CS575 Homework 3

Submit a scanned pdf file through blackboard by 8:59am on April 3, and submit a hard copy at the beginning of the Class on April 3. (The soft and hard copies should be exactly the same.)

PLEASE PRINT DOUBLE SIDED ONLY.

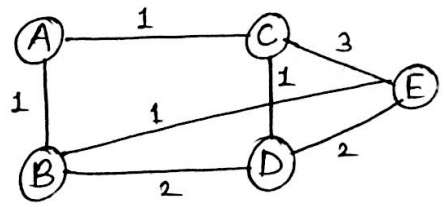
First Name ANIRUDDHA **Last Name** TEKADE **Section** 02

I promise to follow the academic honesty requirements of the Binghamton University. I agree that I will fill out and sign an official form that I have cheated if I get caught cheating. I understand that this form will be stored by the university. Furthermore, I understand that the minimum penalty for cheating is getting a grade of 0 for this assignment

Sign Aniruddha

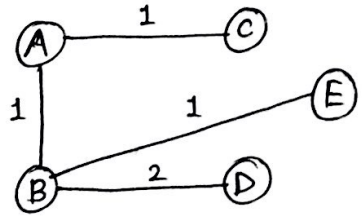
1.[10%] Show that there can be more than one minimum spanning tree in an undirected graph by giving an example

Sol": Let's consider following graph -

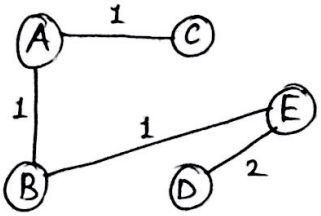


MSTs possible ⇒

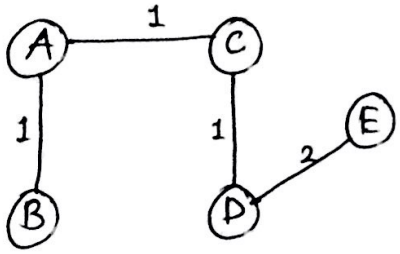
I



II



III



2.[10%] Briefly yet clearly describe how to tell whether an input graph is connected or not.

⇒ (1) If the input graph has connectivity between every pair of vertices, the graph is connected.

(2) In other words, if we can get from any vertex of graph to any other vertex following a sequence of edges, the graph is connected.

(3) No vertices are unreachable in a connected graph.

3. [10%] Briefly describe how to determine whethera directed graph has a cycle using the depth first search algorithm

⇒ (1) Perform DFS algorithm on input graph. DFS actually produces tree (which does not have a cycle present)

(2) But cycle may exist only if there is a back-edge present in the graph. Back-edge is an edge from a node to an ancestor in the tree.

(3) To detect a back edge, we can keep track of visited vertices (using recursive calls). If we reach a particular vertex that is already present in record, this means there exists a back edge and hence a cycle. Our function def^n can return "TRUE".

4. [15%]Suppose that we have n professional wrestlers and a ist of r pairs of wrestlers for which there are rivalries. (Between any pair of wrestlers, there may or may not be a rivalry.) Give an O(n+r) time algorithm that determines whether it is possible to designate some of the wrestlers to Red team and the remainder as Blue team such that each rivalry is between a Red and a Blue team member. If it is possible to perform such an assignment, your algorithm should produce it.

Soln: The simples possible approach for the above problem is as follows ⇒

We can consider that there are –

'n' → vertices (wrestlers) &

'r' → edges (rivalry).

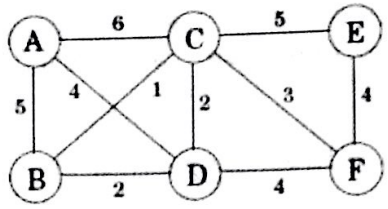
The matric factor for algorithm can be weight/ distance between wrestlers. Odd corresponds to Blue team and Read corresponds to Even distances.

Generalized Possible Pseudocode ⇒

1. While (all vertices are not visited)
2. Perform BFS on graph 'G'.
3. Designate Red Team to wrestler having even unit of distance/ weight of edge.
4. Designate Blue Team to wrestler with odd no. of distance.
5. Repeat through ① to ④ untill condition is true.

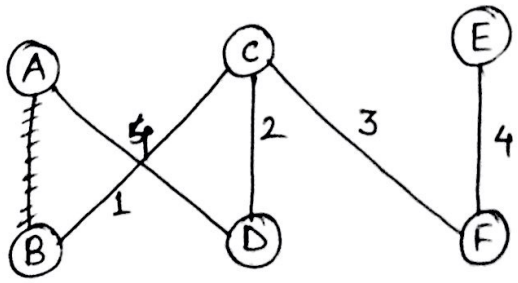
Analysis: BFS needs O(n+r), Process of Design: O(n) O(r) to check edges. **Overall = O(n+r)**.

5. [20%] Use Prim's algorithm to find the minimum spanning tree in the following graph starting from vertex A. Show every step.

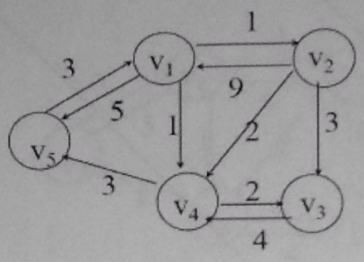


STEP #	DISCOVERED NOTES	A	B	C	D	E	F
(1)	{ }	0	∞	∞	∞	∞	∞
(2)	{ A }	-	5	6	4	∞	∞
(3)	{ A, D }	-	2	2	-	∞	4
(4)	{ A, D, C }	-	1	-	-	5	3
(5)	{ A, D, C, B }	-	-	-	-	5	3
(6)	{ A, D, C, B, E }	-	-	-	-	4	~~3~~
(7)	{ A, D, C, B, E, F }	-	-	-	-	-	-

Final Minimum Spanning Tree ⇒



6. [15%] Using Floyd's algorithm, find all pairs shortest paths in the following graph.



Sol^n:

When, $K = \emptyset$

$D^0 \Rightarrow$

	1	2	3	4	5
1	0	1	∞	1	5
2	9	0	3	2	∞
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	∞	∞	∞	0

$P \Rightarrow$

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

When, $K = 1$

$D^1 \Rightarrow$

	1	2	3	4	5
1	0	1	∞	1	5
2	9	0	3	2	14
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	4	∞	4	0

$P \Rightarrow$

0	0	0	0	0
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0
0	1	0	1	0

When, $K = 2$

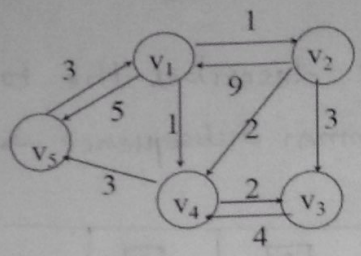
$D^2 \Rightarrow$

0	1	4	1	5
9	0	3	2	14
∞	∞	0	4	∞
∞	∞	2	0	3
3	4	7	4	0

$P \Rightarrow$

0	0	0	0	0
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0
0	1	2	1	0

6. [15%] Using Floyd's algorithm, find all pairs shortest paths in the following graph.



Soln: (contd...)

When, K=3

$D^3 \Rightarrow$

	1	2	3	4	5
1	0	1	4	1	5
2	9	0	3	2	14
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	4	7	4	0

$P \Rightarrow$

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

When K=4

$D^4 \Rightarrow$

	1	2	3	4	5
1	0	1	3	1	4
2	9	0	3	2	5
3	∞	∞	0	4	7
4	∞	∞	2	0	3
5	3	4	6	4	0

$P \Rightarrow$

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

When K=5

$D^5 \Rightarrow$

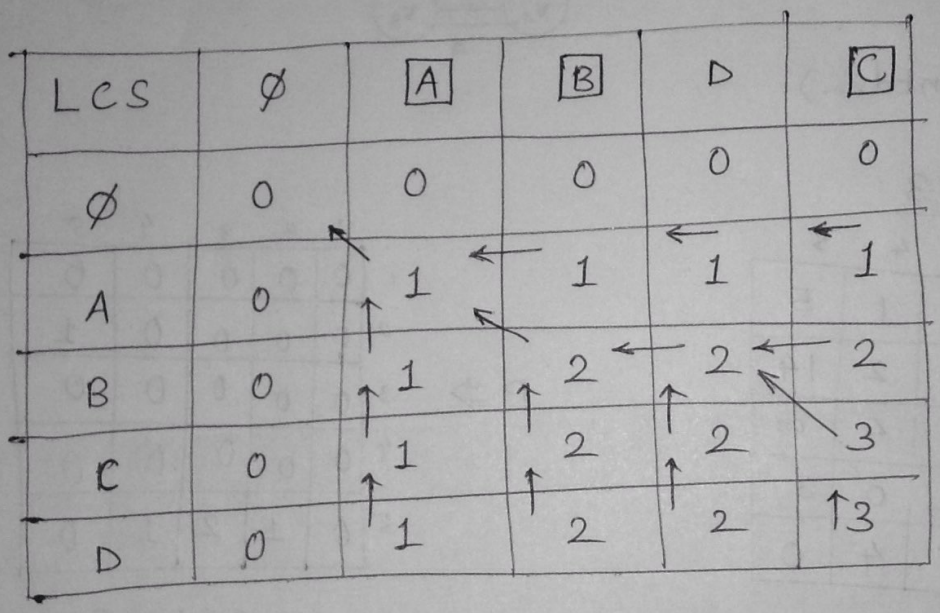
	1	2	3	4	5
1	0	1	3	1	4
2	8	0	3	2	5
3	10	11	0	4	7
4	6	7	2	0	3
5	3	4	6	4	0

$P \Rightarrow$

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

7. [20%] Find the longest common subsequence between two strings ABCE and ABDC via dynamic programming.

Sol^{n}: Following table describes the total processing of Longest Common Subsequence →



LCS	∅	A	B	D	C
∅	0	0	0	0	0
A	0	1	1	1	1
B	0	1	2	2	2
C	0	1	2	2	3
D	0	1	2	2	3

The longest common subsequence found = "ABC".