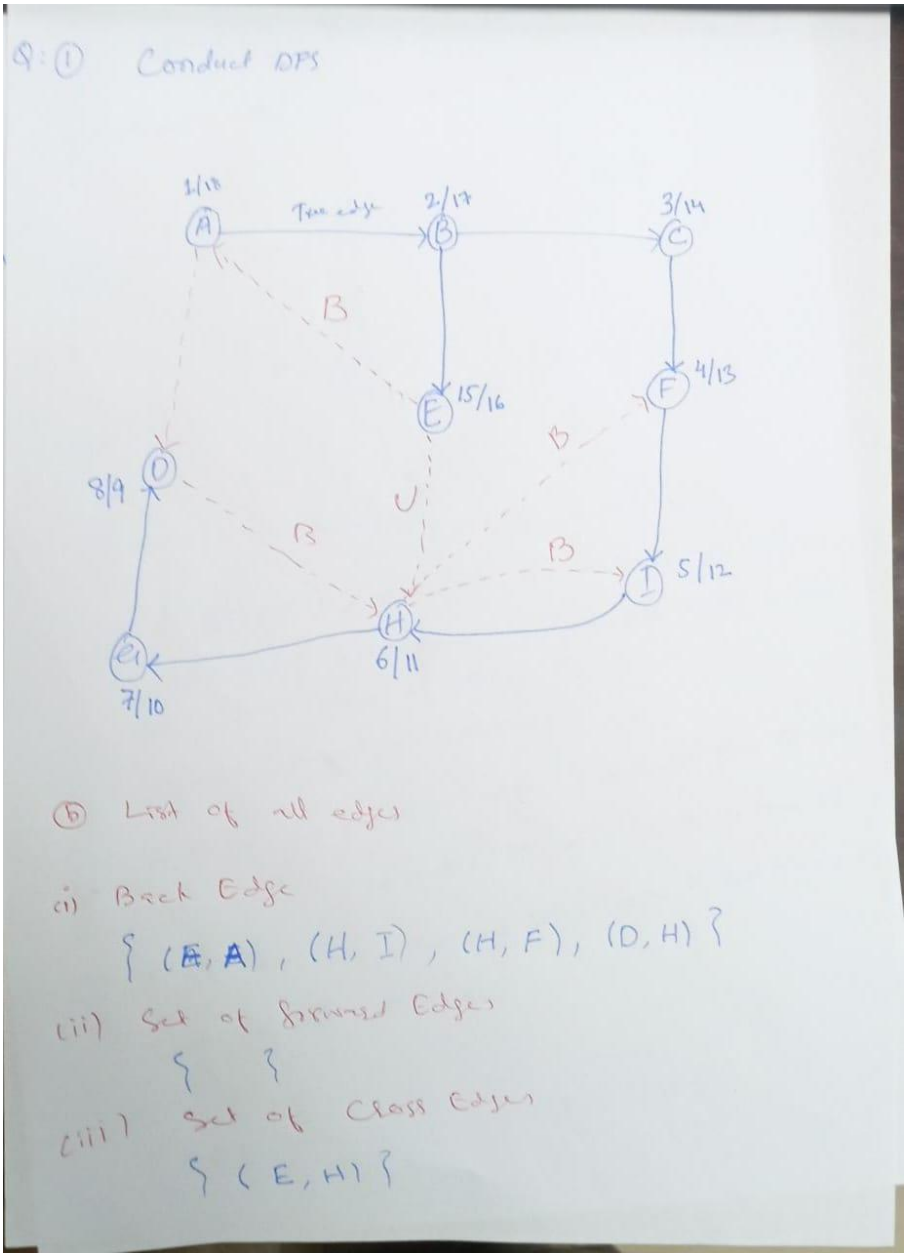
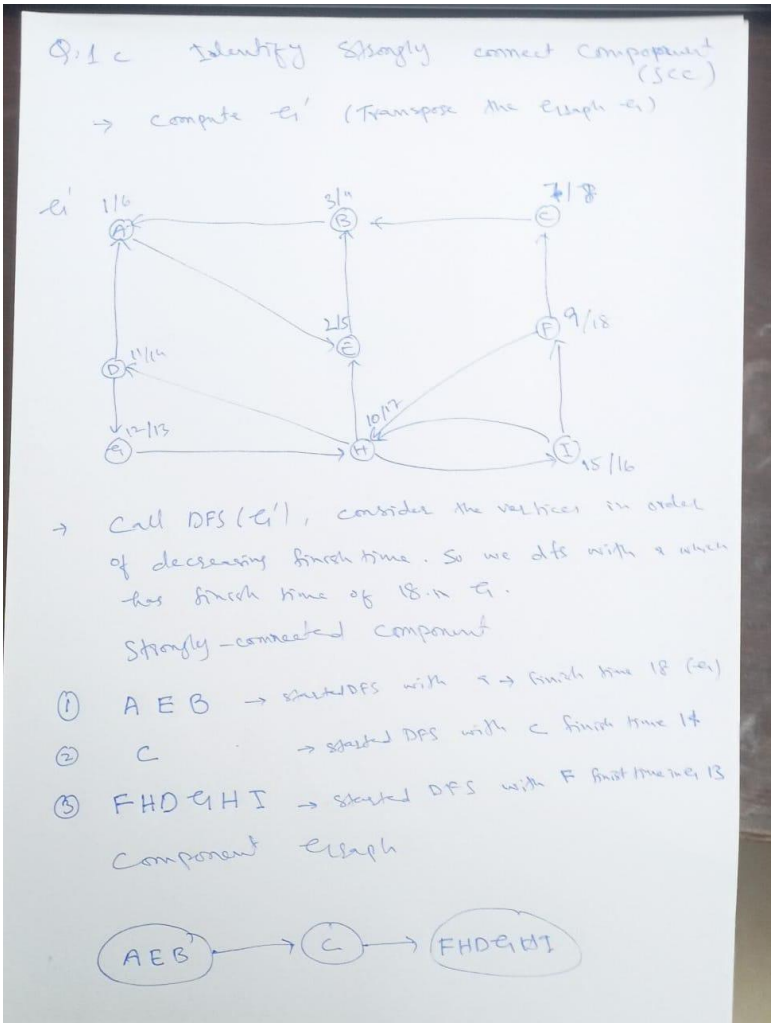
Q1: Conduct a DFS for the following graph. (Label each vertex u with the start time and the finish time) or (Show all steps showing stack and Visit Order). You should start the traversal from vertex A, and follow the alphabetic order whenever you need to make choices.

b) List all edges (back edges,forward edges, cross edges) that belong to each of the following sets: [5 Points]
c) Identify the strongly connected components and draw the component graph. [10 Points]





Q2: It is possible to test whether a graph is bipartite or not using a Depth–first search (DFS) algorithm. There are two ways to check for bipartite graphs:

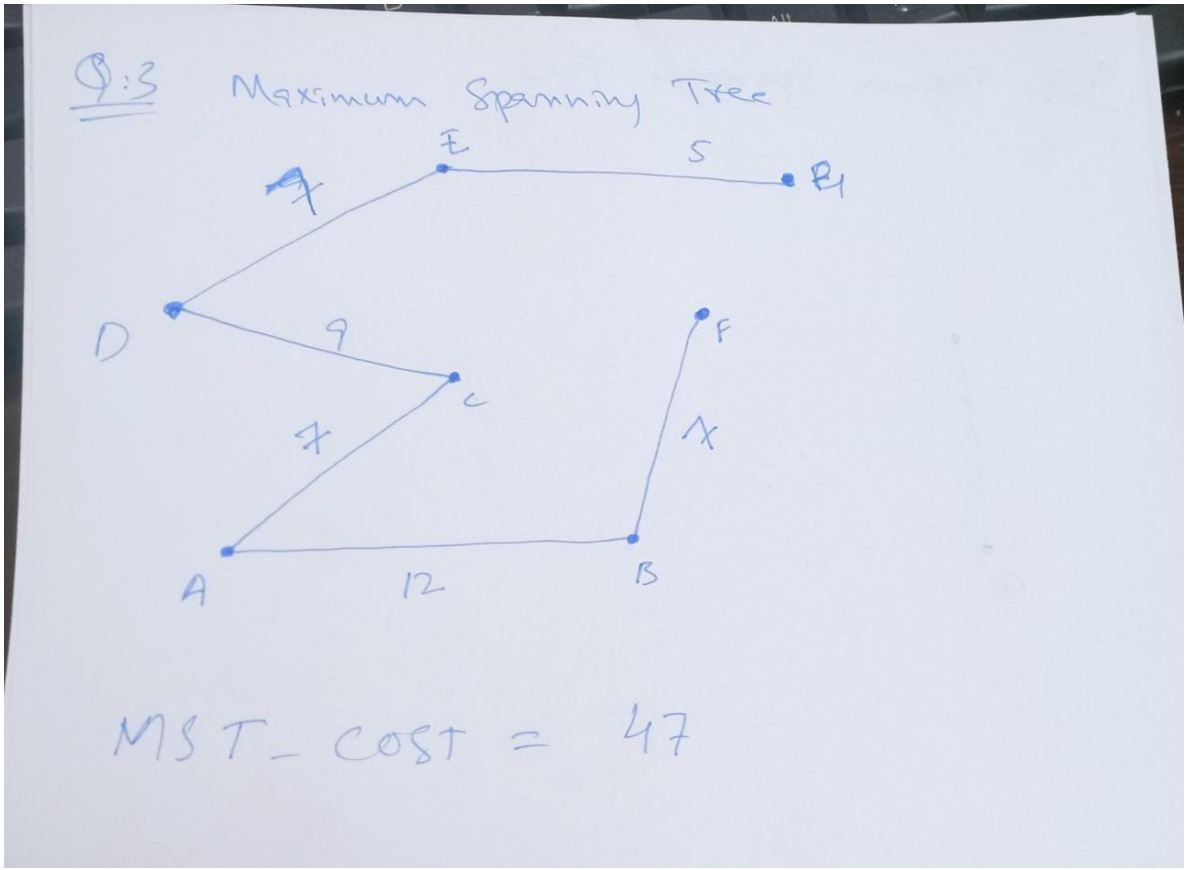
1. A graph is bipartite if and only if it is 2–colorable.
2. A graph is bipartite if and only if it does not contain an odd cycle.

(a) do a dfs but mark each node as belonging to v1 or v2

```
def dfs(G,u,c):
    if colour[u]== -1:  # not coloured
        colour[u] = c
    elif colour[u]!=c:
        return False # graph is not bipartite
    for v in G[u]:
        ans&=dfs(G,v, not(color))
    return ans
```

b) if it has a cycle of odd length. then 0-1-0-1

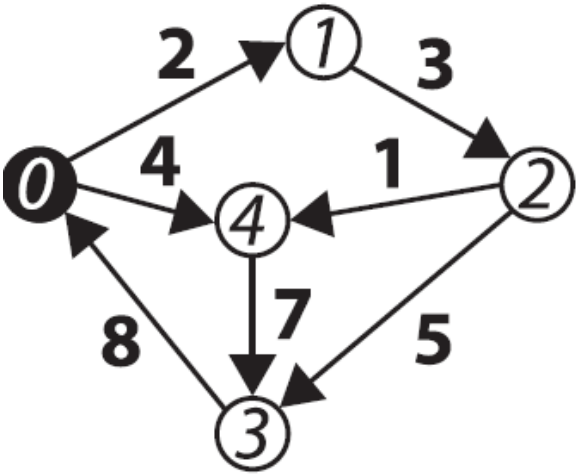
Q3: Use Prim’ s algorithm to compute a maximum spanning tree for the following graph shown in figure 2. Use node A as the root.



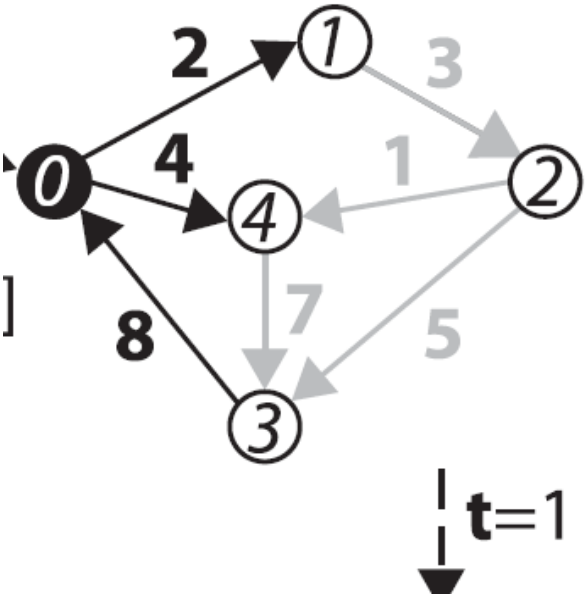
B) If the edge weights in your graph are all different from each other, then your graph has a unique minimum spanning tree, so Kruskal's and Prim's algorithms are guaranteed to return the same tree.

But it is possible that Prim's and Kruskal's algorithm produce different maximum spanning tree. Cost MST will be always same.

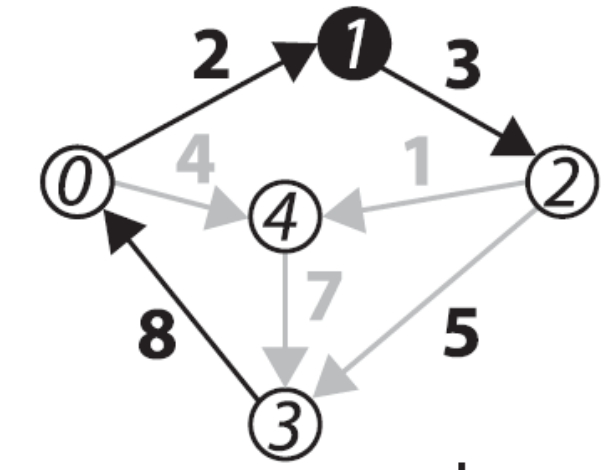
Q4: Using Floyd-Warshall, find all pairs shortest path for the given Figure 3 (D_0 weight matrix is also provided). Discuss its time and space complexity as well



	0	1	2	3	4
0	0	2	∞	∞	4
1	∞	0	3	∞	∞
2	∞	∞	0	5	1
3	8	∞	∞	0	∞
4	∞	∞	∞	7	0

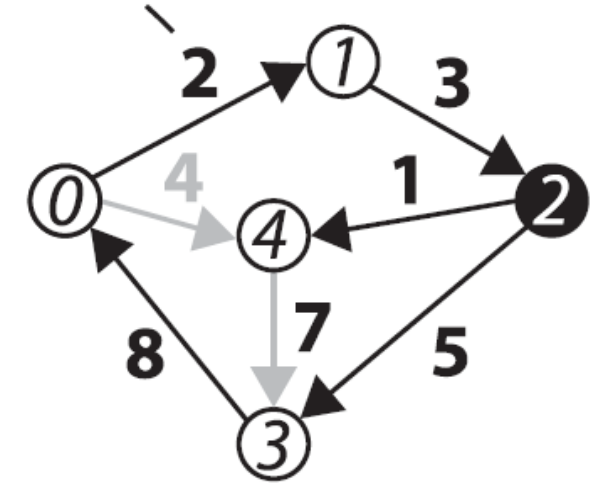


	0	1	2	3	4
0	0	2	∞	∞	4
1	∞	0	3	∞	∞
2	∞	∞	0	5	1
3	8	10	∞	0	12
4	∞	∞	∞	7	0



	0	1	2	3	4
0	0	2	5	∞	4
1	∞	0	3	∞	∞
2	∞	∞	0	5	1
3	8	10	13	0	12
4	∞	∞	∞	7	0

t=2



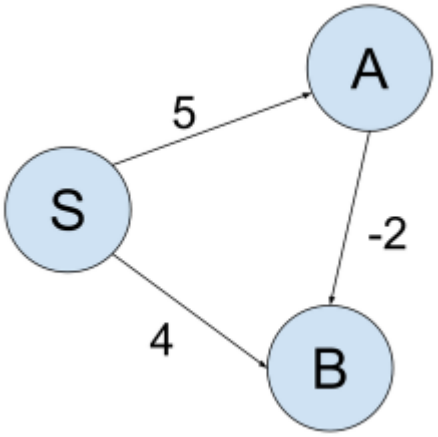
	0	1	2	3	4
0	0	2	5	10	4
1	∞	0	3	8	4
2	∞	∞	0	5	1
3	8	10	13	0	12
4	∞	∞	∞	7	0

Q5: Go through the lecture of Johnson's Algorithm part 1, Johnson's Algorithm part 2 and write its summary.

Give marks if students write summary of Johnson's Algorithm.

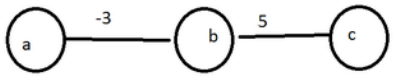
Q6: Give a simple example of a directed graph with negative-weight edges for which Dijkstras algorithm produces.

a)

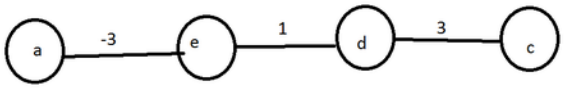


b)

Lets say you have the following 2 paths from a -> c



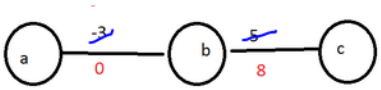
path cost = 2



path cost = 1

So you will select the second path as that has the min cost.

Now lets add 3 to make all edges weights positive(in the hope we can run the Dijkstra's algo):



path cost = ~~7~~ 8

path cost = ~~1~~ 10

See the shortest path now has become the path A(which is wrong). This happens due to the fact that path B has more nodes to reach from a->c. And hence we will be adding 3 to the path cost for all the edges in the path. Number of edges in path A is less and hence that path gets the lower path cost.

Q7:

Read Book, Lectures or Search through Internet to explain the worst time complexities of BFS, DFS, Kruskal's and Prims in your own words using (i) Adjacent Matrix (ii) Adjacent List [10 Points]

Algorithm	Best	Average	Worst
DEPTH-FIRST SEARCH	$V+E$	$V+E$	$V+E$
BREADTH-FIRST SEARCH	$V+E$	$V+E$	$V+E$
DIJKSTRA'S ALGORITHM PQ	$(V+E)$ log V	$(V+E)$ log V	$(V+E)$ log V
DIJKSTRA'S ALGORITHM DG	V^2+E	V^2+E	V^2+E
BELLMAN-FORD ALGORITHM	$V*E$	$V*E$	$V*E$
FLOYD-WARSHALL ALGORITHM	V^3	V^3	V^3
PRIM'S ALGORITHM	$(V+E)$ log V	$(V+E)$ log V	$(V+E)$ log V