Name: ____**ALFREDO PEREZ**

COSC 40403 - Analysis of Algorithms: Homework 5

Due: 23:30 on September 21

Some of the questions below require you to draw a graph or to trace through a graph algorithm. Please use LaTeXor another computer drawing program (i.e. PowerPoint or similar), create your diagrams.

1. (5 points) Given an adjacency-list representation of a directed graph, explain how long does it take to compute the out-degree of every vertex. Explain how long does it take to compute the in-degree?

Solution: The time taken to compute the out-degree of every vertex of a graph using a general adjacency list is O(V+E), where V is the number of vertices in the graph and E is the number of edges in the graph.

The time taken to compute the in-degree of a vertex of a graph using a general adjacency list is O(V+E), where V is the number of vertices in the graph and E is the number of edges in the graph.

2. (5 points) Give an adjacency-list representation for a complete binary tree with 15 vertices. Give an equivalent adjacency-matrix representation. Assume that vertices are numbered from 1 to 15 as in a binary heap.

Solution:

3. (5 points) 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. Explain how this time complexity may change if the graph is represented in an adjacency list.

Solution: What we can do to find out a sink is pick an arbitrary vertex, if the adjacency list of this vertex is null, then it is universal sink otherwise start traversing its adjacency list, for each element in adjacency list, we check if the list of that element is null or not. Since the Universal sink is reachable from every vertex, it must exist in the adjacency list of any arbitrary vertex. So, it is guaranteed to find the Universal sink within this list. Any node in the graph can have at most O(V) entries in its adjacency list. So, time complexity is O(V).

4. (5 points) The BFS algorithm presented in class uses a queue as the data structure for storing discovered vertices. Explain what would happen if you change the data structures to use a stack instead of a queue. Use the graph example presented in lecture to demonstrate your solution.

Solution: If you replace the queue with a stack, you will have Depth First Search (DFS), which will pick one path, and go in it as deep as it can, then pick another path and so on so forth, which is not optimal for finding the shortest path.

5. (5 points) Exercise 3.1 on page 107.

Solution:
a,b,c,d,e,f
a,d,e,b,c,f
a,b,d,c,e,f
a,d,b,c,e,f
a,b,d,e,c,f
a,d,b,e,c,f

6. (5 points) Exercise 3.2 on page 107.

Solution:

```
bool DFS(G)
  for each vertex u belongs to G.V
    u.color = WHITE
    u.parent = NULL
  bool flag = FALSE // initilise flag as false and assign true if loop found
  for each vertex u belongs to G.V
    if u.color == WHITE
      flag = DFS-VISIT(G,u)
    if flag == TRUE
      return TRUE // loop found, hence return TRUE

bool DFS-VISIT(G,u)
  u.color=GRAY
  bool flag = FALSE;
  for each v belongs to G.Adj[u] explore edge (u,v)
  if v.colour == GRAY and u.parent != v
    return TRUE // loop found

  if v.color == WHITE
    v.parent =u
    flag = DFS-VISIT(G,v)
  if flag == TRUE
    return TRUE
  u.color = BLACK
  return FALSE
```

7. (5 points) Trace through the TOPOLOGICAL-SORT algorithm and show the ordering of vertices produced on the following DAG. Assume your DFS algorithm processes the nodes in alphabetical order and the edges for each node in the adjacency list are also listed in alphabetical order.

Solution: After the topological sort the result would be:
p, n, o, s, m, r, y, v, x, w, z, u, q ,t

8. (10 points) Exercise 3.8 on page 109-110.

Solution: The claim is false