Rojin and Suresh Assignment 15

Group - 3

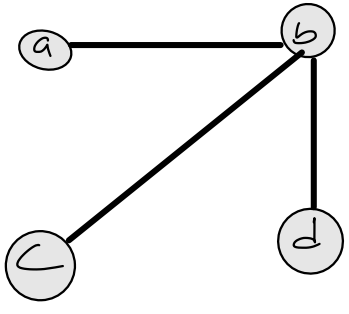
Problem 2

If an adjacency matrix has rows {0, 1, 0, 0}, {1,0,1,1},{0,1,0,0}, and{0,1,0,0}, what is the corresponding adjacency list? You can assume any names for your vertices.

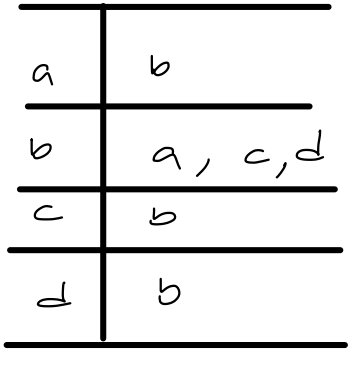
Adjacency Matrix

	a	b	c	d
a	0	1	0	0
b	1	0	1	1
c	0	1	0	0
d	0	1	0	0

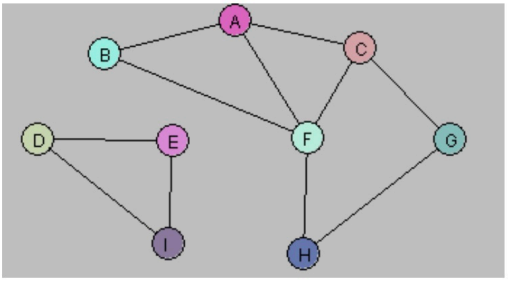
Graph



Adjacency List



Problem 1



A) Is G connected?

→ No. the given graph is not connected.

How many connected compants are in G?

→ There are to connected components are in graph G.

B) Is there path from B to D?

→ No, because there is no connection or path between two components in the given graph G.

C) Is there spanning tree for G. Explain

→ First, given graph is not a tree. There is a cycle.

No, there is no spanning thee for graph G, because, there is neither any path nor any connection between given graph component.

Problem 3

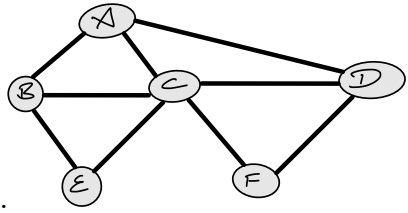
A simple connected undirected graph must have a cycle if

A any vertex can be reached from some other vertex.

B the number of paths is greater than the number of vertices.

C the number of edges is equal to the number of vertices.

Problem 4 - DFS



Step 1: Pick B as start point and mark B as visited and push.

S.push(B)

STACK
B

Step 2: Pick A as adjacent of B and mark A as visited and push. S.push(A)

T = { BA}

STACK
A
B

Step 3: Pick D as adjacent of B and mark D as visited and push.

S.push(D)

T = { BA, AD }

STACK
D
A
B

Step 4: Pick F as adjacent of D and mark F as visited and push.

S.push(F)

T = { BA, AD, DF }

STACK
F
D
A
B

Step 5: Pick C as adjacent of F and mark C as visited and push. S.push(C)

T = { BA, AD, DF, FC}

STACK
C
F
D
A
B

Step 6: Pick E as adjacent of C and mark E as visited and push.

S.push(E)

T = { BA, AD, DF, FC, CE}

STACK
E
C
F
D
A
B

Pop steps:

S.pop() ➔ E

S.pop() ➔ C

S.pop() ➔ F

S.pop() ➔ D

S.pop() ➔ A

S.pop() ➔ B

Now whole stack is empty.

STACK
E
C
F
D
A
B

STACK
C
F
D
A
B

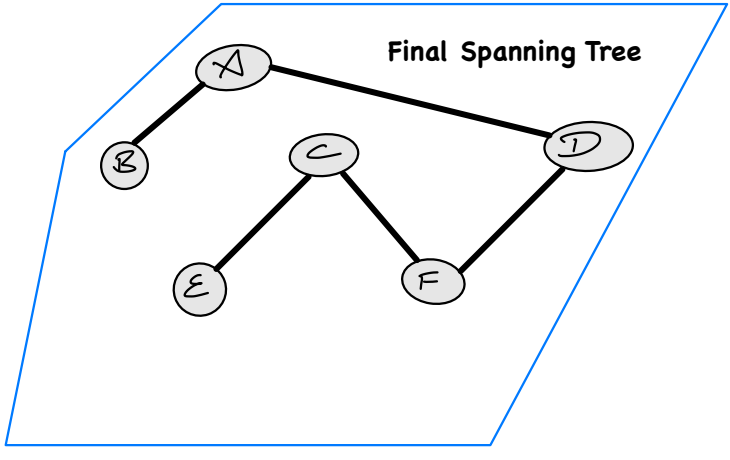
STACK
F
D
A
B

STACK
D
A
B

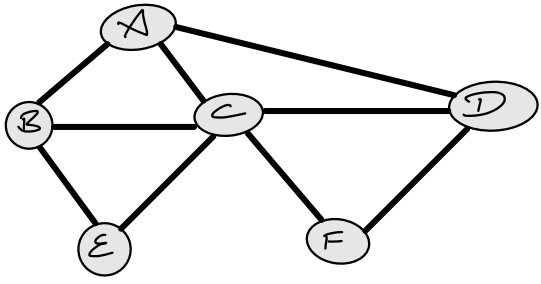
STACK
A
B

STACK
B

STACK



Problem 5 - BFS



T= { }
Q = [ ]
Step 1:
Start with B, add (B), Mark
Q = [B]

Step 2:
Loop is Started.
Add Adjacent's of B
Dequeue() -> B Q= [A | C | E]
T = { (B, A) , (B, C), (B, E)}

Step 3:
Add Adjacent's of A
Dequeue() -> A Q= [C | E | D]
T = { (B, A) , (B, C), (B, E), (A, D)}

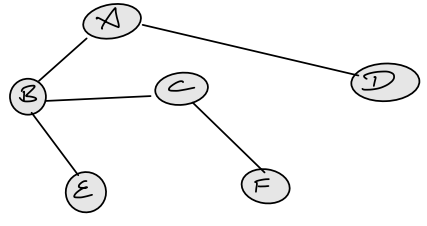
Step 4:
Add Adjacent's of C
Dequeue() -> C Q= [E | D | F]
T = { (B, A) , (B, C), (B, E), (A, D), (C, F) }

Step 5:
There is no unvisited Adjacent's of E
Dequeue() -> E Q= [D | F]
T = { (B, A) , (B, C), (B, E), (A, D), (C, F) }

Step 6:
There is no unvisited Adjacent's of D
Dequeue() -> D Q= [F]
T = { (B, A) , (B, C), (B, E), (A, D), (C, F) }

Step 7:
There is no unvisited Adjacent's of F
Dequeue() -> F Q= [ ]
T = { (B, A) , (B, C), (B, E), (A, D), (C, F) }

Final Spanning Tree



No, The Spanning Tree is different from Problem 4.