf(input.charAt(0));
(h)         int finalpos = rotated.length()-index;
(i)
(j)         System.out.println(rotated.substring(0,index));
(k)         System.out.println(input.substring(finalpos));
(l)         System.out.println(input.substring(0,finalpos));
(m)         System.out.println(rotated.substring(index));
(n)
(o)        // then we can check conditions
(p)     }
(q) }
```

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 [0.5 Point]
- Run Floyd-Warshall [5 Points; 1 for each vertex]
- Check if all $d_{ij} < n$ i.e. all values in matrix D is less than all values in matrix n. [0.5]

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

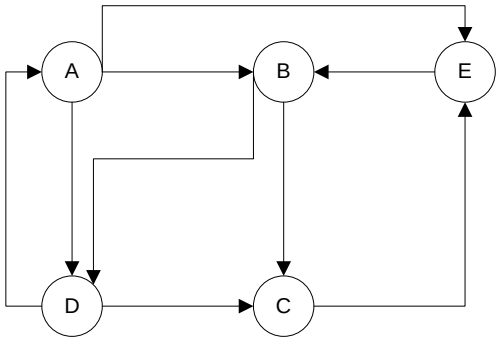
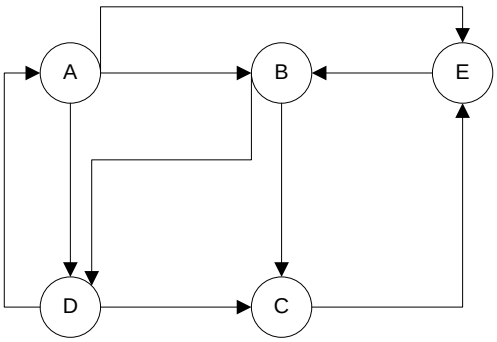
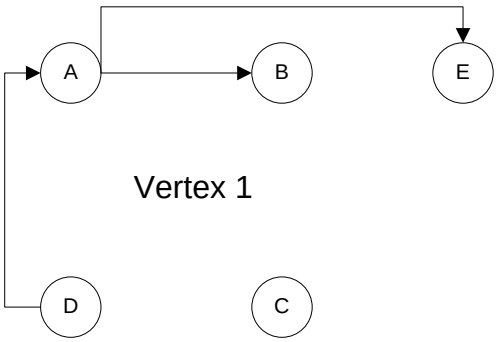
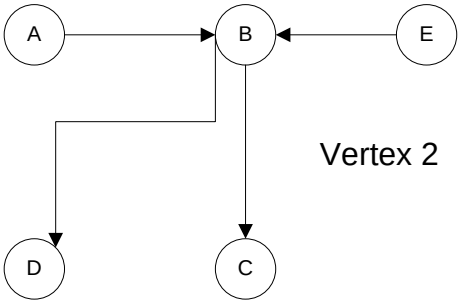
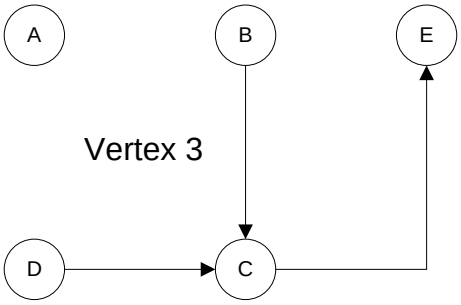


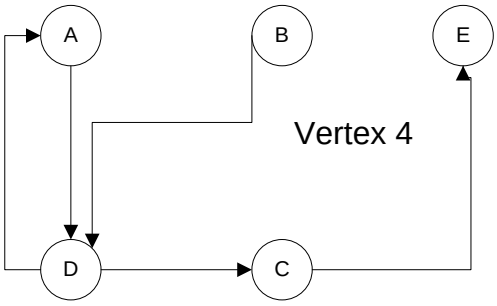
Figure 1

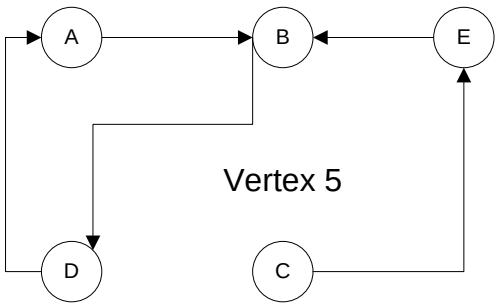












For iteration 0 (Matrix D)

D	1	2	3	4	5
1	0	1	Inf	1	1

Matrix n	1	2	3	4	5
1	-	1	-	1	1

2	inf	0	1	1	Inf
3	inf	inf	0	inf	1
4	1	inf	1	0	Inf
5	inf	1	inf	inf	0

2	-	-	2	2	-
3	-	-	-	-	3
4	4	-	4	-	-
5	-	5	-	-	-

For iteration 1 through vertex 1 (Matrix D)

D	1	2	3	4	5
1	0	1	Inf	1	1
2	Inf	0	1	1	Inf
3	Inf	inf	0	inf	1
4	1	2	1	0	2
5	inf	1	inf	inf	0

Matrix n	1	2	3	4	5
1	-	1	-	1	1
2	-	-	2	2	-
3	-	-	-	-	3
4	4	1	4	-	1
5	-	5	-	-	-

For iteration 1 through vertex 2 (Matrix D)

D	1	2	3	4	5
1	0	1	2	1	1
2	Inf	0	1	1	Inf
3	Inf	inf	0	inf	1
4	1	2	1	0	2
5	inf	1	2	2	0

Matrix n	1	2	3	4	5
1	-	1	2	1	1
2	-	-	2	2	-
3	-	-	-	-	3
4	4	1	4	-	1
5	-	5	2	2	-

For iteration 1 through vertex 3 (Matrix D)

D	1	2	3	4	5
1	0	1	2	1	1
2	Inf	0	1	1	2
3	Inf	inf	0	inf	1
4	1	2	1	0	2
5	inf	1	2	2	0

Matrix n	1	2	3	4	5
1	-	1	2	1	1
2	-	-	2	2	3
3	-	-	-	-	3
4	4	1	4	-	1 or 3
5	-	5	2	2	-

For iteration 1 through vertex 4 (Matrix D)

D	1	2	3	4	5
1	0	1	2	1	1
2	2	0	1	1	2
3	Inf	inf	0	inf	1
4	1	2	1	0	2
5	inf	1	2	2	0

Matrix n	1	2	3	4	5
1	-	1	2	1	1
2	4	-	2	2	3
3	-	-	-	-	3
4	4	1	4	-	1 or 3
5	-	5	2	2	-

For iteration 1 through vertex 5 (Matrix D)

D	1	2	3	4	5
1	0	1	2	1	1
2	2	0	1	1	2
3	4	2	0	3	1
4	1	2	1	0	2
5	3	1	2	2	0

Matrix n	1	2	3	4	5
1	-	1	2	1	1
2	4	-	2	2	3
3	5	5	-	5	3
4	4	1	4	-	1 or 3
5	5	5	2	2	-

Question 4: **(5 points)**

Complete the following table. [0.5 each; worst case 0.25 and 0.25 for belonging]

Algorithm	**Worst Case**	**Write below whether the algorithm belongs to Dynamic Programming / Greedy Algorithms / None of them**
0/1 Knapsack using Dynamic Programming	O(nW)	Dynamic Programming

0/1 Knapsack using Brute Force	$O(2^n)$	None of them
Breadth First Search	O(V + E)	Greedy
Depth First Search	O(V + E)	Greedy
Kruskal's algorithm for finding MST	O(ElogV)	Greedy
Prim's algorithm for finding MST	O(ElogV)	Greedy
Shortest path by Dijkstra	O((V + E)logV)	Greedy
Shortest path by Bellman Ford	O(V * E) or O(V^3)	Dynamic Programming
Matrix Chain Multiplication using Dynamic Programming	$O(n^3)$	Dynamic Programming
Matrix Chain Multiplication using Brute Force	$O(4^n)$	None of them

Question 5: **(6 points)**

Answer the following questions

(a) Explain the difference between greedy algorithms and dynamic programming **[2 Points]**

Greedy Algorithms: A greedy algorithm is a mathematical process that looks for simple and easy-to-implement solutions to complex, multi-step problems by deciding which next step will provide the most obvious benefit.
Dynamic Programming: It is often used to solve optimization problems. Its is a tabular method in which we break down the problem into subproblems and place the solution to the subproblems in a matrix

(b) Explain what P, NP and NP-Complete problems are? What is meant by "is P = NP"? **[4 Points]**

"P" – Problems that can be solved in polynomial time on a deterministic machine.

"NP" – Problems that cannot be solved in polynomial time on a deterministic machine but their solution can be verified in polynomial time on a deterministic machine. Alternatively, problems that can be solved in polynomial time on a non-deterministic machine.

"NP-Complete" – A problem is NP-Complete if it belongs to class NP and all NP problems can be reduced to it in polynomial time.

"is P = NP" – it means whether an NP problem can belong to class P problem. In other words whether every problem whose solution can be verified by a computer in polynomial time can also be solved by a computer in polynomial time.

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, F)
1  U = X
2  C = ∅
3  while U ≠ ∅
4      select an S ∈ F that maximizes |S ∩ U|
5      U = U − S
6      C = C ∪ {S}
7  return C
```

Since all of the words have no repeated letters, the first word selected will be the one that appears earliest on among those with the most letters, this is "thread". Now, we look among the words that are left, seeing how many letters that aren't already covered that they contain. Since "lost" has four letters that have not been mentioned yet,and it is first among those that do, that is the next one we select. The next one we pick is "drain" because it is has two unmentioned letters. This only leave "shun" having any unmentioned letters, so we pick that, completing our set. So, the final set of words in our cover is $\{thread, lost, drain, shun\}$.

1 point for each explanation

1 point for correct order

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? [0.5 for T or F; 1 point for explanation]

T **F** For all positive f(n), f(n) + o(f(n)) = Θ(f(n)).

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

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

T **F** If f(n) = O(g(n)) and f(n) = Ω (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. [3.5 points until OPT – w(e); 1.5 for w<0]

Take T and remove an edge e. T is now a spanning tree.

Because M is the MST, $w(M) \leq w(T-e) = w(T)-w(e) = OPT-w(e)$

Since one of the edge has a negative edge

w(e) < 0, thus w(M) not always less than OPT.

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.

Cost Table: [3 Points]

i/j	1	2	3	4	5
1	**0**	**60**	**135**	**180**	**192**
2		**0**	**60**	**108**	**132**
3			**0**	**60**	**96**
4				**0**	**60**
5					**0**

Optimal Cost = **192**

Parenthesis Table: [2 Points]

i/j	1	2	3	4	5
1	**1**	**2**	**2**	**2**	**1**
2		**2**	**2**	**2**	**2**
3			**3**	**3**	**4**
4				**4**	**4**
5					**5**

Optimal parenthesis = **(A1(A2((A3A4)A5))) [1 Point]**