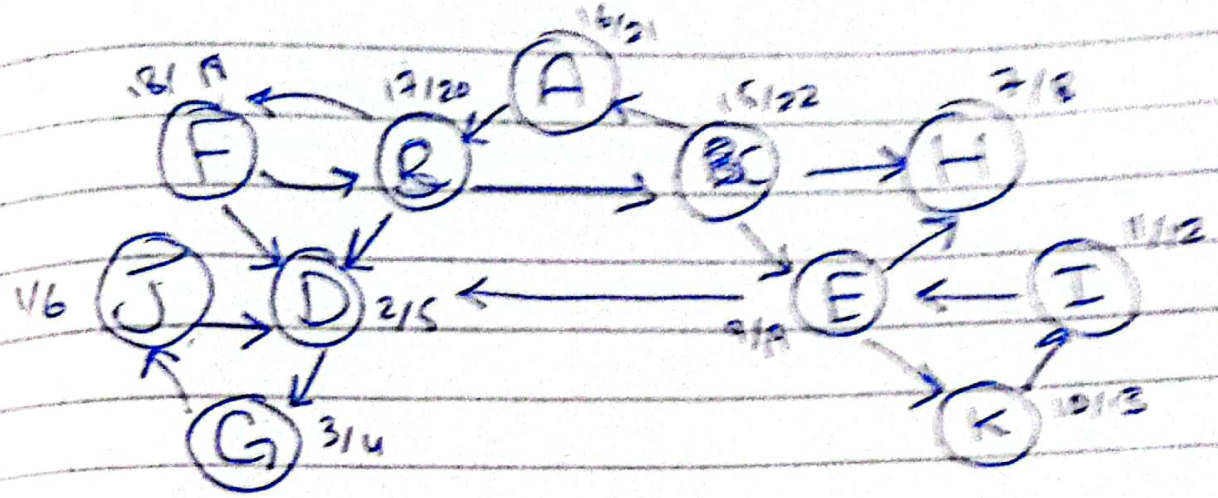
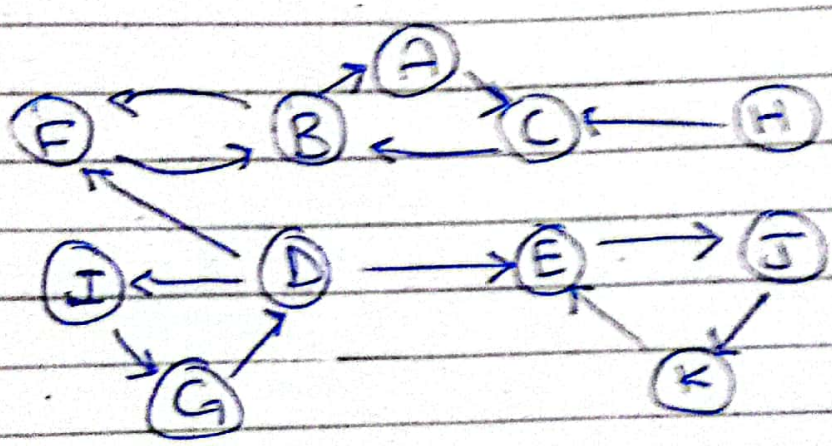
Rohan Jamadi
L21-5625

Assignment 3

(Q1)





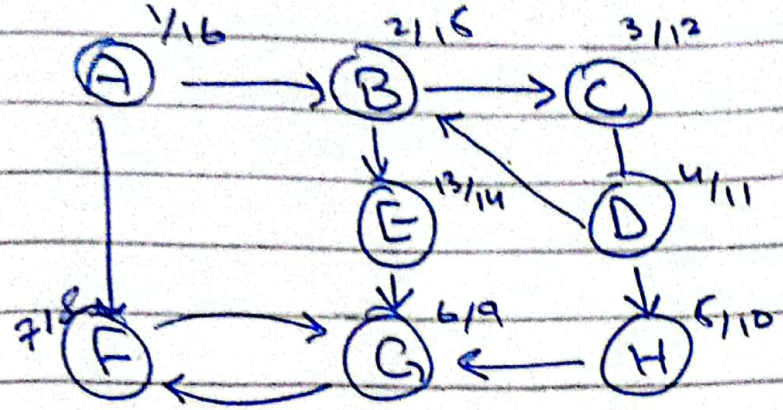
dec = I, D, D, H, E, J, K, C, F, B, A

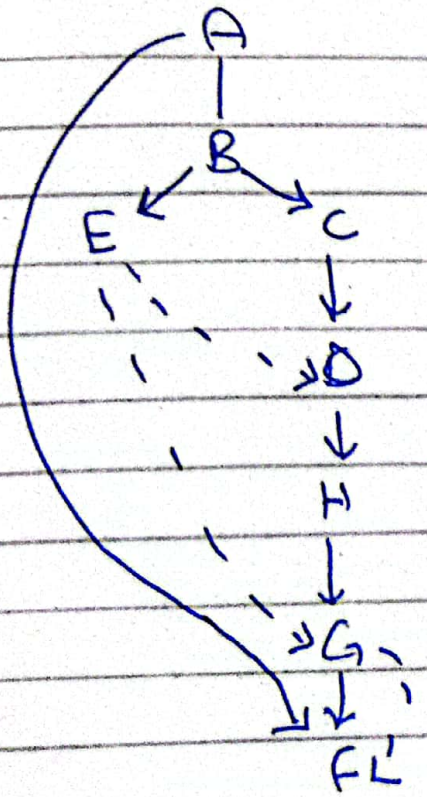
1- I, G, D
2- H
3- E, J, K
4- C, F, B, A

4 cycles

(Q2)

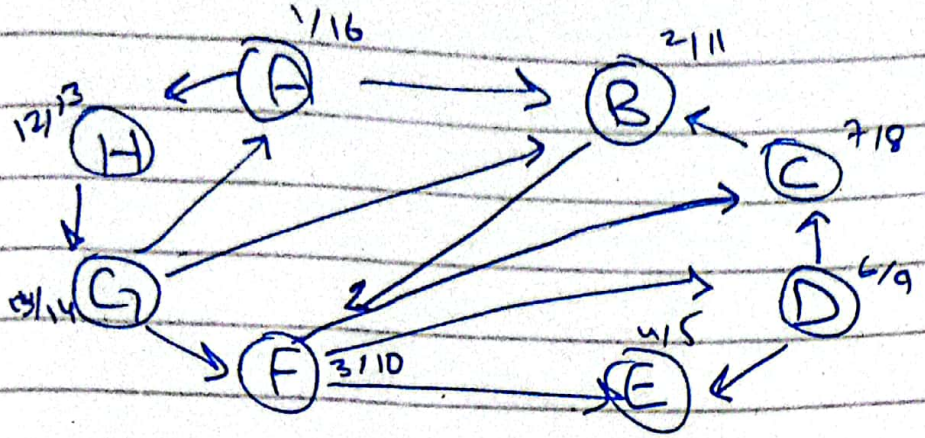
a)

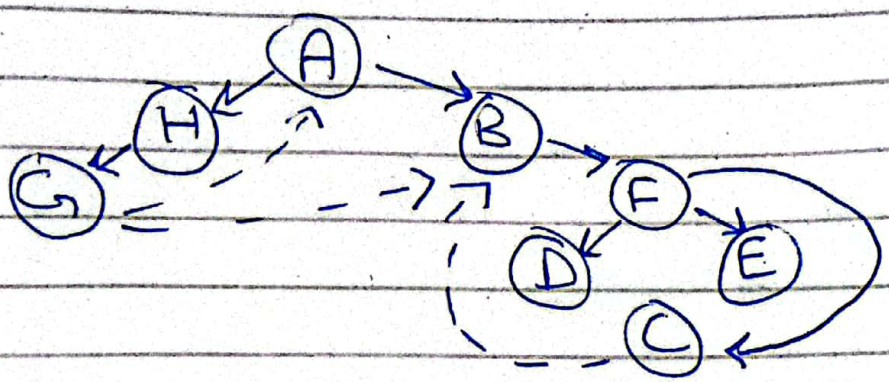




A, B, E, C, D, H, G, F

b)

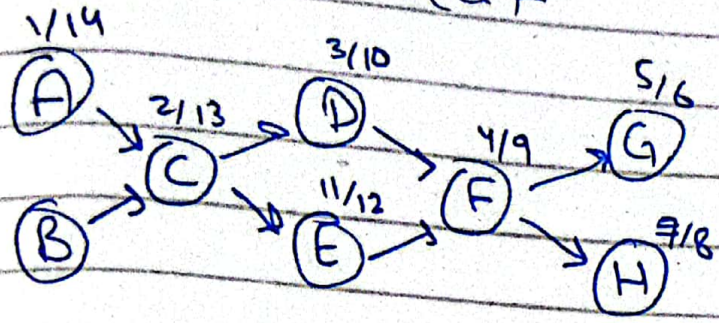




A H G B F D C E

(Q3)

(a)



(b)

Sink : G , H

Source : A, B

(c)

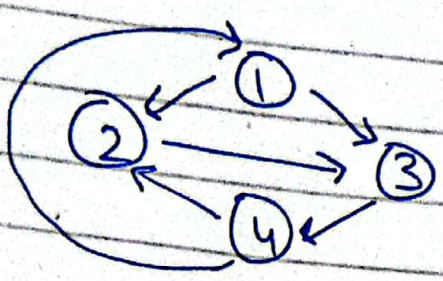
B, A, C, E, D, F, H, G

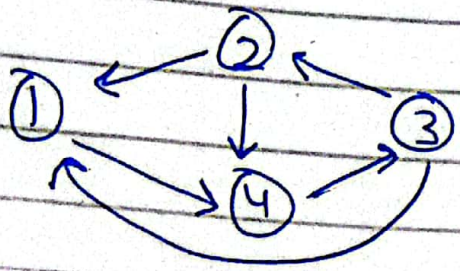
(d)

No. of topological orders = 2

No matter if you start from A or B.

-(04)-



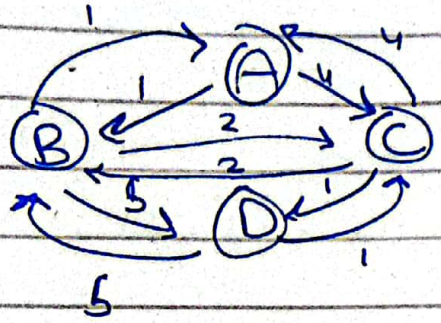



```
Reverse Graph ( Graph ) {
    reverse_graph = V → [] in graph
for  v in graph
    visited = [∞,∞,∞ ...]
    Q =  deque (v)
    while Q not empty
    cvv_v = Q.pop()
    if cvv_v not visited
        visit cvv_v
    for neighbors in G(cvv_v].
    reverse_graph(neighbors] = cvv_v
    if  neighbors inot visited
        Q. push[neighbors]
return  reverse_graph   }
```

(Q5)

(a)

It is false. Dijkstra's shortest path algo takes $O(|n| \log |n|)$ time. It can also take $O(|n|^2)$ on Korchiff graph because the graph becomes unbalanced.



Shortest paths from A

A : A = 0
A : B = 1
A : C = 3
A : D = 4

```
SpKorchiff(G, S, R)
for v in V:
    d[v] = 0
    d[s] = 0
Q (queue)
    Q_push(S)
```

```
while Q not empty
    u = Q.pop(0)
    for v in G(u)
    // min distance
    if d(u) + w(u,v) < d(v)
    d(v) = d[u] + w(u,v)
```

if v not visited
    Q.push(v)
return d
We assume all nodes are visited

(Q6)

We will use Dikjstra's Algo that takes $O(|V| + |E| \lg |V|)$ time

```
Dijkstra(Graph, start)
    pq
    distance(n) = [∞, ∞, ∞ ... ]
    distance[start] = 0
    parent[start] = 0
    pq.push(0, start)
     while pq is not empty
      curr_distance, curr_node = pq.pop()

    if  curr_distance > dist[curr_node]
          //do nothing

    for neighbor, weight in graph[curr_node]
    dist = curr_distance + weight
    if distance < distance[neighbour]
    distance[neighbor] = curr_distance
    parent[neighbor] = curr_node
```

```
        pq.push (distance, neighbor)
return parent, distance

construct_path (parent, t)
    path = []
    while t is not null.
    path.insert (0, t)
        t = parent [t]
    return path.
```