CSci 242: Algorithms and Data Structures

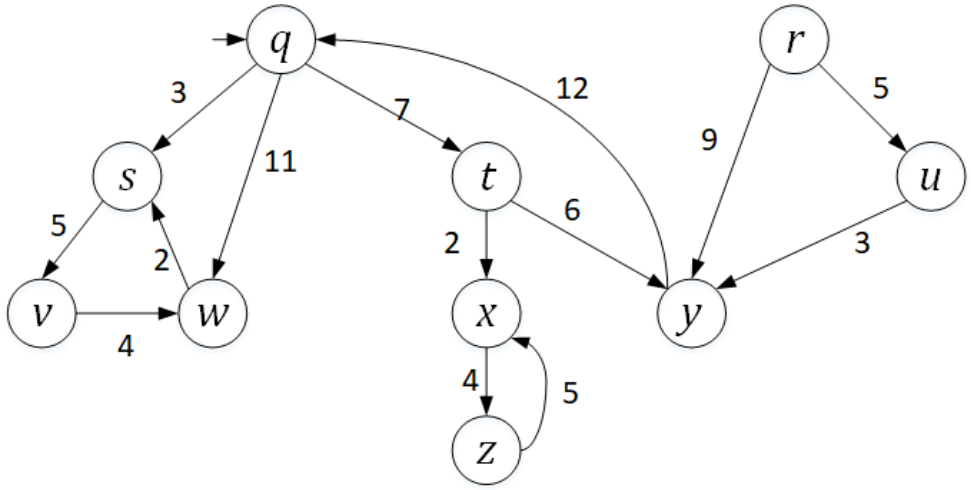
Fall, 2019 Instructor: Dr. M..E. Kim Date: November 26 (Tue.), 2019

Due: by the end of day, December 10th (Tue.)

Home Assignment 8: Graph Algorithms (200 + 100 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. [25] Breadth First Search (BFS)

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

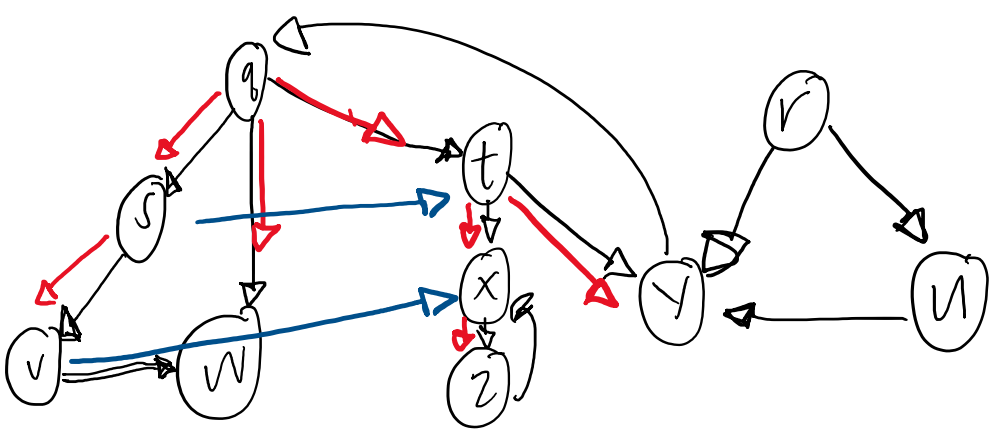
1.1) [10] List the vertices in the order of traversal

q s t w v x y z

1.2) [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

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



Q2. [35] Depth First Search (DFS)

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

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

Vertex	Start	Finish
Q	1	9
s	2	5
v	3	5
w	4	5
t	5	9
x	6	8
z	7	8
y	8	9

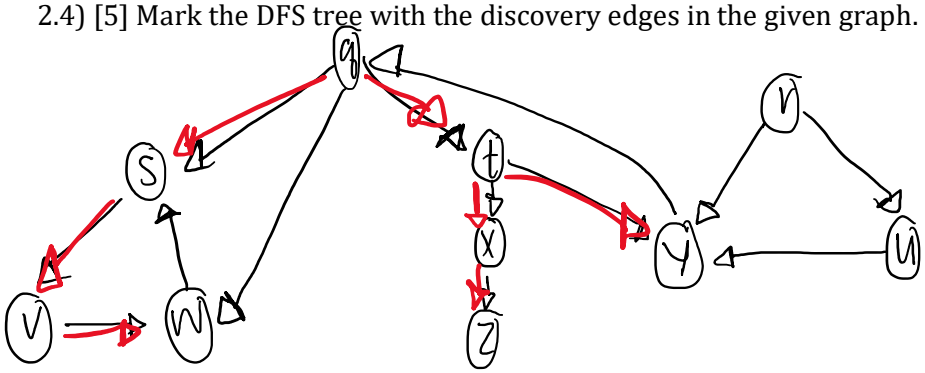
2.2) [10] Give a list of the discovery edges in your DFS tree.

Q : S
S : V
V : W
Q : T
T : X
X : Z
T : Y

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

Back edge:
W:S
Y:Q
Z: X
Forward Edge:
R:Y
R : U
U : Y

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



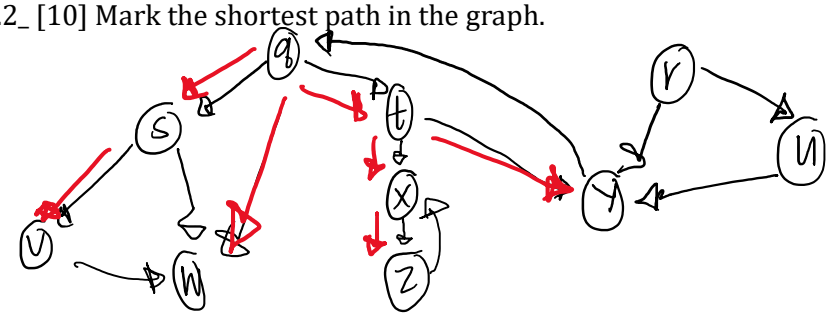
Q3. [30] A Single-Source Shortest Path (SSSP)

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.

3.1) [20] List the edges in the shortest path.

Vertex	Path	Distance from Source Q
q		0
r		INF
s	q	3
t	q	7
u		INF
v	sq	8
w	q	11
x	tq	9
y	tq	13
z	xtq	13

3.2_ [10] Mark the shortest path in the graph.



Q3B. [25, optional] Implementation in Python/Java. Print the outcomes of 3.1).
Specify which algorithm you've applied.

```
elenacorpus@Elenas-MBP ~ % /Library/Developer/CommandLineTools/usr/bin/python3 /Users/elenacorpus/Desktop/HW8/q3B.py
Dikstra's algorithm
Graph:
q : s t w
r : u y
s : v
t : x y
u : y
v : w
w : s
x : z
y : q
z : x
Vertex Path  Distance from Source q
q              0
r              9223372036854775807
s       q      3
t       q      7
u              9223372036854775807
v       s q    8
w       q      11
x       t q    9
y       t q    13
z       x t q  13
elenacorpus@Elenas-MBP ~ %
```

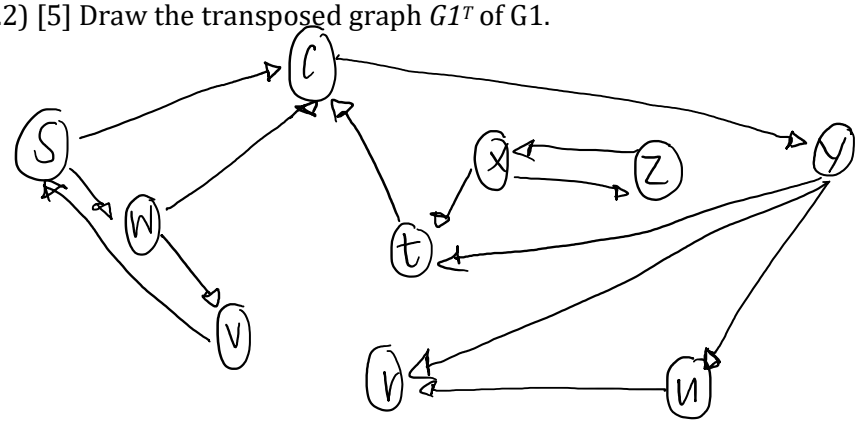
Q4. [40] Strongly Connected Component (SCC)

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

4.1) [10] Arrange the vertices in decreasing order of its finishing time,

w v s z x y t q u r

4.2) [5] Draw the transposed graph $G1^T$ of G1.



4.3) [10] Perform DFS on $G1^T$. Show the DFS tree(s) in the $G1^T$ in the map.

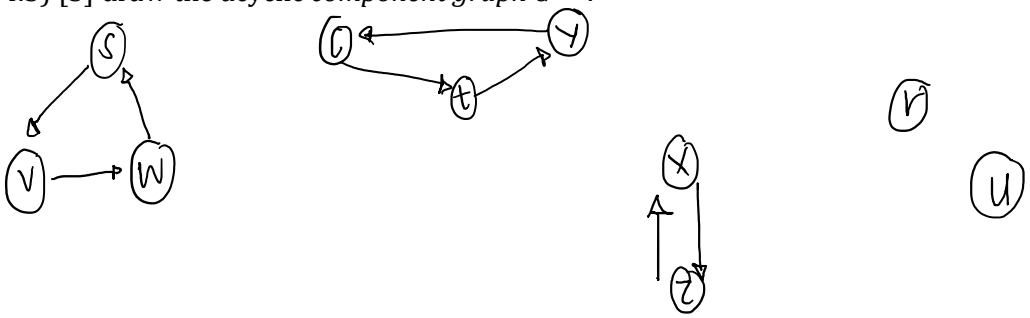
r: r

u: u
q: q y t
t:
y:
x: x z
z:
s: s w v
v:
w:

4.4) [10] Show each *SCC* of *G1*.: e.g.) SCC1 = {*q, s, t*}, SCC2 = {*x, y, z*}

q y t SCC1
x z SCC2
s w v SCC3

4.5) [5] draw the acyclic *component graph* G^{SCC} .

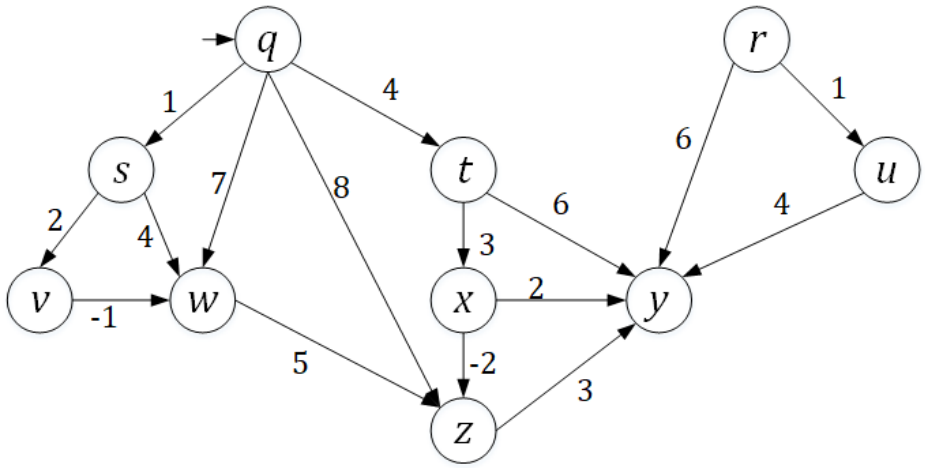


Assume that the loop of DFS considers vertices in alphabetical order.

Q4B. [25, optional] Implementation in Python/Java. Print the outcomes of 4.1) and 4.4)

```
elenacorpus@Elenas-MBP ~ % /Library/Developer/CommandLineTools/usr/bin/python3 /Users/elenacorpus/Desktop/HW8/q4B.py
SCC graph:
q : s t w
r : u y
s : v
t : x y
u : y
v : w
w : s
x : z
y : q
z : x
transpose graph
q : y
r :
s : q w
t : q
u : r
v : s
w : q v
x : t z
y : r t u
z : x
finishing time order
w v s z x y t q u r
SCC:
r: r
u: u
q: q y t
t: y: x: x z
z: s: s w v
v: w:
elenacorpus@Elenas-MBP ~ %
```

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

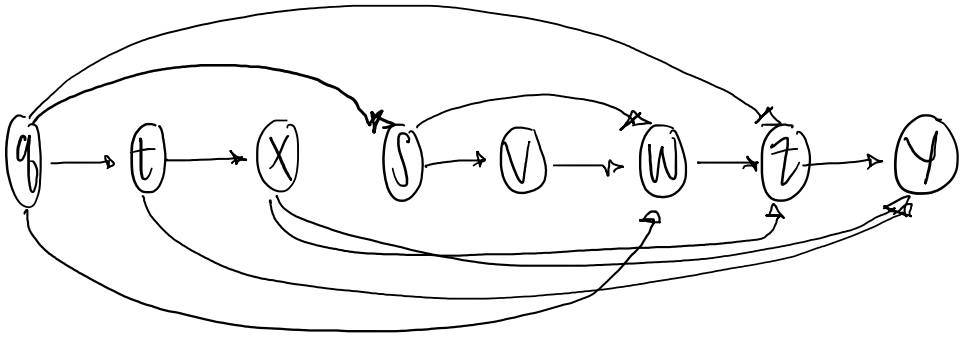


Q5. [40] Single Source Shortest Path in the DAG

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

q t x s v w z y

5.2) [10] Redraw the graph by arranging the vertices in the sorted order.



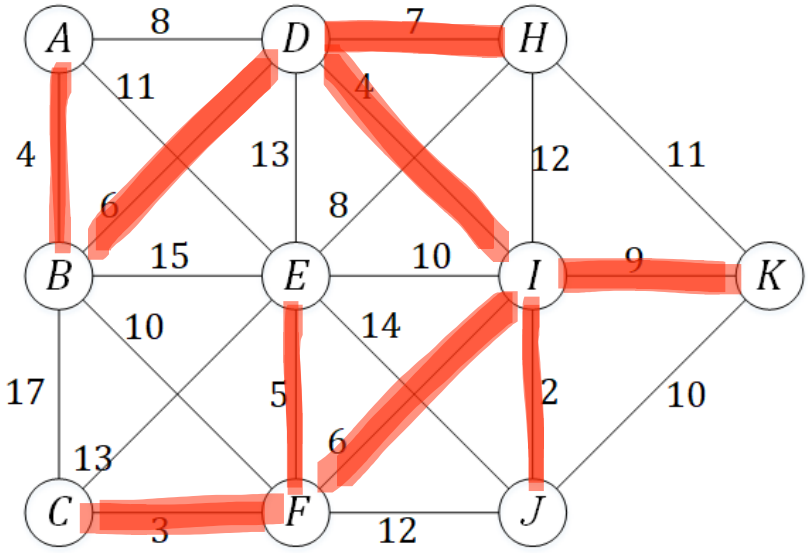
5.3) [15] Find the shortest path from a vertex q to each vertex. You have to show the proper steps of edge relaxations, updating a key, D[v] of each vertex v, $v \in$ V(G2).

Vertex	Path	Distance from the Source Q
q		0
r		INF
s	q	1
t	q	4
u		INF
v	sq	3
w	vsq	2
x	tq	7
y	zxtq	8
z	xtq	5

Q5B. [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.

```
elenacorpus@Elenas-MBP ~ % /Library/Developer/CommandLineTools/usr/bin/python3 /Users/elenacorpus/Desktop/HW8/q5B.py
Graph:
q : s t w z
r : u y
s : v w
t : x y
u : y
v : w
w : z
x : y z
y :
z : y
toplogical sort:
q t x s v w z y
Vertex Path  Distance from Source q
q               0
r               9223372036854775807
s       q       1
t       q       4
u               9223372036854775807
v       s q     3
w       v s q   2
x       t q     7
y       z x t q         8
z       x t q   5
elenacorpus@Elenas-MBP ~ %
```

Q6. In the given undirected graph G3 below:



Q6. [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] List the edges in the MST.

Edge	Weight
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] Mark the edges of the MST in the graph.

Q6B. [25, optional] Implementation in Python/Java. Print the outcomes of 6.1) or 6.2). Specify which algorithm you've applied.