National University of Computer & Emerging Sciences, Karachi
Fall-2023 Department of Computer Science
Final Exam
18th December 2023, 09:30 AM – 12:00 PM
Part (B)

Course Code: CS2009	**Course Name:** Design and Analysis of Algorithm
Instructor Name / Names: Dr. Muhammad Atif Tahir, Dr. Fahad Sherwani, Dr. Farrukh Saleem	
Student Roll No:	**Section:**

Instructions:

- Return the question paper.
- Read each question completely before answering it. There are 8 questions on 4 pages.
- In case of any ambiguity, you may make assumptions. But your assumption should not contradict any statement in the question paper.

Time: 150 minutes **Max Marks: 40**

Question # 3 [CLO:1] Answer the following questions and explain in only 2-3 lines only. [5 marks]

Maring Scheme: Correct Sol: Full marks; Incorrect: 0; Partial Correct: Partial marks only for parts c, d, and g

a) [0.5 Point] Floyd-Warshall algorithm computes shortest path from every vertex to every other vertex using $O(n^3)$ complexity, whereas Bellman-Ford can compute in $O(n^3)$, only from a single source to other vertices. Why one needs to use Bellman-Ford algorithm, in presence of more information rich Floyd-Warshall algorithm. (apart from detecting a negative cycle)

Sol: Space complexity of Bellman-Ford is O(n), versus Folyd-Warshall of O(n^2)

b) [0.5 Points] True / False: Does Directed Acyclic Graph (DAG) can have negative weight edges?

Sol: True

c) [0.5 Point] Describe the difference between tight and loose bounds in Big-O notation. Provide an example scenario where a tight bound is preferable and explain why.

Sol: Tight bounds in Big-O notation provide a more accurate description of an algorithm's complexity, while loose bounds may overestimate or underestimate. In scenarios where precise analysis is crucial, tight bounds are preferred. For example, in real-time systems, where response times must be accurately predicted, a tight bound is essential for planning.

d) [0.5 Point] What do you mean by the optimal solution?

Sol: Given the problem with inputs, we recover a subset that appeases come constraints. Any subset that satisfies these constraints is known as a feasible solution. A feasible solution, which either maximizes or minimizes a given purpose method is known as an optimal solution.

e) [1 Point] Problem A reduces to problem B if you can use an algorithm that solves B to help solve A. Let's say problem A is to find median and problem B uses any sorting method with O(n2) complexity. What would be the complexity of Problem A

Sol: $O(n^2) + 1$

f) [0.5 Point] True or False: In dynamic programming, a subproblem do share sub-subproblems

Sol: True: Subproblem do share sub-subproblems by storing it in matrix.

g) [1.5 Points] Differentiate b/w Single-source, Single-pair and All-pairs shortest path and one algorithm for each studied during this course

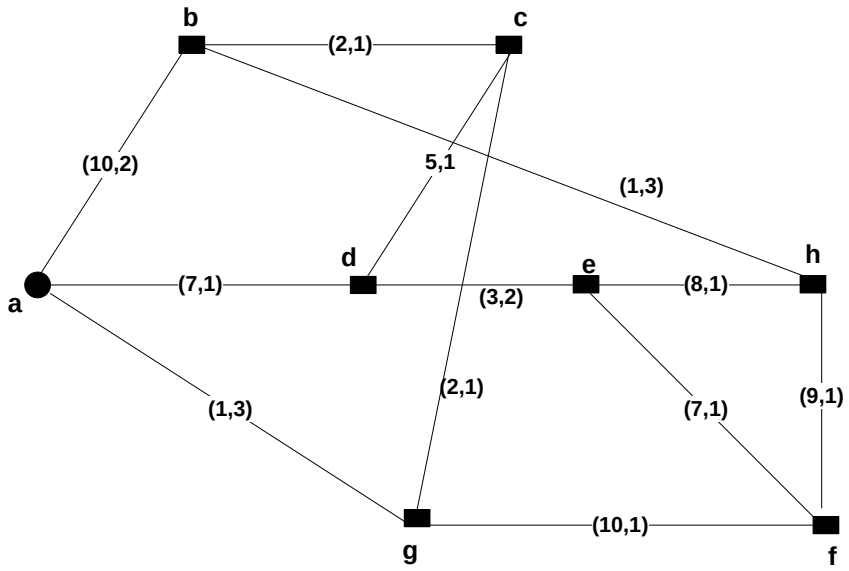
Sol: Sol: Single Source: Find the shortest path from a given source to each of the vertices e.g. Dijkstra.

Single-pair. Given two vertices, find the shortest path between them. Solution to single-source problem solves this problem efficiently, too e.g. Dijkstra

All-pairs. Find shortest-paths for every pair of vertices e.g. Floyed Warshall Algorithm

Question # 4 [CLO: 3] **[5 marks]**

Demand for multimedia, combining audio, video, and data streams over a network, is rapidly increasing. Some of the most popular uses of multimedia are real-time interactive applications such as desktop video and audio conferencing, collaborative engineering, shared white boards, transmission of university lectures to a remote audience, and animated simulations. With the advent of real-time interactive applications, **delay constraint** is also an important objective along with minimizing bandwidth cost. Delay constraint means that a packet must be reached within that interval. Figure below shows an example tree. Each edge consists of 2 values (**first one is bandwidth cost** and **second one is delay**). Your objective is to design a spanning tree that optimizes both bandwidth cost and delay constraint simultaneously. Later show the dry run using Figure below. [Hint: Modify Prims or Kruskal using some weighted function].



Sol:

Ideally, 1st normalize both bandwidth cost and delay constraint to give equal importance to both.

Then apply following.
Modify Cost function = 0.5 * bandwidth + 0.5 * delay constraint (Ideally first normalize both parameters using max-min or other normalize function but for simplicity the above solution is partially acceptable)

b-c = 0.5 * 2 + 0.5 * 1 = 1.5; Similarly for others and then apply either Kruskal or Prims

Marking Scheme: Correct Modification of normalized cost function = 2 Points.
Correct Modification without normalized cost function = 1 Point
Running of Kruskal or Prims = 3 marks showing steps correctly. 3 marks to be given if they just attempt to run prims or Kruskal independently.

Question # 5 [CLO: 3] **[5 marks]**

Longest palindrome subsequence
A palindrome is a nonempty string over some alphabet that reads the same forward and backward. Examples of palindromes are all strings of length 1, civic, racecar etc. Give an efficient algorithm to find the longest palindrome that is a subsequence of a given input string. For example, given the input "character", your algorithm should return "carac".
The idea is to check and compare the first and last characters of the string. There are only two possibilities for the same: If the first and the last characters are the same, it can be ensured that

both the characters can be considered in the final palindrome and hence, add 2 to the result, since we have found a sequence of length 2 and recurse the remaining substring S[i + 1, j – 1]. In case, the first and the last characters aren't the same, the following operation must be performed:
Recurse S[i + 1, j]
Recurse S[i, j – 1]

a) Design an algorithm using a brute force approach (2^n possible subsequences) to find the maximum length obtained amongst them [2 Points]
b) Design an algorithm using dynamic programming approach (polynomial time solution required) to find the maximum length obtained amongst them [3 Points]

For both parts (a) and (b), only pseudocode is required but must be with clear formulas / loops/ if-else statements not just plain text. You can show dry run for your own understanding, but dry run would not have any weightage.

Solution:
Brute Force Algorithm

```
LPS(1..n) {
        if (n == 1)
                return 1
        if (S[1] == S[n])
                return 2 + LPS(2..n-1)
        else
                return max( LPS(1..n-1), LPS(2..n) )
}
```

Marking Scheme: Correct Sol 2 Points; Partial: 0.5-1.5 Points, No Solution: 0

The **Dynamic Programming** Algorithm

```
int LPS[1..n][1..n];

DPLPS(char S[1..n]) {
        for (i = 1; i <= n; i++)
                LPS[i][i] = 1;
        for (len = 1; len < n; len++) {
                for (i = 1; i < n-len; i++) {
                        j = i + len;
                        if (S[j] == S[j])
                                LPS[i][j] = 2 + LPS[i+1[j-1];
                        else
                                LPS[i][j] = max{LPS[i][j-1], LPS[i+1][j]}
                }
        }
        return LPS[1][n];
}
```

Marking Scheme: Correct Sol 3 Points; Partial: 0.5-2.5 Points, No Solution: 0

Question # 6 [CLO: 1] **[5 marks]**

Given an array of integers arr[] of size N and an integer, the task is to rotate the array elements to the left by d positions, (where d<=n).
a) Design an algorithm to solve the above task with O(n) time complexity, and O(n) space complexity [2.5 Points].
b) Design an algorithm to solve the above task with O(n^2) time complexity, and O(1) space complexity [2.5 Points]

Pseudocode in plain text is acceptable for this question but must be clear.

Example:
Input: arr[] = {3, 4, 5, 6, 7, 1, 2}, d=2
Output: 5 6 7 1 2 3 4
Solution:
Algorithm a (Using temp array):

- First store the elements from index d to N-1 into the temp array.
- Then store the first d elements of the original array into the temp array.
- Copy back the elements of the temp array into the original array

Marking Scheme: Correct Sol 2.5 Points; Partial: 0.5-2.0 Points, No Solution: 0

Algorithm b (Rotate one by one):

- At each iteration, shift the elements by one position to the left circularly (i.e., first element becomes the last).
- Perform this operation d times to rotate the elements to the left by d position.

Marking Scheme: Correct Sol 2.5 Points; Partial: 0.5-2.0 Points, No Solution: 0

Question # 7 [CLO:3] **[5 marks]**

Calculate the Levenshtein-Distance for str1 = “PLASMA”, and str2 = “ALTRUISM” using Dynamic Programming Approach of Levenshtein-Distance.
Pseudocode Levenshtein-Distance is given below.

Levenshtein-Distance is the minimum edit distance between two strings is the minimum number of editing operations:

- Insertion
- Deletion
- Substitution

Needed to transform one into the other. Examples include Spell correction, computational biology etc. Normally, each operation has cost of 1. However, in Levenshtein, the substitution cost can be taken as 2.

For two strings X of length n Y of length m. We define D(i,j) the edit distance between X[1..i] and Y[1..j] i.e., the first i characters of X and the first j characters of Y. The edit distance between X and Y is thus D(n,m).

Algorithm

//Initialization D(i,0) = i

D(0,j) = j

//Recurrence Relation:

For each i = 1…n

For each j = 1…m

D(i,j)= min(

D(i-1, j) + 1,

D(i, j-1) + 1,

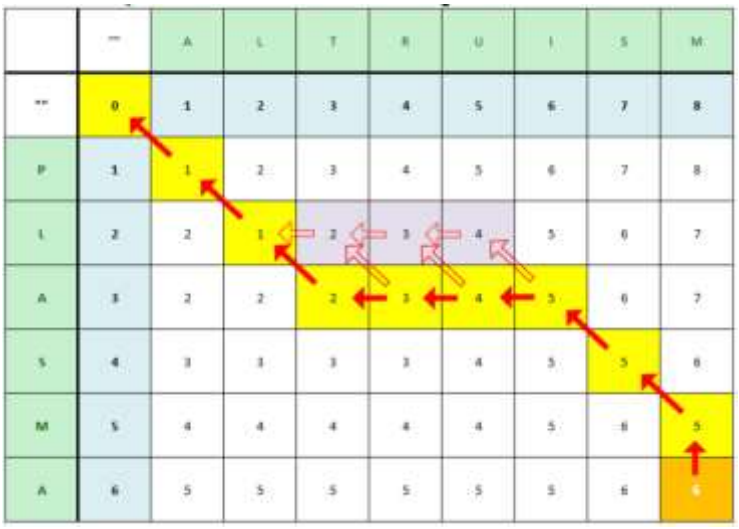
D(i-1, j-1) + 2, if $X(i) \neq Y(j)$

D(i-1, j-1) + 0 if X(i) = Y(j))

//Termination: D(n,m) is distance

Sol:

Marking Scheme: For each error; deduct: 0.5 marks



Example: Transform "PLASMA" into "ALTRUISM"

The steps are

Step	Comparison	Edit necessary	Total Editing
1.	"" to ""	no edits necessary!	"" to "" in 0 edits
2.	"P" to "A"	replace: +1 edit	"P" to "A" in 1 edit
3.	"L" to "L"	no edits necessary!	"PL" to "AL" in 1 edit
4.	"A" to "T"	replace: +1 edit	"PLA" to "ALT" in 2 edits
5.	"A" to "R"	insert: +1 edit	"PLA" to "ALTR" in 3 edits
6.	"A" to "U"	insert: +1 edit	"PLA" to "ALTRU" in 4 edits
7.	"A" to "I"	insert: +1 edit	"PLA" to "ALTRUI" in 5 edits
8.	"S" to "S"	no edits necessary!	"PLAS" to "ALTRUIS" in 5 edits
9.	"M" to "M"	no edits necessary!	"PLASM" to "ALTRUISM" in 5 edits
10.	"A" to "M"	delete: +1 edit	"PLASMA" to "ALTRUISM" in 6 edits

The above steps are shown as the yellow blocks below with the solid red arrows pointing in the opposite direction from the process above:

Question # 8 [CLO: 5] **[5 marks]**

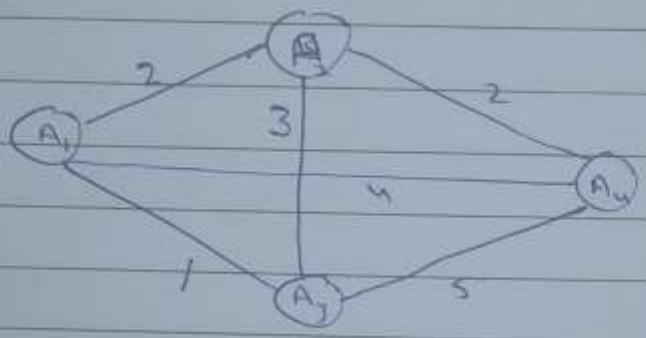
While visiting the Northern Areas of Pakistan you asked your travel guide to provide you information regarding the famous tourist points. Your travel guide handed you the following table showing four tourist points and the distances between these locations. You are determined to visit all tourist points once and return to the starting location besides reducing your travel expenses. Based on your knowledge in this course how would travel to these locations. Show all the computations and steps to solve it

	A_1	A_2	A_3	A_4
A_1	0	2	1	4
A_2	2	0	3	2
A_3	1	3	0	5
A_4	4	2	5	0

Marking Scheme: Correct Naming of Solution i.e. TSP: 2 Points followed by marks on correct steps and solution b/w 0 -3

Any Greedy Solution e.g. Steiner Tree etc without within 2 times of optimal solution = 2 Points (Maximum)

Solution:

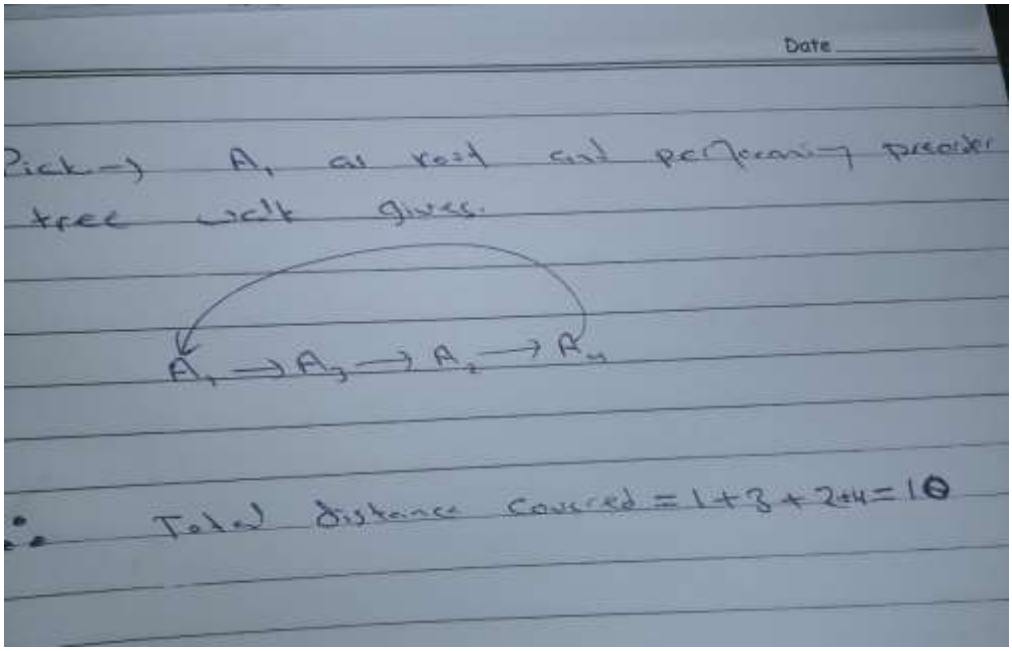


As the given distance graph holds triangle inequality. i.e

$$\text{Weight}(A_i, A_j) \leq \text{Weight}(A_i, A_k) + \text{Weight}(A_k, A_j)$$

So we can use 2-approximation polynomial time algorithm to solve Min metric TSP.

Using Prim's algorithm and starting from A_1 we get the following tree.



Question # 9 [CLO: 5] **[5 marks]**

Let $\varphi = (x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$ which is 3-CNF formula with m=2 variables and L=3 clauses.

a) Find the value of k = m + 2L [0.25 Points]
b) Prove that Φ is satisfiable iff G_Φ has a vertex cover of size k [3.75 Points]
   [Hint: You can directly think of a solution, or two step solution i) reduce 3SAT to Independent Set ii) Reduce Independent Set to k-Vertex Cover]
c) From k-vertex cover solution above, find whether x_1x_2 = {00, 01, 10, 11} is satisfiable or not [1 Point]

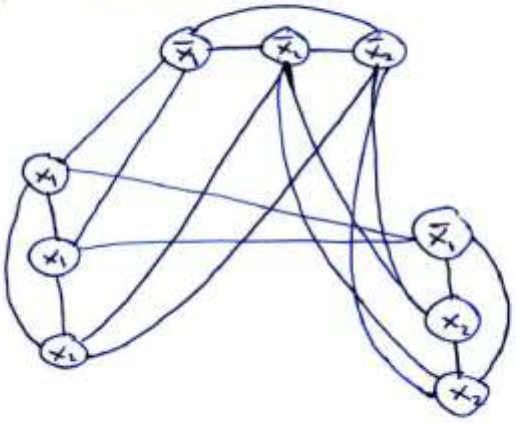
Solution:

Marking Scheme

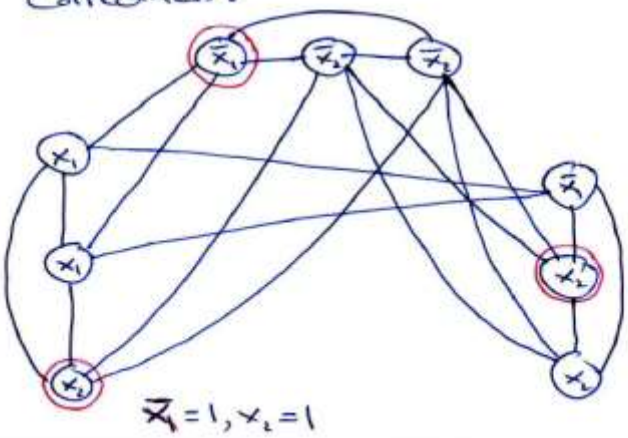
a) Marking Scheme: Correct Sol 0.5 Points; Any other / no solution: 0
b) Correct Sol 3.75 Points; Partial: 0.5-3.5 Points, No Solution: 0
c) Correct Sol for x_1x_2 0.25 Points each

a) $K = m + 2L = 2 + 2(3) = 8$

b) Reduction:



Correctness:



$x_1 = 1, x_2 = 1$

The red circled nodes are independent nodes and $\bar{x}_1 = 1$, and $x_2 = 1$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$$

$$= (0 \vee 0 \vee 1) \wedge (1 \vee 0 \vee 0) \wedge (1 \vee 1 \vee 1)$$

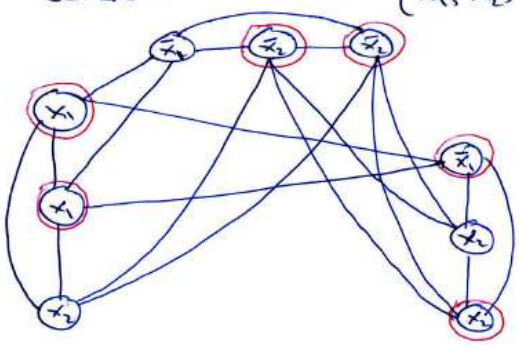
$$= 1$$

Hence True values for independent set satisfies the 3-SAT formula.

(4)(b)

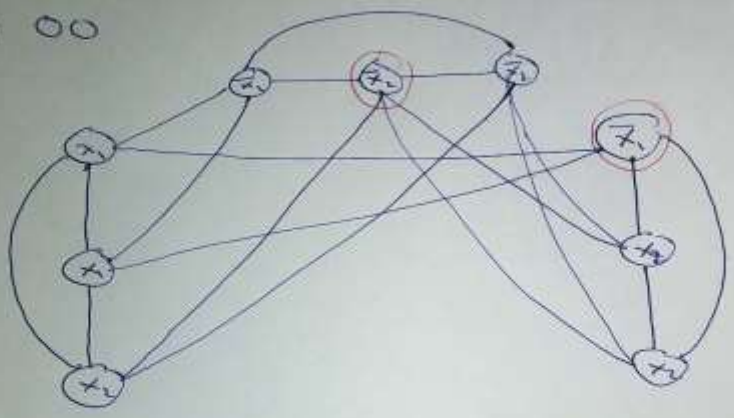
Let $V = \{x_1, \bar{x}_1, x_2, \bar{x}_2\}$ be the vertices of the reduction graph. An independent set for the graph is $S = \{\bar{x}_1, x_2\}$. Then the vertex cover Cover is given by

Vertex cover $= V - S = \{x_1, \bar{x}_2\}$



9c: For 00, 01, 10 and 11 to satisfy the 3-sat formula they should be ~~reduform~~ and independent nodes set for all clauses and then the inverse will form set cover.

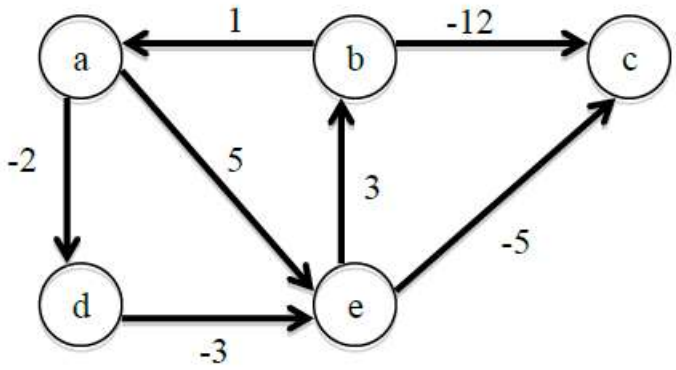
For 00



We can not select a node in the first clause that is independent of other clause nodes and also true. Hence ~~Empty~~ It wouldn't satisfy the formula. Similarly we can check for other values

Question # 10 [CLO: 3] **[5 marks]**

Apply the Bellman-Ford shortest path algorithm on the following graph, starting from vertex "a". Show all steps



Solution

4 Points: There is negative cycle. Students should complete and optionally show 2nd loop of the algorithm. -0.5 for each wrong entry

# edges	A	b	C	d	e
0	0	∞	∞	∞	∞
1	0	∞	∞	-2	5
2	0	8	0	-2	-5
3	0	-2	-10	-2	-5
4	-1	-2	-14	-2	-5

1 Point for this table or showing of correct Prev [vertex]

Vertex	a	b	c	d	e
Prev[vertex]	b	e	b	a	d