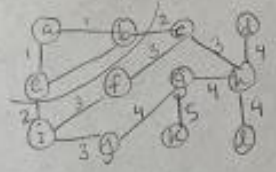
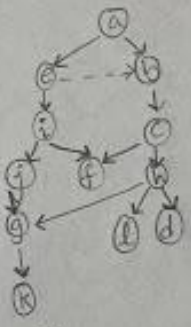
ComS311 Final Exam
Neha Maddali - 110122037 - nmaddali@iastate.edu
Recitation Section 3

Problem 1a (i)

u	a	b	c	d	e	f	g	h	i	j	k	l
u.dist	0	1	2	4	1	3	4	3	2	3	5	4



Problem 1a (ii)



Problem 1b (i)

a b e c f d

Problem 1b (ii)

a b e f c d

Problem 1b (iii)

a e b f c d

Problem 1c (i)

True

Problem 1c (ii)

False

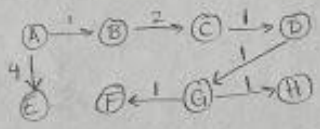
i.e there exists a minimum spanning tree not containing e

Neha Maddali
Recitation Section 3

Problem 2a

	A.d	A.p	B.d	B.p	C.d	C.p	D.d	D.p	E.d	E.p	F.d	F.p	G.d	G.p	H.d	H.p
	0	Nil	∞	Nil	∞	Nil	∞	Nil	∞	Nil	∞	Nil	∞	Nil	∞	Nil
1	0		1	A	∞	Nil	∞	Nil	4	A	8	A	∞	Nil	∞	Nil
2	0				3	B	∞	Nil			7	B	7	B	∞	Nil
3	0						4	C			8	G	5	C	8	G
4	0										6	G	5	D	16	G
5	0															
6	0															
7	0															
8	0															

Problem 2b



Problem 3a

```
foreach vertex v in G.adj(u)
    foreach vertex w in G.adj(v)
        edg(u,w) ∈ E
        u.starred = = true
        u.numStarredNbrs ++
return u.numStarredNbrs
```

Problem 3b

Iterate through all the vertices in the graph. For each of the vertices, iterate through their graph adjacency list. While iterating through the graph adjacency list of a vertex u, add the neighbor to the new adjacency list of u and u to the new adjacency list of the neighbor, if the neighbor is not vertex u and if the neighbor is not already present in the new adjacency list of u

roblem 3c

because of the nested for loop... O(V+E) is runtime

Neha Maddali
Recitation 3

Problem 4a

```
NumPaths(G,s,t)
    initialize p[t] = 1
    and for all other vertices v, p[t] = 0
    TopologicalOrdering(G) to get list L of vertices in topological order
    for i = n to 1:
        let v be the i^th element of list L
        for every neighbor u of v
            p[v] = p[v] + p[u]
    return(p[s])
```

Problem 4b

This will give the # of paths from s to t. The num of paths are denoted by p(v). For any vertex u in the DAG G, let $v_1, v_2 \dots v_k$ denote all vertices on the out-going edges of u. If we compute p values of all vertices in reverse topological order, then we are sure that for any vertex u and its neighbors $v_1 \dots v_k$, the p values of $v_1 \dots v_k$ is already known when we get to u as per the reverse topological ordering

Problem 4c

Finding topological ordering is $O(V+E)$ time. After this, this algo goes over the vertices and its neighbors. So overall running time is $O(V+E)$

Problem 5a

opt(i, 0) → value = 0

Problem 5b

opt(0, j) → value = 0

Problem 5c

$$opt(i,j) = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ \max(opt(X_{i-1}, Y_{j-1}) + 1, opt(X_i, Y_j)) & \text{if } i = j \\ \max(opt(i, j-1), opt(i-1, j)) & \text{if } i \neq j \end{cases}$$

Neha Maddali
Recitation Section 3

Problem 5d

```
MCS(x,y)
  //input: binary strings x = x1,x2...xn and y = y1,y2...ym
  //output: max weight common subsequence
  for i=0 to n
  |  M[i,0]=0
  for j=0 to m
  |  M[0,j]=0
  for i=1 to n
  |  for j=1 to m
  |  |  if xi == yj
  |  |  |  M[i,j] = M[i-1,j-1]+1
  |  |  |  if character of M[i,j] == 0
  |  |  |  |  weight = +1
  |  |  |  else
  |  |  |  |  weight = +2
  |  |  else
  |  |  |  M[i,j] = max{M[i-1,j], M[i,j-1]}
  return M[n,m]
```

Problem 5c

$O(mn)$ just like when finding LCS

Problem 6a

Certificate: G contains simple cycle w/ k or more nodes (polynomial size)
given a simple path, we can check that it is at least length k by computing the sum of lengths of all edges in it

Problem 6b

verifier: set all edges' lengths equal to one and set $k = |V| - 1$.
check all pairs of vertices in the graph (s,t), $s != t$ and if there is at least one pair return true
otherwise return false, checking that there is a path of length $|V|-1$