National University of Computer & Emerging Sciences, Karachi
Fall-2018 Department of Computer Science
Mid Term-2
12th November 2018, 11:00 AM – 12:00 PM

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

Instructions:

- Return the question paper.
- Read each question completely before answering it. There are **5 questions** on **2 pages.**
- In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper.

Time: 60 minutes. **Max Marks: 12.5**

Question # 1 [2 marks]

Use Topological Sort to determine whether **Figure: 1** is a Directed Acyclic Graph (DAG) or not. Show all steps.

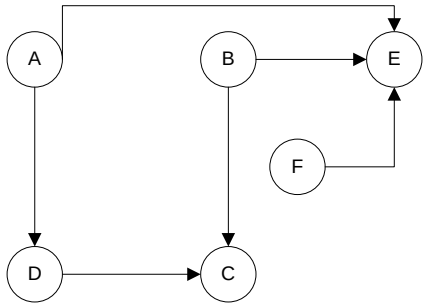


Figure: 1

Solution#1 [1.5 Points for steps and 0.5 for writing whether DAG or not]

Solution:
Find DFS_Traversal

DFS(A)

DFS(A) -> DFS(D), DFS(E)
DFS(D) -> DFS(C)
DFS(E) -> dead end
DFS(C) -> dead end

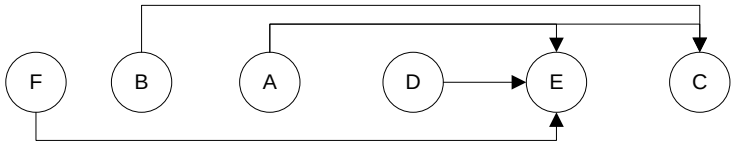
DFS(B) -> DFS(C) or DFS(E), already covered

DFS(F)
DFS (F) to DFS(E), E already covered

DFS_Traversal = C, E, D, A, B, F

Reverse

F,B,A,D,E,C



DAG since all edges from left to right

Question # 2 **[4 marks]**

A) Execute the following dynamic programming algorithm on the provided input strings X and Y and show the output with intermediate steps.

X=ACTCSSSAAA

Y=ACCSGSSAAA

Z is a matrix of size (m+1)*(n+1) for results of the provided algorithm. Here ***m*** is the length of sequence (string) X and ***n*** is the size of sequence Y.

$$Z[i,j] = \begin{cases} Z[i-1,j-1]+1 & if(i>0 \;\&\&\; j>0 \;\&\&\; X[i]=Y[j]) \\ \max(Z[i-1,j], Z[i,j-1]) & if(i>0 \;\&\&\; j>0 \;\&\&\; X[i]\neq Y[j]) \\ 0 & if(i=0 \;||\; j=0) \end{cases}$$

You are required to fill matrix Z (as provided below) for the above example.

X=ACTCSSSAAA

Y=ACCSGSSAAA

B) What is going on in this algorithm? Explain in 1-2 sentences, otherwise answer will not be checked.

Solution#2(A) [Check the concept. If concept is clear and table is perfect, then Full Marks
If concept is clear but error propagated from somewhere 2 Points
Some Attempt: 0.5 – 1.5 Point
No Attempt: 0 Point

	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10
X0	0	0	0	0	0	0	0	0	0	0	0
X1	0	0+1=1	0,1=1	0,1=1	0,1=1	0,1=1	0,1=1	0,1=1	0+1=1	0+1=1	0+1=1
X2	0	0,1=1	1+1=2	1+1=2	1,2=2	1,2=2	1,2=2	1,2=2	1,2=2	1,2=2	1,2=2
X3	0	0,1=1	1,2=2	2,2=2	2,2=2	2,2=2	2,2=2	2,2=2	2,2=2	2,2=2	2,2=2
X4	0	0,1=1	1+1=2	2+1=3	3,2=3	3,2=3	3,2=3	3,2=3	3,2=3	3,2=3	3,2=3
X5	0	0,1=1	1,2=2	2,3=3	3+1=4	4,3=4	3+1=4	3+1=4	4,3=4	4,3=4	4,3=4
X6	0	0,1=1	1,2=2	2,3=3	3+1=4	4,4=4	4+1=5	4+1=5	5,4=5	5,4=5	5,4=5
X7	0	0,1=1	1,2=2	2,3=3	3+1=4	4,4=4	4+1=5	5+1=6	6,5=6	6,5=6	6,5=6
X8	0	0+1=1	1,2=2	2,3=3	3,4=4	4,4=4	4,5=5	5,6=6	6+1=7	6+1=7	6+1=7

X9	0	0+1=1	1,2=2	2,3=3	3,4=4	4,4=4	4,5=5	5,6=6	6+1=7	7+1=8	7+1=8
X10	0	0+1=1	1,2=2	2,3=3	3,4=4	4,4=4	4,5=5	5,6=6	6+1=7	7+1=8	8+1=9

Solution#2(B)

The above algorithm is finding the length of longest subsequence common to all given sequences.

Question # 3 **[1 marks]**

Differentiate between greedy and dynamic algorithms.

Solution#3

- Both techniques are optimization techniques.
- A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum.
- In DP, to solve a given problem, we need to solve different parts of the problem (subproblems), ***then combine the solutions of the subproblems to reach an overall solution***.

Question # 4 **[0.25*8=2 marks]**

Provide the worst case time complexity and the name design technique of the following algorithms.

Solution#4

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/ElogE)	Greedy
Prim's algorithm for finding MST	O(ElogV)	Greedy
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 **[2+1.5=3.5 marks]**

A) Design a time efficient algorithm for the computation of **maximum** spanning tree from a provided graph G(V,E).

Solution#5(A)

```
1.  Q ← ∅
2.  for each u ∈ V
3.      do key[u] ← 0
4.         π[u] ← NIL
5.         INSERT(Q, u)
6.  UPDATE-KEY(Q, r, 0)   ► key[r] ← 0
7.  while Q ≠ ∅
8.         do u ← EXTRACT-MAX(Q)
9.            for each v ∈ Adj[u]
10.               do if v ∈ Q and w(u, v) > key[v]
11.                     then π[v] ← u
12.                          UPDATE-KEY(Q, v, w(u, v))
```

π[u]→ Parent of u
Key[u]→ current weight of vertex (u)

Or Student can describe in words b/c algorithm memorization can be tricky

Since the minimum spanning tree algorithm works for negative edge costs, an obvious solution is to replace all the edge costs by their negatives and use the minimum spanning tree algorithm. Alternatively, we can change the logic so that "is replaced by", *min* by *max*, and vice versa.

B) Apply designed algorithm from (A) on the following graph (**Figure: 2**) to compute the maximum spanning tree**:**

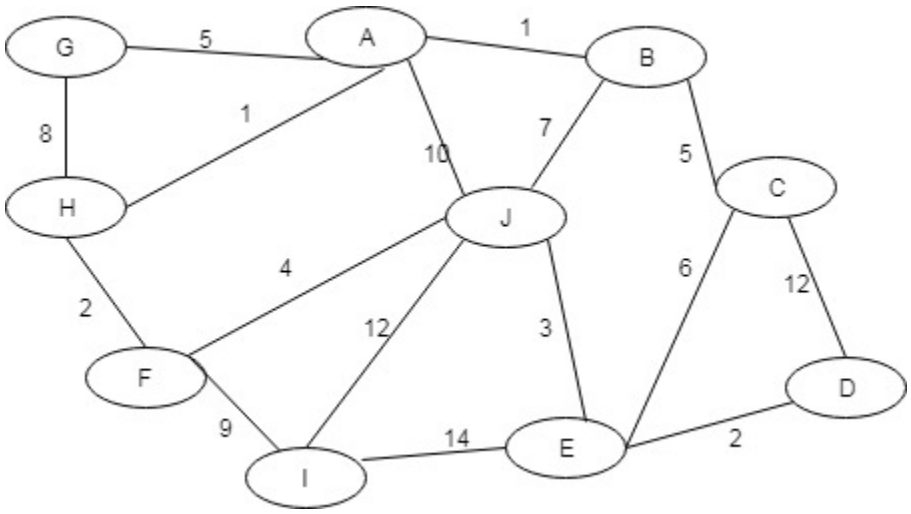


Figure: 2

Solution#5(B)