Name: Sabyasachi Sahoo
COSC 40403 - Analysis of Algorithms: Fall 2018: Homework 5
Due: 23:59:59 on October 21, 2018

Question	Points	Score
1	5	
2	5	
3	5	
4	5	
5	5	
6	5	
7	5	
Total:	35	

1. (5 points) Exercise 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-degree?

Solution: Given an adjacency-list representation Adj of a directed graph the out-degree of a vertex u is equal to the length of Adj[u], and the sum of lengths of all the adjacency lists in Adj is $|E|$.And hence, the time to compute the out degree of every vertex is $\Theta(V + E)$.
The in-degree of a vertex u is the number of times it appears in all the lists in Adj. So we have to search all the list for each vertex. And hence, the time to compute the in-degree of every vertex is $\Theta(VE)$.

2. (5 points) Exercise 22.1-2: Give an adjacency-list representation for a compute binary tree with 7 vertices. Give an equivalent adjacency-matrix representation. Assume that vertices are numbered from 1 to 7 as in a binary heap.

Solution:

Adjacency-list representation:

$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{3}\boxed{/}$

$\boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{5}\boxed{/}$

$\boxed{3} \rightarrow \boxed{6} \rightarrow \boxed{7}\boxed{/}$

$\boxed{4}\boxed{/}$

$\boxed{5}\boxed{/}$

$\boxed{6}\boxed{/}$

Equivalent adjacency-matrix representation:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

3. (5 points) Exercise 22.1-6: Most graph algorithms that take an adjacency-matrix representation as input require time $\Omega(V^2)$, but there are some exceptions. Show how to determine whether a directed graph G contains a ***universal sink*** (a vertex with in-degree $|V| - 1$ and out-degree 0) in time $O(V)$, given an adjacency matrix for G.

Solution:
If vertex u is a universal sink according to the definition, it means that all the other vertices have an edge to it and it has no edges to other vertices. Suppose we are looking at cell A(i, j) for i not equal to j. If A(i, j) = 0, it means vertex i does not have an edge to vertex j. This means vertex j cannot be a universal sink. On the other hand, if A(i, j) = 1, it means vertex i has an edge to vertex j, and thus vertex i cannot be a universal sink. Therefore, by looking at each cell of the matrix, you can remove one vertex from the set of potential universal sinks.

The algorithm for finding the universal sink of the graph is as follows:

Let S denote set of all vertices. Each time pick two different vertices x and y from S and look at the cell A(x, y). If A(x, y) is 1, remove x from S, and otherwise remove y from S. Therefore, in n 1 steps, and by looking at one cell of the matrix at each step, you can remove n 1 vertices from S and you only have one potential universal sink say vertex u. Check the whole column and row corresponding to vertex u in matrix A and declare u as a universal sink if the whole row is zero and the whole column is one. Otherwise, the graph does not have a universal sink. In this Algorithm, we check n 1 cells of the matrix to end up with one vertex and then we check the whole row and column corresponding to that vertex for a total of (n 1) + (2n 1) cells. Hence, the algorithm has O(n) running time

4. (5 points) The BFS algorithm listed on page 595 of our textbook and presented in class uses a queue as the data structure for storing discovered vertices. What happens if you change the data structures to use a stack instead (Be specific)? Use the example graph on page 596 to demonstrate your solution.

Solution: If you were to use a stack instead of a queue as your data structure, instead of a inorder traversal we would get a preorder or depth first traversal. In case of stack the right child pushes out before the left child is pushed out.
Steps in Stack as follows:
S- s
S- rw
S- vw
S- w
S- xt
S- uyt
S- yt
S- t
S Φ
Traversal - s,r,v,w,x,u,y,t (Preorder traversal)

5. (5 points) Exercise 22.2-7: There are two types of professional wrestlers: "babyfaces" ("good guys") and "heels" ("bad guys"). Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n professional wrestlers and we have 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 babyfaces and the remainder as heels such that each rivalry is between a babyface and a heel. If it is possible to perform such a designation, your algorithm should produce it.

Solution: Represent the problem as a graph, where each vertex represents a wrestler and each edge represents a rivalry. The graph will have n vertices and r edges. Perform as many BFSs as are needed to visit all n vertices. Assign the first wrestler to be a good guy and then assign all its immediate neighbors to be bad guys, and so on. If a wrestler is assigned to be a good guy (or bad guy), but one of its neighbors has already been assigned to be a good guy (or bad guy), report that a desired designation is not possible. The solution has breadth first search's complexity, which is O(n + r).

6. (5 points) Exercise 22.3-2: Show how the depth-first search works on the graph of Figure 22.6. 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 time and finishing times for each vertex, and show the classification of each edge.

Solution:
Graph:
Tree edges: (q,s), (s,v), (v,w), (q,t),(t,x),(x,z),(t,y),(r,u)
Back edges: (w,s),(z,x),(y,q)
Forward edges: (q,w)
Cross edges: (r,y),(u,y)

Depth-search in order: (q,s),(s,v),(v,w),(q,t),(t,x),(x,z),(t,y),(r,u)

Discovery times: (q = 1),(s = 2), (v = 3),(w = 4), (t = 8), (x = 9), (z = 10), (y = 13), (r = 17), (u = 18)
Finish times: (q = 16), (s = 7), (v = 6), (w = 5), (t = 15), (x = 12), (z = 11), (y = 14), (r = 20), (u = 19)

7. (5 points) Exercise 22.4-1: Show the ordering of vertices produced by TOPOLOGICAL-SORT on the dag of Figure 22.8, under the assumption of Exercise 22.3-2.

Solution:
nodes
m 1 20
q 2 5
t 3 4
r 6 19
u 7 8
y 9 18
v 10 17
w 11 14
z 12 13
x 15 16
n 21 26
o 22 25
s 24 24
p 27 28

The resulting sequence in decreasing order of finish time is:
p, n, o, s, m, r, y, x, w, z, u, q, t