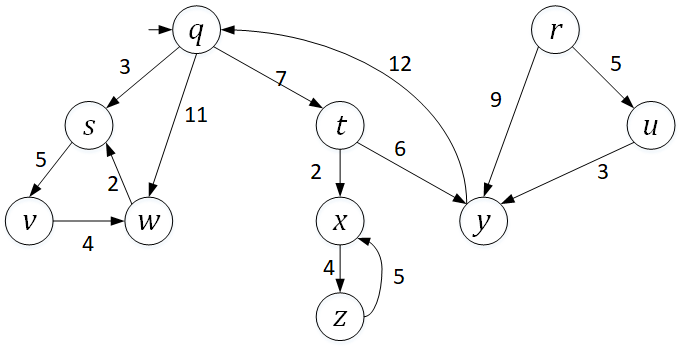
CSci 242: Algorithms and Data Structures  **Spring, 2020** Instructor: Dr. M..E. Kim Date: April 15 (Wed), 2020

**Due: by the end of day, May 1st (Fri.)**

**Home Assignment 8: Graph Algorithms (256/250 + 50 optional)**

**Q1 – Q4.** For a given graph *G1=(V, E)* in the figure, perform the given tasks.

In DFS and BFS, a weight of edge is not considered and a priority for selection is given to the vertex whose alphabetic order is the lower: e.g.) s < w < y in DFS or in BFS from the starting vertex *q.*



**Q1. [20/25] Breadth First Search (BFS)**

Traverse the graph *G1* from a start vertex *q* by *breadth first search (BFS)*.

1.1) [8/10] List the vertices in the order of traversal.

Q S T W V X Y Z # r and u are missing.

1.2) [10/10] Give a list of the discovery edges in your DFS tree

(Q, S)

(Q, T)

(Q, W)

(S, V)

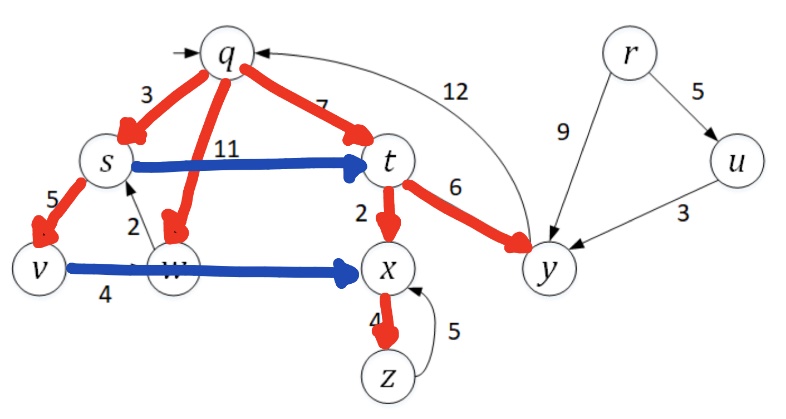
(T, X)

(T, Y)

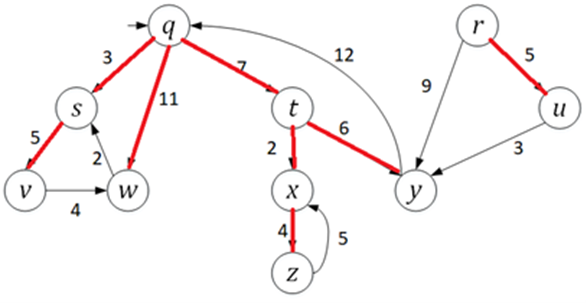
(X, Z)

(R, U)

1.3) [2/5] Mark the DFS tree with the discovery edges in red and the cross edges in blue, respectively in the given graph.



# correct answer is as follows:



**Q2. [33/35] Depth First Search (DFS)**

Traverse the graph *G1* from a start vertex *q* by *depth first search (DFS)*.

2.1) [10/10] List the vertices in the order of traversal with their start time & finish time.

|  |  |  |
| --- | --- | --- |
| Vertex | Start | Finish |
| Q | 1 | 16 |
| S | 2 | 7 |
| V | 3 | 6 |
| W | 4 | 5 |
| T | 8 | 15 |
| X | 9 | 12 |
| Z | 10 | 11 |
| Y | 13 | 14 |
| R | 17 | 20 |
| U | 18 | 19 |

2.2) [9/10] Give a list of the discovery edges in your DFS tree.

(Q, S)

(S, V)

(V, W)

(Q, T)

(T, X)

(X, Z)

(T, Y) # ru is missing.

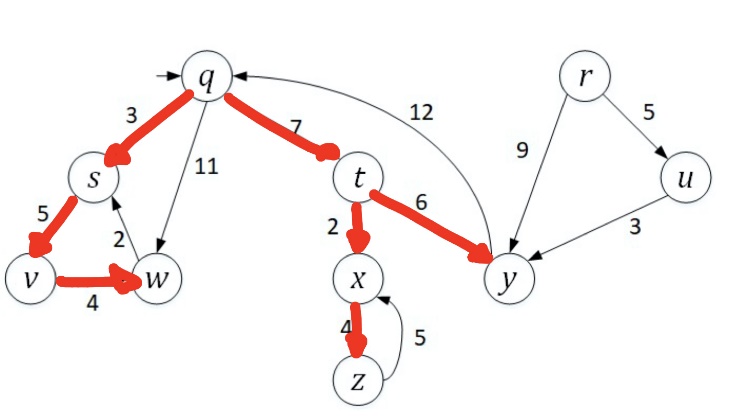
2.3) [10/10] Give a list of back edge, forward edge and cross edge, respectively, if there were any.

Back Edge : (W,S) , (Z, X) , (Y, Q)

Forward Edge : ((Q, W)

Cross Edge : (R, Y), (U, Y)

2.4) [4/5] Mark the DFS tree with the discovery edges in the given graph.



**# the edge ru needs to be included as well.**

**Q3.** **[33/40]** **Strongly Connected Component (SCC)**

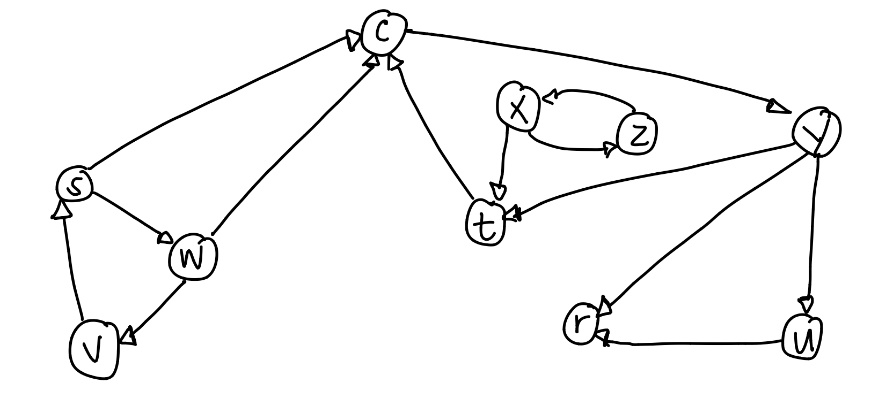
From the Depth-First Search(DFS) in Q2, showing the finishing times of the vertices,

3.1) [8/10] Arrange the vertices in decreasing order of its finishing time,

W V S Z X Y T Q U R

# r, u, q, t, y, x, z, s, v, w is the correct order. You have mentioned according to the increasing order (opposite).

3.2) [5/5] Draw the transposed graph *G1T* of G1.



3.3) [10/10] Perform DFS on *G1T*. Show the DFS tree(s) in the G1T in the map.

R : 1 , 2

U : 3 , 4

Q ; 5, 10

Y : 6, 9

T : 7, 8

X : 11 , 14

Z : 12 , 13

S : 15 , 20

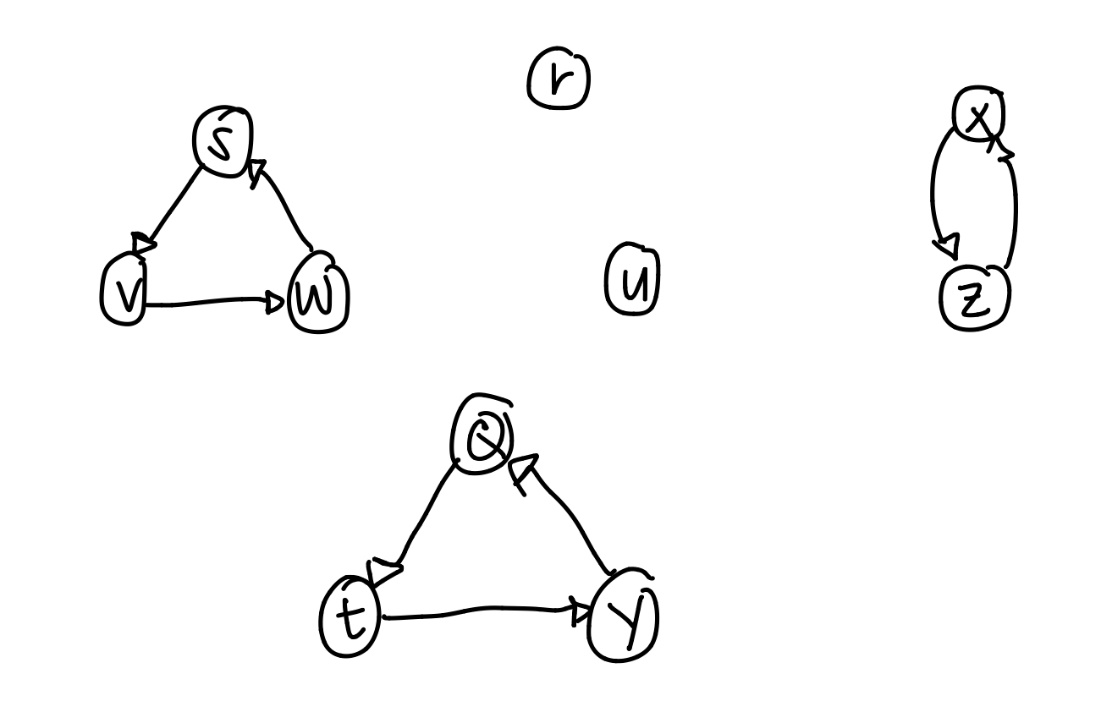
W : 16, 19

V : 17 , 18

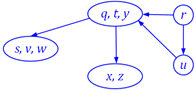
3.4) [10/10] Show each *SCC* of *G1*.: e.g.) SCC1 = {*q, s, t*}, SCC2 = {*x, y, z*}

SCCS : {R}, {U}, {Q, T, Y}, {X, Z}, {S, W, V}

3.5) [0/5] draw the acyclic *component graph* *GSCC* .



# correct graph is as follows:



Assume that the loop of DFS considers vertices in alphabetical order.

**Q3B. [20/25, optional]** Implementation in Python/Java. Print the outcomes of 3.1) and 3.4) #implementation done in increasing order of finish time but in the question asked according to the decreasing order of finish time.



**Q4. [30/30] A Single-Source Shortest Path (SSSP)**

EITHER by applying *Dijkstra*’s algorithm

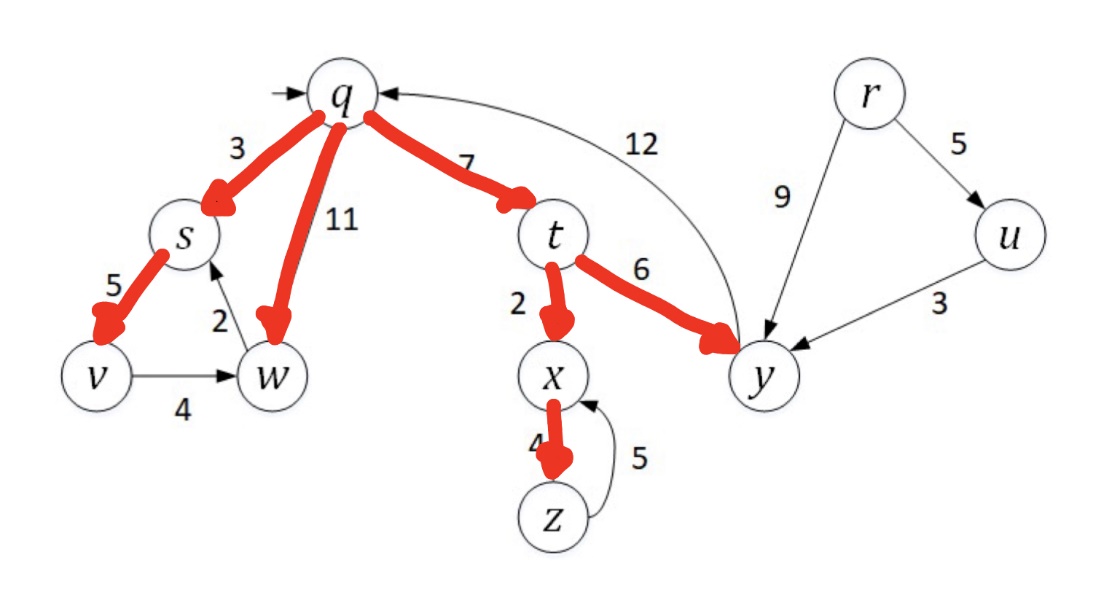
OR by applying *Bellman-Ford* algorithm to the directed graph G1,

find the shortest path from *q* to each vertex, respectively.

4.1) [20/20] List the edges in the shortest path.

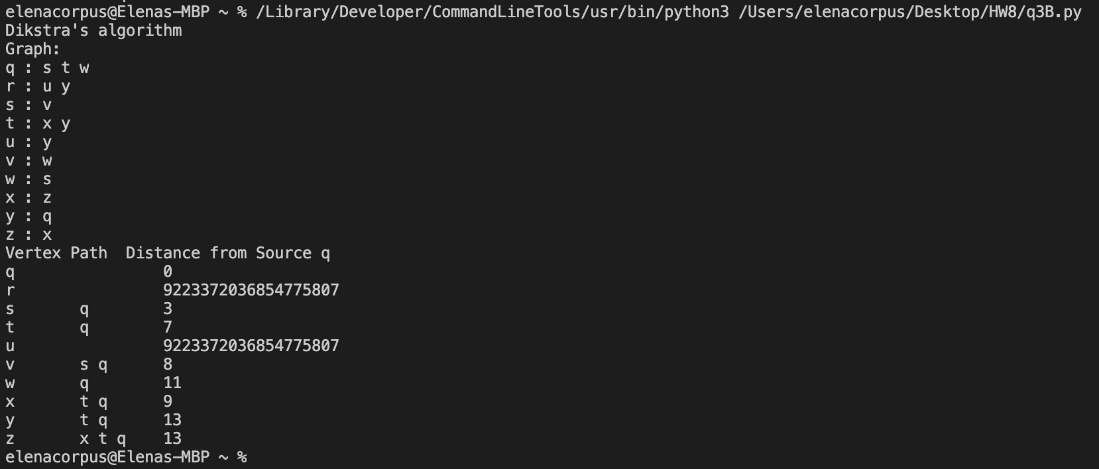
|  |  |  |
| --- | --- | --- |
| Vertex | Path | Distance |
| Q | S | 3 |
| S | V | 5 |
| Q | W | 11 |
| Q | T | 7 |
| T | X | 2 |
| X | Z | 4 |
| T | Y | 6 |

4.2\_ [10/10] Mark the shortest path in the graph.

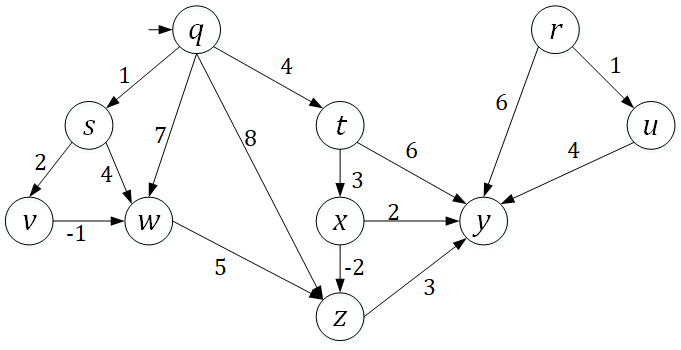


**Q4B. [25/25]** Implementation in Python/Java. Print the outcomes of 4.1).

Specify which algorithm you’ve applied.



**Q5.** In the given modified Directed Acyclic Graph (DAG) G2,

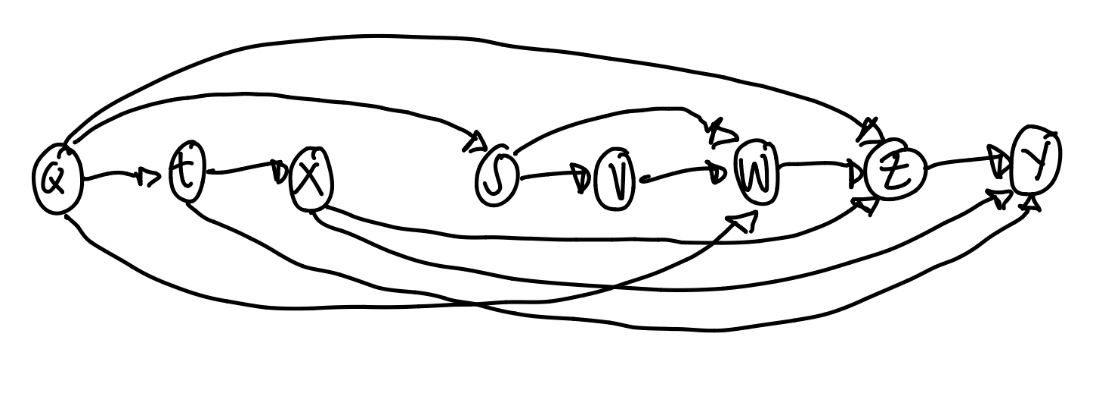


**Q5. [18/40] Single Source Shortest Path in the DAG**

5.1) [10/15] Sort the vertices in the ***topological order*** starting from *q* and give its list.

Q T X S V W Z Y # r and u are missing. They can be introduced before y.

5.2) [8/10] Redraw the graph by arranging the vertices in the sorted order.# r and u missing.

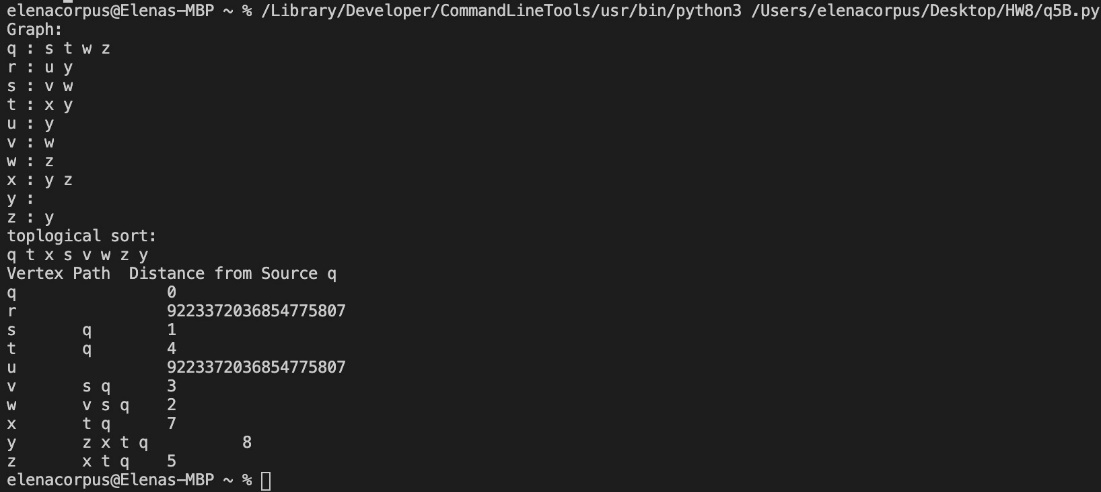


5.3) [0/15] Find the shortest path from a vertex *q* to each vertex. You have to show the proper

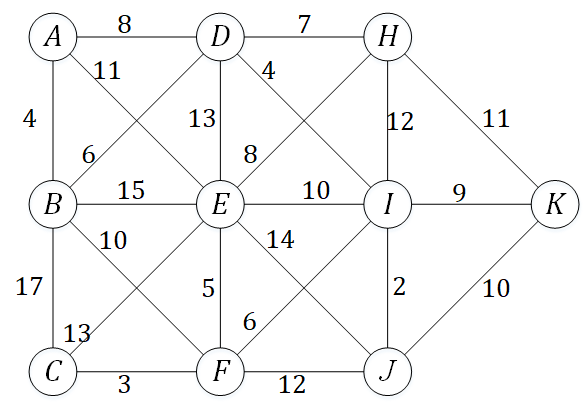
steps of edge relaxations, updating a key, D[*v*] of each vertex *v, v*  V(G2).

**Q5B. [22/25, optional]** Implementation in Python/Java. Print the outcomes of 5.1) and 5.3): the vertices in the topological order and the list of edges in the shortest path from *q*, respectively.

# r and u missing from your implementation in 5.1



**Q6.**  In the given undirected graph G3 below:



**Q6. [30/30] Minimum Spanning Tree**

EITHER by applying *Prim*’s algorithm

OR by applying Kruskal’s algorithm

find the Minimum Spanning Tree (MST) of G3.

6.1) [20/20] List the edges in the MST.

(A, B) : 4

(F, C) : 3

(B, D) : 6

(F. E) : 5

(I, F) : 6

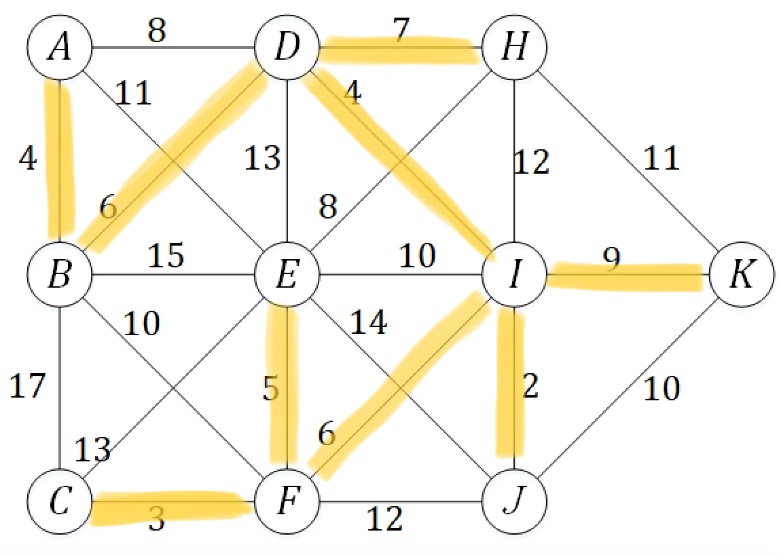
(D, H) : 7

(D, I) : 4

(I, J) : 2

(I, K) : 9

6.2) [10/10] Mark the edges of the MST in the graph.



**Q6B. [25/25]** Implementation in Python/Java. Print the outcomes of 6.1) or 6.2).

Specify which algorithm you’ve applied.

