MCS 401 Spring 2020
Homework 5

1. Problem 1 [Exercise 22-2-5, page 602:]

   By Theorem 22.5 from the book, for every vertex u we have $u.d = \delta(u, s)$ where δ represents the shortest path distance between s and u in the graph. The shortest path distance between vertices in any graph does not depend on the representation of the graph, it is an innate property of the graph.

   Consider Figure 22.3 from the book. From this example, it follows that the adjacency list contains the following list for the vertex w:
$$(w) - -(t) - -(x).$$
   Now assume that adjacency list is slightly different; for w we have
$$(w) - -(x) - -(t)$$
   and the rest of the adjacency list stays the same. If we do BFS with this new adjacency list, then the steps a),b) and corresponding queues Q do not change. The step c) does change and the new queue is
$$\begin{matrix} [r] & [x] & [t] \\ 1 & 2 & 2 \end{matrix}$$
   At the step d) we add r to the tree and get a queue
$$\begin{matrix} [x] & [t] & [v] \\ 2 & 2 & 2 \end{matrix}$$
   At the step e) we add x to the tree and add unexplored neighbors of x namely u, y to the queue:
$$\begin{matrix} [t] & [v] & [u] & [y] \\ 2 & 2 & 3 & 3 \end{matrix}$$
   It means that **both** u and y are be children of the vertex x in this BFS tree. In other words, BFS tree build from the changed adjacency list contains edges (x, y) and (x, u). However, the BFS tree from the adjacency list in the book **does not** contain an edge (x, u). Therefore, those two trees are different.

2. Problem 2 [Exercise 22-3-2, page 610:]

   Figure 2 shows the steps followed by DFS; this yields the following DFS forest:

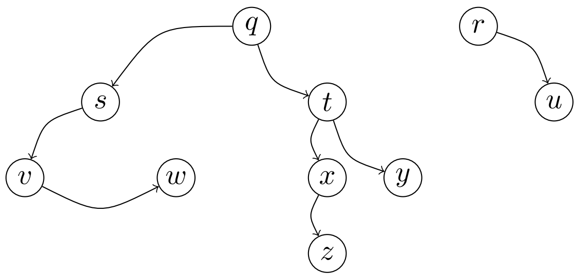


Figure 1: The final DFS forest

Thus, the edges are classified as follows:

- Tree edges – qs, qt, ru, sv, tx, ty, vw, xz
- Back edges – ws, yq, zx
- Cross edges – ry, uy
- Forward edges – qw

NOTE: the images below are missing the edge from qw.

The discovery and finishing times for each vertex are as follows:
$(q, 1, 16), (r, 17, 20), (s, 2, 7), (t, 8, 15), (u, 18, 19), (v, 3, 6), (w, 4, 5), (x, 9, 12), (y, 13, 14), (z, 10, 11)$.

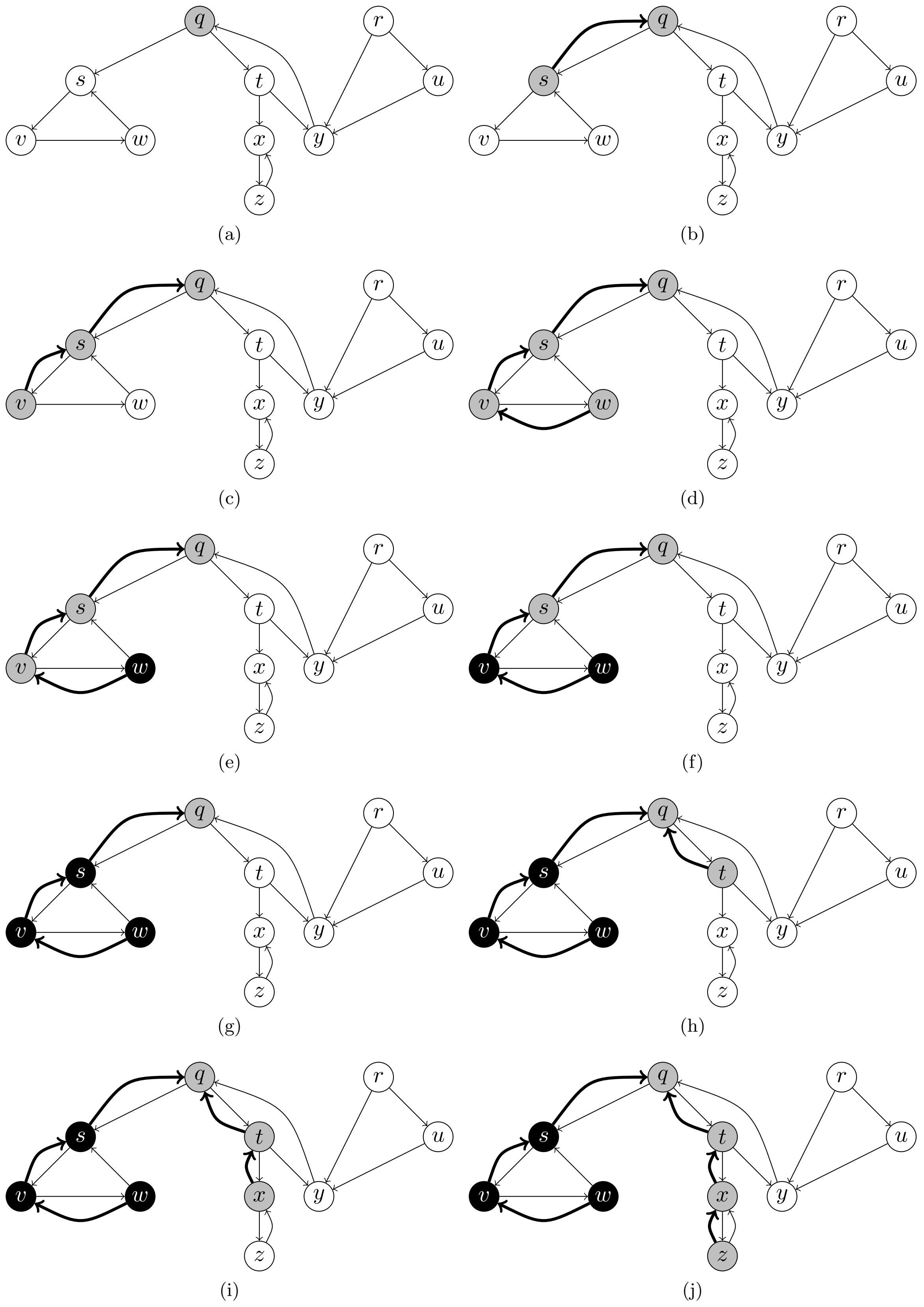
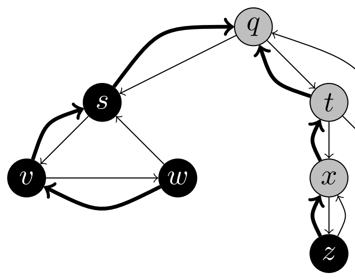
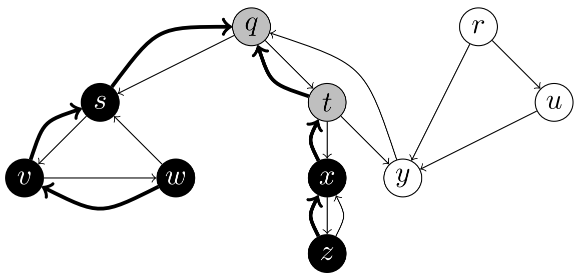
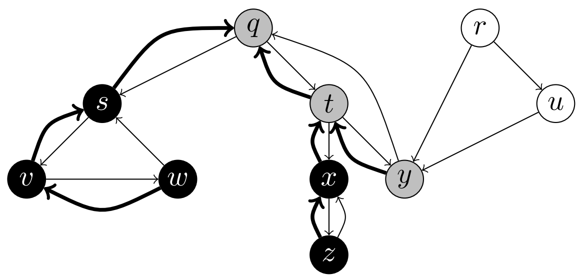
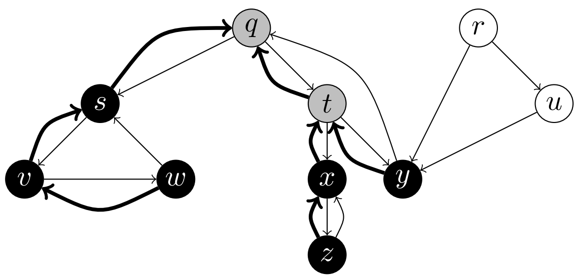


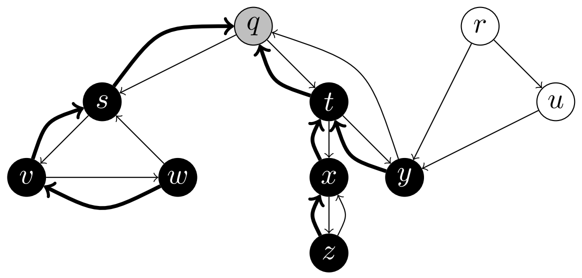
Figure 2: The steps followed by DFS. Bold edges represent the predecessor relation.

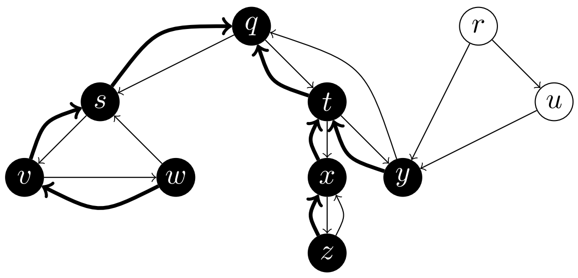

q
r
s
t
u
v
w
x
y
z
(k)

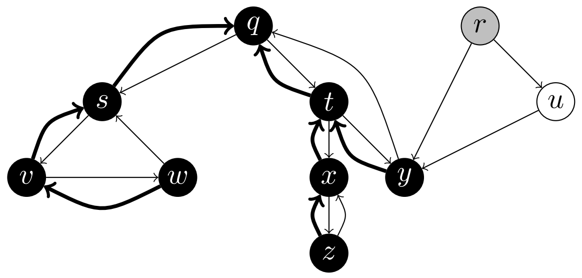

q
r
s
t
u
v
w
x
y
z
(l)

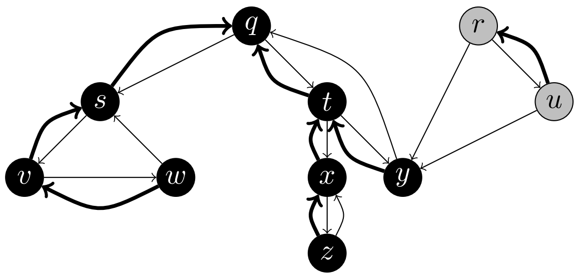

q
r
s
t
u
v
w
x
y
z
(m)

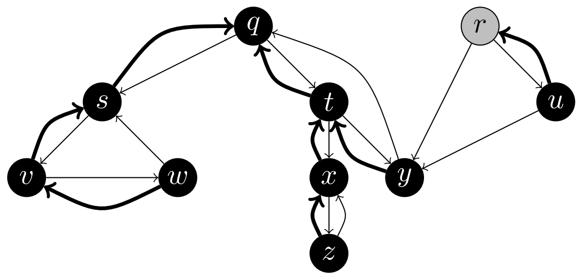

q
r
s
t
u
v
w
x
y
z
(n)

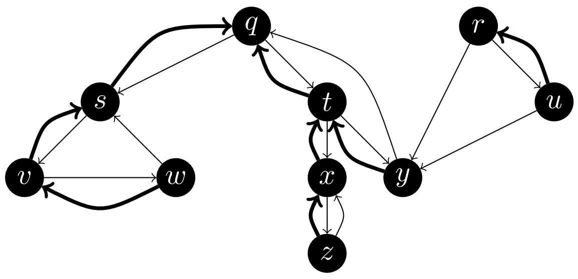

q
r
s
t
u
v
w
x
y
z
(o)


q
r
s
t
u
v
w
x
y
z
(p)


q
r
s
t
u
v
w
x
y
z
(q)


q
r
s
t
u
v
w
x
y
z
(r)


q
r
s
t
u
v
w
x
y
z
(s)


q
r
s
t
u
v
w
x
y
z
(t)

3. Problem 3 [Exercise 22-4-4, page 615]:

   This is not true. Consider a graph G consisting of vertices a, b, c, d and edges (a, b),(b, c),(a, d),(d, c), and (c, a) [see Figure 2]. Suppose that we start the DFS of TOPOLOGICAL-SORT at vertex c. Assuming that b appears before d in the adjacency list of a, the order, from latest to earliest, of finish times is c, a, d, b. The "bad" edges in this case are (b, c) and (d, c). However, if we had instead ordered them by a, b, d, c then the only bad edges would be (c, a). Thus TOPOLOGICAL-SORT doesn't always minimizes the number of "bad" edges.

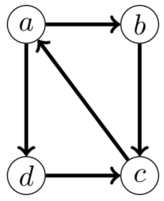
   

   Figure 2

4. Problem 4 [Exercise 23-1-5, page 629]:

   Consider a spanning tree T of G. If T does not contain e, we are done.

   Now suppose T contains e. Then removing e from T, we get two subtrees T_1 and T_2 such that the endpoints of e are in different subtrees.

   Consider the path p obtained from the cycle by deleting e. As the endpoints of p are in different subtrees, it contains an edge f connecting the two subtrees. Adding f we get a new spanning tree. But the weight of f is at most the weight of e, so the new tree is also minimal and it doesn't contain e.