CSCE 750, Homework 7

This assignment covers material from the lectures on Chapters 19–22 [3rd ed. Chapters 21–24], in preparation for Quiz 7.

Page 523, Exercise 19.1-1 [3rd ed. Page 564, Ex 21.1-1]: The CONNECTED-COMPONENTS procedure is run on the undirected graph $G = (V, E)$, where $V = \{a, b, c, d, e, f, g, h, i, j, k\}$, and the edges of E are processed in the order $(d, i), (f, k), (g, i), (b, g), (a, h), (i, j), (d, k), (b, j), (d, f), (g, j), (a, e)$. List the vertices in each connected component after each iteration of lines 3–5.

Page 523, Exercise 19.1-3 [3rd ed. Page 564, Ex 21.1-3]: During the execution of CONNECTED-COMPONENTS on an undirected graph $G = (V, E)$ with k connected components, how many times is FIND-SET called? How many times is UNION called? Express your answers in terms of $|V|$, $|E|$, and k.

Page 526, Exercise 19.2-2 [3rd ed. Page 567, Ex 21.2-2]: Show the data structure that results and the answers returned by the FIND-SET operations in the following program. Use the linked-list representation with the weighted-union heuristic. Assume that if the sets containing x_i and x_j have the same size, then the operation UNION(x_i, x_j) appends x_j's list onto x_i's list.

```
 1  for i = 1 to 16
 2      MAKE-SET(x_i)
 3  for i = 1 to 15 by 2
 4      UNION(x_i, x_{i+1})
 5  for i = 1 to 13 by 4
 6      UNION(x_i, x_{i+2})
 7  UNION(x_1, x_5)
 8  UNION(x_11, x_13)
 9  UNION(x_1, x_10)
10  FIND-SET(x_2)
10  FIND-SET(x_9)
```

Page 531, Exercise 19.3-1 [3rd ed. Page 572, Ex 21.3-1]: Redo Exercise 9.2-2 using a disjoint-set forest with union by rank and path compression. Show the resulting forest with each node including its x_i and rank.

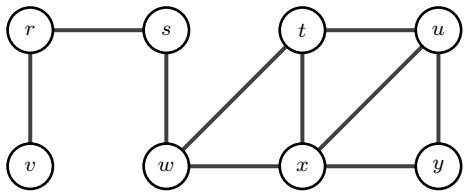
Page 552, Exercise 20.1-1 [3rd ed. Page 592, Ex 22.1-1]: Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees?

Page 553, Exercise 20.1-3 [3rd ed. Page 592, Ex 22.1-3]: The *transpose* of a directed graph $G = (V, E)$ is the graph $G^{\mathrm{T}} = (V, E^{\mathrm{T}})$, where $E^{\mathrm{T}} = \{(v, u) \in V \times V : (u, v) \in E\}$. That is, G^{T} is G with all its edges reversed. Describe efficient algorithms for computing G^{T} from G, for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

Page 562, Exercise 20.2-1 [3rd ed. Page 601, Ex 22.2-1]: Show the d and π values that result from running breadth-first search on the directed graph of Figure 20.2(a), using vertex 3 as the source.

Page 562, Exercise 20.2-2: Show the d and π values that result from running breadth-first search on the undirected graph of Figure 20.3, using vertex u as the source. Assume that neighbors of a vertex are visited in alphabetical order.

NIT1: Redo the previous exercise (Exercise 20.2-2) except use this graph instead:



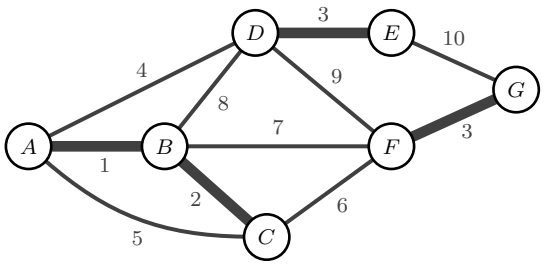
Use vertex u as the source. [3rd ed. Page 601, Ex 22.2-2]

Page 563, Exercise 20.2-7 [3rd ed. Page 602, Ex 22.2-7]: There are two types of professional wrestlers: "faces" (short for "babyfaces," i.e., "good guys") and "heals" ("bad guys"). Between any pair of professional wrestlers, there may or may not be a rivalry. You are given the names of n professional wrestlers and a list of r pairs of wrestlers for which there are rivalries. Give an $O(n + r)$-time algorithm that determines whether it is possible to designate some of the wrestlers as faces and the remainder as heels such that each rivalry is between a face and a heel. If it is possible to perform such a designation, your algorithm should produce it.

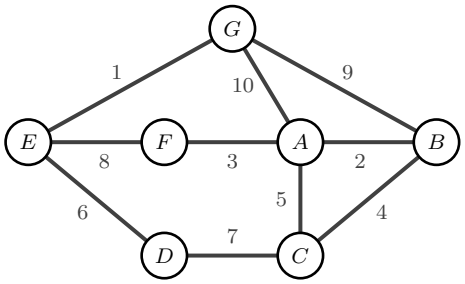
Page 571, Exercise 20.3-2 [3rd ed. Page 610, Ex 22.3-2]: Show how depth-first search works on the graph of Figure 20.6. Assume that the **for** loop of lines 5–7 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the discovery and finish times for each vertex, and show the classification of each edge.

Page 575, Exercise 20.4-1 [3rd ed. Page 614, Ex 22.4-1]: Show the ordering of vertices produced by TOPOLOGICAL-SORT when it is run on the dag of Figure 20.8. Assume that the **for** loop of lines 5–7 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically.

NIT2: Recall that Corollary 21.2 [3rd ed. Corollary 23.2] identifies certain edges that can safely be added to a partial minimum spanning tree. The diagram below shows a partially completed minimum spanning tree, with edges selected for the MST in bold. Which edges does this corollary guarantee are safe to add to the MST next?



NIT3: Use Kruskal's algorithm to find a minimum spanning tree of the graph below. List the edges considered by the algorithm in order, and indicate whether each one is selected or rejected by the algorithm.



Page 598, Exercise 21.2-6 [3rd ed. Pages 637–638, Ex 23.2-8]: Professor Borden proposes a new divide-and-conquer algorithm for computing minimum spanning trees, which goes as follows. Given a graph $G = (V, E)$, partition the set V of vertices into two sets V_1 and V_2 such that $|V_1|$ and $|V_2|$ differ by at most 1. Let E_1 be the set of edges that are incident only on vertices in V_1, and let E_2 be the set of edges that are incident only on vertices in V_2. Recursively solve a minimum-spanning-tree problem on each of the two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Finally, select the minimum-weight edge in E that crosses the cut (V_1, V_2), and use this edge to unite the resulting two minimum spanning trees into a single spanning tree.

Either argue that the algorithm correctly computes a minimum spanning tree of G, or provide an example for which the algorithm fails.

NIT4: Use Prim's algorithm to find a minimum spanning tree of the graph of the previous NIT problem, shown above. Assume that all of the vertices except the root are added to the priority queue with key ∞ at the start. Use node A as the root. List the vertices added to the MST, in order, along with any DecreaseKey operations performed on the queue.

Page 624, Exercise 22.3-1 [3rd ed. Page 662, Ex 24.3-1]: Run Dijkstra's algorithm on the directed graph of Figure 22.2, first using vertex s as the source and then using vertex z as the source. In the style of Figure 22.6, show the d and π values and the vertices in set S after each iteration of the **while** loop.

Page 625, Exercise 22.3-5 [3rd ed. Page 663, Ex 24.3-4]: Professor Gaedel has written a program that he claims implements Dijkstra's algorithm. The program produces $v.d$ and $v.\pi$ for each vertex $v \in V$. Give an $O(V + E)$-time algorithm to check the output of the professor's program. It should determine whether the d and π attributes match those of some shortest-paths tree. You may assume that all edge weights are nonnegative.

Page 625, Exercise 22.3-7 [3rd ed. Page 663, Ex 24.3-6]: Consider a directed graph $G = (V, E)$ on which each edge $(u, v) \in E$ has an associated value $r(u, v)$, which is a real number in the range $0 \leq r(u, v) \leq 1$ that represents the reliability of a communication channel from vertex u to vertex v. Interpret $r(u, v)$ as the probability that the channel from u to v will not fail, and assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices.