Homework – 3

(Solution - Set)

Homework for Notes 14 - 15

3.72. Adjacency Matrix:

I/P pairs :- 0 - 2, 1 - 4, 2 - 5, 3 - 6, 0 - 4, 6 - 0, & 1 - 3

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	1	0	1	0	1	0	1
1	0	1	0	1	1	0	0
2	1	0	1	0	0	1	0
3	0	1	0	1	0	0	1
4	1	1	0	0	1	0	0
5	0	0	1	0	0	1	0
6	1	0	0	1	0	0	1

3.73. Adjacency List:

0	->	2	->	4	->	6
1	->	4	->	3		
2	->	0	->	5		
3	->	6	->	1		
4	->	1	->	0		
5	->	2				
6	->	3	->	0		

3.74. Directed Graph:

Adjacency Matrix:

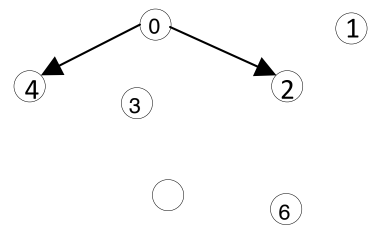
	0	**1**	**2**	**3**	**4**	**5**	**6**
0	1	0	1	0	1	0	0
1	0	1	0	1	1	0	0
2	0	0	1	0	0	1	0
3	0	0	0	1	0	0	1
4	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0

6	1	0	0	0	0	0	1

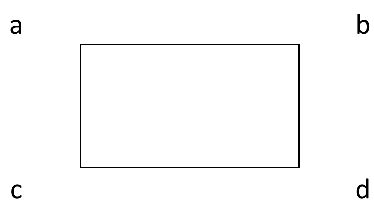
Adjacency List:

0	->	2	->	4
1	->	4	->	3
2	->	5		
3	->	6		
4				
5				
6	->	0		

Graph:



17.19.



Adjacency Matrix

->

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

If the graph is undirected, then the adjacency matrix of the 2nd graph is same as the adjacency matrix of 1st graph.

For a directed graph, Adj (Graph (G')) = Transpose of Adj (Graph (G))

17.25.

```
P = l + 1 ,  q = v

for (i = 1;   v-1;  i++)

{

        if (j = k) then

        {       p = k + 1;
```

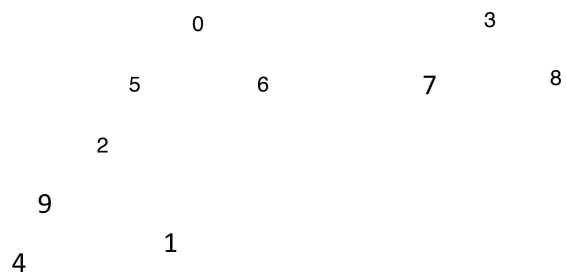
```
            q = v ;
        } end if
        for (j = p;   q;  j++)
        {       if adj[ i , j ]  = 1 then
                continue;
                else  {
                q =  j  -1 ;
                k = j;
                }
        }
    }
```

18.50. **3 – 7 , 1 – 4 , 7 – 8 , 0 – 5 , 5 – 2 , 3 – 8 , 2 – 9 ,**

0 – 6 , 4 – 9 , 2 – 6 , 6 - 4

Adjacency List:

0	->	**5**	->	**6**		
1	->	**4**				
2	->	**5**	->	**9**	->	**6**
3	->	**7**	->	**8**		
4	->	**9**	->	**1**	->	**6**
5	->	**0**	->	**2**		
6	->	**0**	->	**4**	->	2
7	->	**8**	->	**3**		
8	->	**7**	->	**3**		
9	->	**4**	->	**2**		



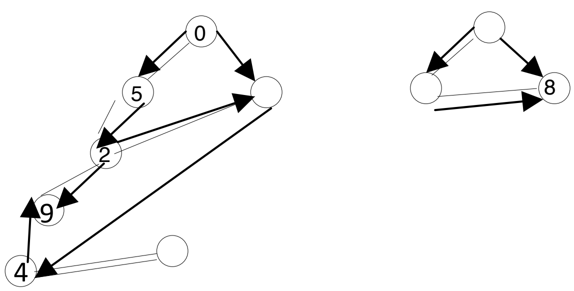
BFS Traversal Results:

0 – 5 , 0 – 6 , 5 – 2 , 6 – 4 ,
2 – 9 , 4 – 1 , 3 – 7 , 3 – 8

Answer.

<u>19.30.</u> Adjacency List of Digraph

0	->	5	->	6
1	->	4		
2	->	6	->	9
3	->	7	->	8
4	->	9		
5	->	2		
6	->	4		
7	->	8		
8	->			
9	->			



BFS Traversal Results:

1. **0 – 5 , 5 – 2 , 2 – 9 , 2 – 6 , 6 – 4**
2. **1 – 4**
3. **3 – 7 , 7 - 8**

Answer.

<u>18.9</u>

0	->	5	->	6		
1	->	4				
2	->	5	->	9	->	6
3	->	7	->	8		
4	->	1	->	9	->	6
5	->	2	->	0		
6	->	0	->	2	->	4

7	->	3	->	8
8	->	3	->	7
9	->	4	->	2

1st Tree

0-0
0-5
5-2
2-5
2-9
9-4
4-1
1-4
4 - 9
4 - 6
6 – 0
6 – 2
6 – 6

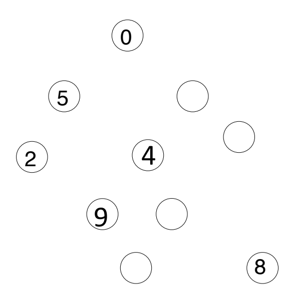
2nd Tree

3-3
3-7
7-8
8-7
7-3

0	1	2	3	4	5	6	7	8	9
0	*	*	*	*	*	*	*	*	*
0	*	*	*	*	1	*	*	*	*
0	*	2	*	*	1	*	*	*	*
0	*	2	*	*	1	*	*	*	3
0	*	2	*	4	1	*	*	*	3
0	5	2	*	4	1	*	*	*	3
0	5	2	*	4	1	6	*	*	3
0	5	2	7	4	1	6	*	*	3
0	5	2	7	4	1	6	8	*	3
0	5	2	7	4	1	6	8	9	3

19.95

0	->	5	->	6		
1	->	4				
2	->	9	->	6	->	3
3	->	7	->	8		
4	->	9	->	3		
5	->	2				
6	->	4				
7	->	8				



	8		
	9		

DFS Forest:

0 – 5 , 5 – 2 , 0 – 6 , 6 – 4 , 4 – 9 , 4 – 3 , 3 – 7 , 1 – 4

Topological Sort: Ordering:

0 1 5 6 4 2 9 3 7 8 (Answer)

<u>19.96</u>

Preorder numbering can be used to do a topological sort. Since preorder numbering simulates the behavior of a DFS in a graph which in turn can be considered as topological sort for that component.

For example:

0

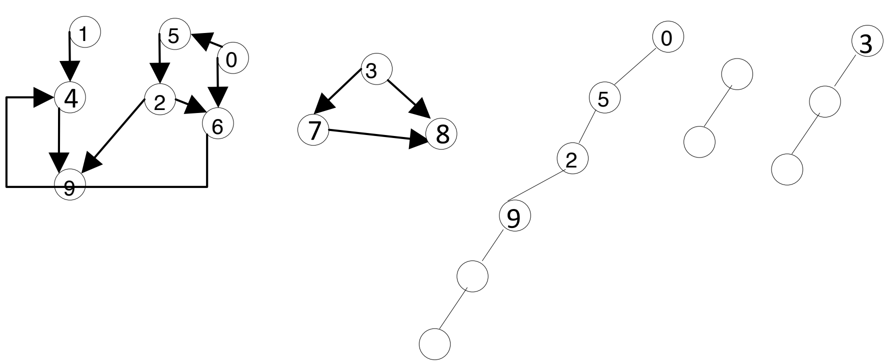
1

2

DFS in this graph produces 0 -> 1 -> 2 which is the topological sorting and inorder number of the vertices **(Answer)**

<u>19.124</u>

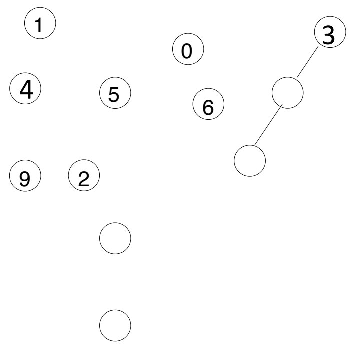
3 – 7 , 1 – 4 , 7 – 8 , 0 – 5 , 5 – 2 , 3 – 8 , 2 – 9 , 0 – 6 , 4 – 9 , 2 – 6 , 6 – 4



0 1 2 3 4 5 6 7 8 9

Postorder numbering										
	4	6	9	2	5	0	1	8	7	3

	0	1	2	3	4	5	6	7	8	9
id	1	2	1	0	2	1	1	0	0	2



20.1

Suppose the weight set of the graph = { w_1, w_2, ... , w_n } in ascending order. where $w_1 \le w_2 \dots \le w_n$.When we add (multiply) a factor say T with all of this,

They scale by ,

$w_1 + t_1$ ($*t_1$), $w_2 + t_2$($*t_2$) $\le$ .. . . and so on

Hence the proof.

But if the factor is –ve, say t_1

Then if we multiply ($-t_1$)

i.e, $w_1 * (-t_1)$, $w_2 *(-t_2)$, ... and so on

the ordering property reverses

i.e if previously $w_1 \le w_2 \le \dots w_n$

now $w_1 * (-t_1) \ge w_2 *(-t_2)$ ---- -- - and so on because of the multiplication by a –ve values **(proved)**

20.4 **Maximum ST:-**

Perform any spanning tree algo. M (say Prim's and Kruskal) by examining the edge in order of non-increasing weights (largest first, smallest last). If two or more edge have the same weights, order them arbitrarily **(Answer)**

20.5

Has a unique MST if the edge weights are distinct.

The unique MST will be produced by using Kruskal's algorithm. **(Answer)**

20.27