Student ID: N2537
Name: Haoze He

EL9343 Homework 9

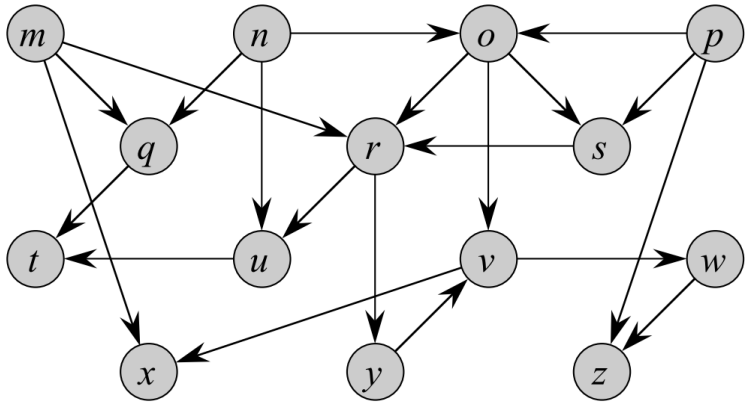
(Due Nov 22th, 2021)

No late submission accepted

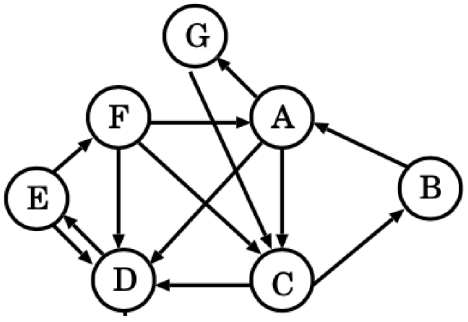
All problem/exercise numbers are for the third edition of CLRS textbook

---

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.



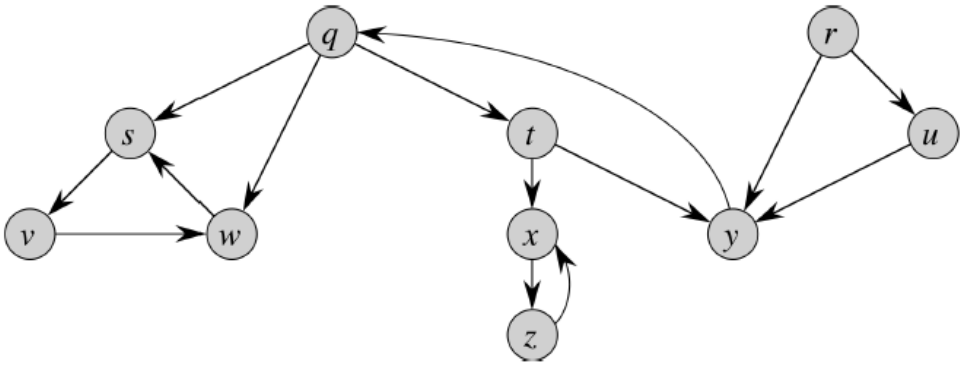
2. Run the procedure STRONGLY-CONNECTED-COMPONENTS on the graph below, show the finishing times for each node after running DFS in line 1 and DFS forest produced by line 3.



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.

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.



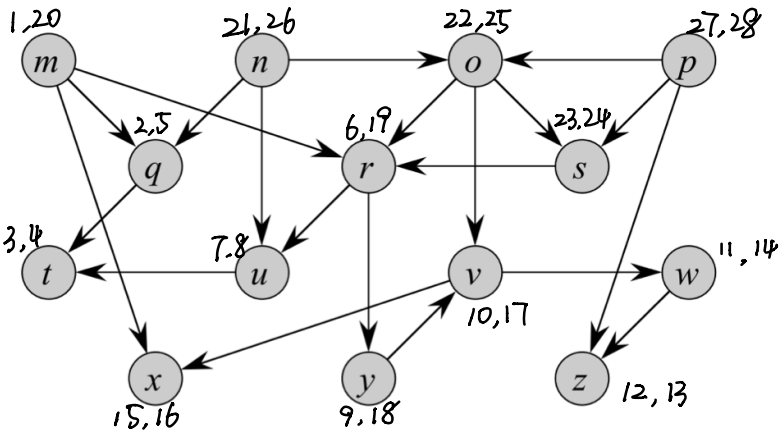
4. Given an $M \times N$ matrix D and two coordinates (a, b) and (c, d) which represent top-left and bottom-right coordinates of a sub-matrix of the given matrix, propose a **dynamic-programming approach** to calculate the sum of all elements in the sub-matrix. What is the time complexity of your solution?

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

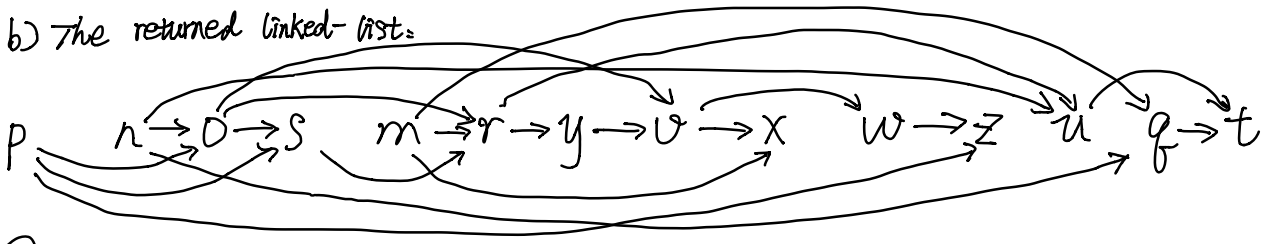
Example of a sub-matrix where $(a, b) = (1, 0)$ and $(c, d) = (3, 1)$

Q1:

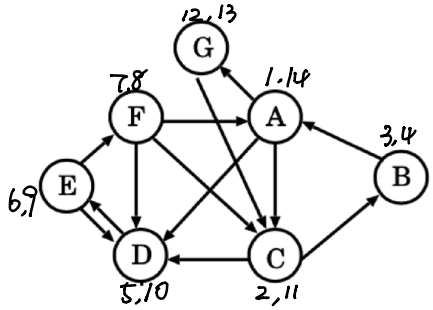
a)



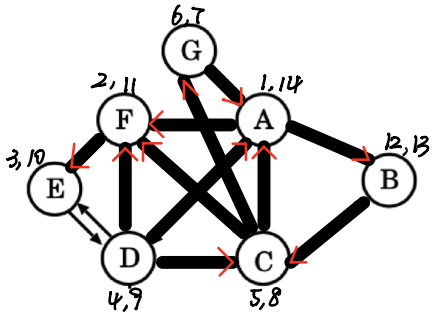
b) The returned linked-list:



Q2 Original Graph G:



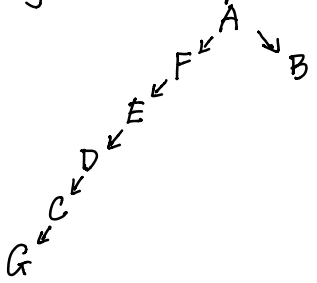
Reversed Graph G^T:



Finish Time of original Graph is: (Rank from later to earlier)

A(14), G(13), C(11), D(10), E(9), F(8), B(4)

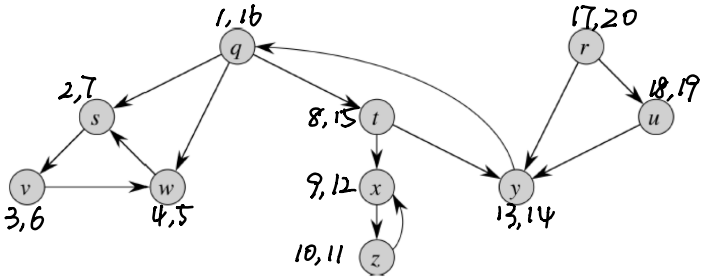
Strong Connected Component:



The whole graph with all vertices is a Strong-Connected-Component. {A, B, C, D, E, F, G}

Q3.

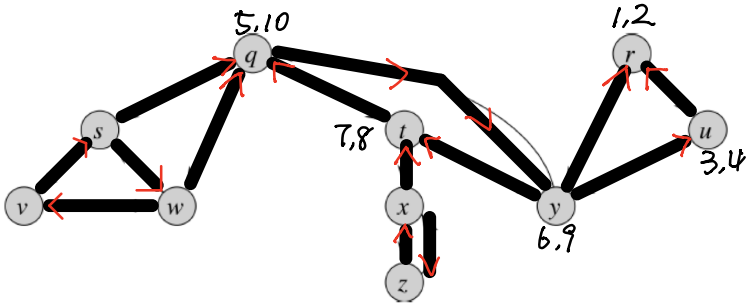
a)



Finish Time of original Graph is: (Rank from later to edier)

r(20), u(19), q(16), y(14), t(15), x(12), z(11), s(7), v(6), w(5)

b)



Strong Connected Component

{r} {u} {q, y, t} {x, z} {s, w, v}

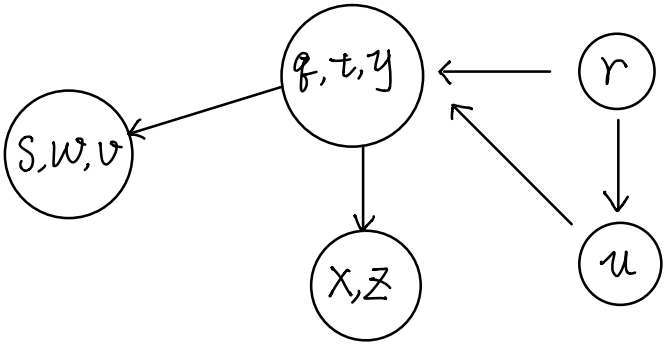
r u

q → y → t

x → z

s → w → v

c)



Q4

```python
# Call this function to get the sum of submatrix
def findSubmatrixSum(mat, p, q, r, s):

    if (not mat or not len(mat)):
        return 0

    # Call the preprocess function to store
    # All the necessary information
    mat = preprocess(mat)
    totalsSum = mat[r][s]
    if (q - 1 >= 0):
        totalsSum -= mat[r][q - 1]
    if (p - 1 >= 0):
        totalsSum -= mat[p - 1][s]
    if (p - 1 >= 0 and q - 1 >= 0):
        totalsSum += mat[p - 1][q - 1]
    return totalsSum

def preprocess(mat):
    # `M × N` matrix
    (M, N) = (len(mat), len(mat[0]))

    # Store useful information which will be used again
    # information: Sum of elements in the matrix from (0, 0) to (i, j)

    store = [[0 for x in range(len(mat[0]))] for y in range(len(mat))]
    store[0][0] = mat[0][0]

    # preprocess the first row
    for j in range(1, len(mat[0])):
        store[0][j] = mat[0][j] + store[0][j - 1]

    # preprocess the first column
    for i in range(1, len(mat)):
        store[i][0] = mat[i][0] + store[i - 1][0]

    # preprocess the rest of the matrix
    for i in range(1, len(mat)):
        for j in range(1, len(mat[0])):
            store[i][j] = mat[i][j] + store[i - 1][j] + store[i][j - 1] - store[i - 1][j - 1]

    return store
```

The Time Complexity is $O(MN)$ [input matrix is $m \times n$]