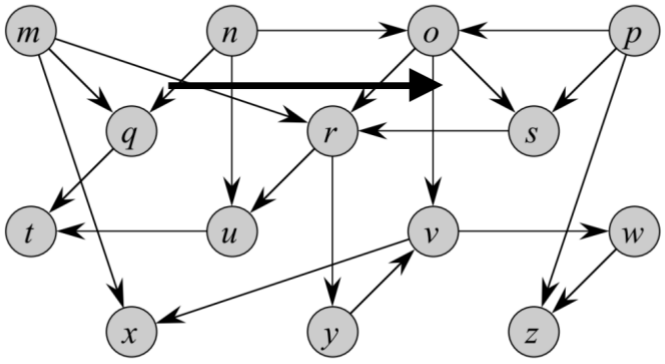
1. Run TOPOLOGICAL-SORT on the graph below. Show the:
   (a) The discovery time and finish time of each node.
   (b) The returned linked-list.
   Assume that **for** loop of lines 5—7 of the DFS procedure (page 604 in CLRS) considers the vertices in alphabetical order, and assume the adjacency list is ordered alphabetically.

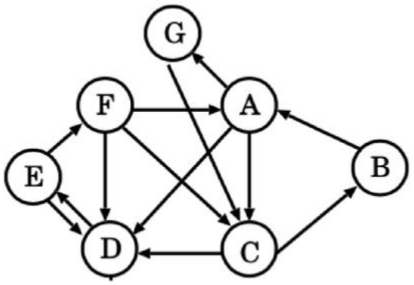


A)

The discovery time and running time for these nodes are :
M(1,20)
N(21,26)
O(22,25)
P: (27, 28)
Q(2,5)
R(6,19)
S(23,24)
T(3,4)
U(7,8)
V(10,17)
W(11,14)
X(15,16)
Y(9,18)
Z(12,13)

B) the returned linked_list after topological sort would be : p, n, o, s, m, r, y, v, x, w, z, u, q, t

2. Run the procedure STRONGLY-CONNECTED-COMPONENTS on the graph below, show the finishing times for each node after running DFS in line 1 and DFS forest produced by line 3.



Assume that **for** loop of lines 5—7 of the DFS procedure (page 604 in CLRS) considers the vertices in alphabetical order, and assume the adjacency list is ordered alphabetically.

Running time and discovery time would be:
A(1,14)
B(3,4)
C(2,11)
D(5,19)
E(6,9)
F(7,8)
G(12,13)

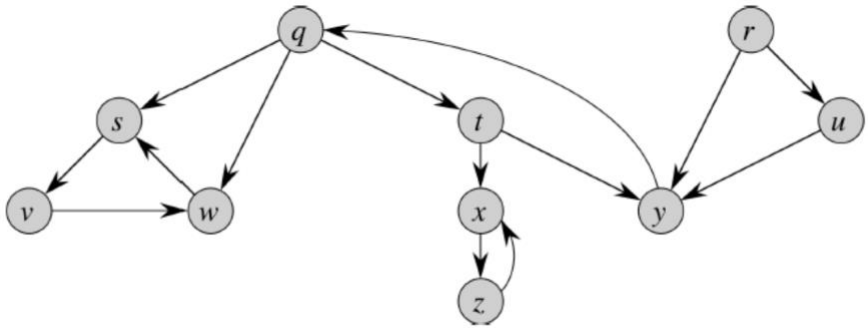
3. Run the procedure STRONGLY-CONNECTED-COMPONENTS(page 617 in CLRS) on the graph below. Show the:

   (a) The discovery time and finish time for each node after running DFS in line 1

   (b) The DFS forest produced by line 3

   (c) The component DAG

   Assume that **for** loop of lines 5—7 of the DFS procedure (page 604 in CLRS) considers the vertices in alphabetical order, and assume the adjacency list is ordered alphabetically.

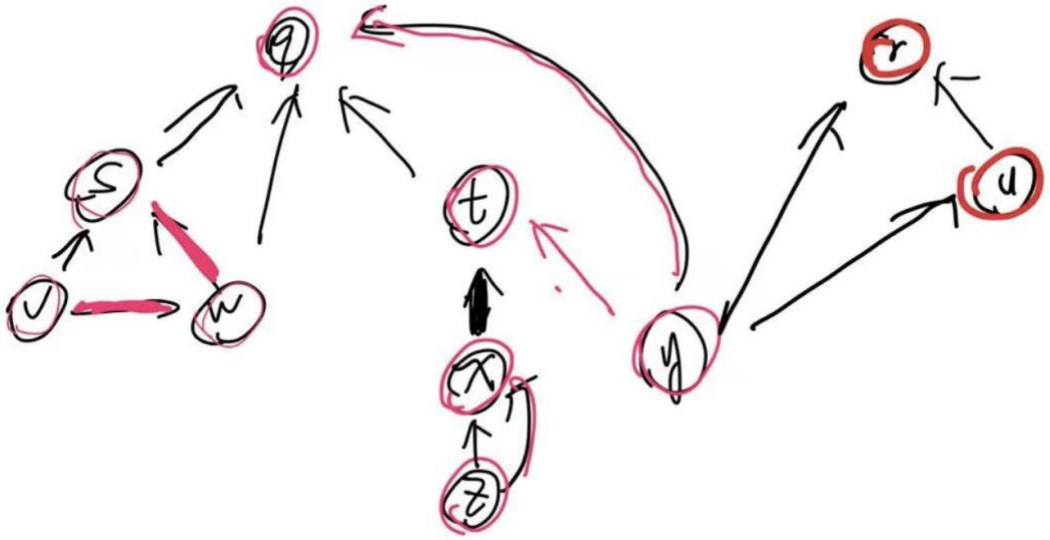


Answers:

a)

	q	r	s	t	u	v	w	x	y	z
discovery	1	17	2	8	18	3	4	9	13	10
finish	16	20	7	15	19	6	5	12	14	11

b)



The red mark are the DFS tree, there are five different tree in the DFS forest.

c):

components: {r} -> {u} -> {q, y, t} -> {x, z} -> {s, w, v}

4. Given an $M \times N$ matrix D and two coordinates (a, b) and (c, d) which represent top-left and bottom-right coordinates of a sub-matrix of the given matrix, propose a **dynamic-programming approach** to calculate the sum of all elements in the sub-matrix. What is the time complexity of your solution?

0	-2	-7	0
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

Example of a sub-matrix where $(a, b) = (1, 0)$ and $(c, d) = (3, 1)$

In this solution, we could using the presume algorithm in the dynamic programming.
Here I set (a, b) = (row1, col1)

And (C,d) = (row2, col2)

Code in python:

```
presum = [[0 for _ in range(len(matrix[0])+ 1)] for i in range(len(matrix) + 1)]

    if len(matrix[0]) == 0 or len(matrix) == 0:
        return

    for i in range(len(matrix)):
        for j in range(len(matrix[0])):

            presum[i + 1][j + 1] = presum[i+1][j] + presum[i][j+1] + matrix[i][j] - presum[i][j]


result = presum[row2 + 1][col2 + 1] - presum[row1][col2 + 1] - presum[row2 + 1][col1] +
presum[row1][col1]

return  result
```

The time and space complexity would be both O(1)