Algorithm Design
Solutions to Homework Assignment 6

---

1. Given a directed graph G with n vertices represented using an $n \times n$ adjacency matrix, give an algorithm that determines whether there is a node in G whose indegree is $n-1$ and outdegree is 0.

Soln. Let A be the adjacency matrix of G. Let us call the node in question as the star node. Our algorithm is based on the following simple observation. By examining any entry of A, we can eliminate one vertex from the candidate list for start node. If $A[i,j] = 1$ then i can not be the star node and if $A[i,j] = 0$ then j can not be the star node. The linear-time algorithm is given below.

```
FindStar(A)
    i ← 1
    j ← 2
    next = 3
    while next ≤ n + 1 do
    if A[i, j] = 1 then
        i ← next
    else
        j ← next
    next ← next + 1
    if i = n + 1 then
        candidate ← j
    else
        candidate ← i
    ▷ check if the candidate is indeed the star node
    for each vertex v do
        if v ≠ candidate then
            if A[v, candidate] = 0 or A[candidate, v] = 1 then
                return no star node
    return candidate
```

2. Chapter 3, Problem 2 (page 107). Also, design an algorithm that takes an undirected graph G and a particular edge e in it, and determines whether G contains a cycle containing e.

Solution. We run DFS on each connected component of the graph G. If DFS yields a back edge (since G is undirected, each edge is either a tree edge or a back edge) then there is a cycle, otherwise, G is acyclic. If there is a back edge (u, v) then the path from u to v

in the DFS forest along with the back edge (u, v) will give us a cycle. The running time of the algorithm is $O(n+m)$.

Let $e = (a, b)$. Do a DFS on $G - e$ starting from a. If b is in the same tree as a in the DFS forest then we know that the path from a to b in the tree along with the edge e forms a cycle. This algorithm also takes $O(n+m)$ time.

3. Chapter 3, Problem 6 (page 108).

Solution. Assume for contradiction that there is an edge (x, y) that is in G but not in T. Without loss of generality, assume that $d[x] < d[y]$ in the depth-first search starting at u. Thus at time $d[x]$ there is a path consisting only of white vertices from x to y in G. One such path consists of only the edge (x, y). By the White Path Theorem, y is a descendant of x in T. Since G is undirected and (x, y) is not a tree edge, (x, y) must be a back edge. Thus x and y must be at least two levels apart in T (G is a simple graph). This contradicts that T is also the output of BFS, as in a BFS tree the endpoints of any edge can be at most one level apart.

4. Give an efficient algorithm to find a longest path in an unrooted tree.

Solution. The following algorithm will find a longest path in an unrooted tree, T.

1. Perform DFS in T starting from any node u.
2. Let v be the node that is farthest from u in the DFS tree rooted at u.
3. Perform BFS/DFS in T starting from v.
4. Let w be the farthest node from v in the BFS/DFS tree rooted at v.
5. The path from v to w in T is a longest path in T.

The running time of the algorithm is dominated by the two DFS searches and hence is $O(|V| + |E|)$. For a tree, $|E| = |V| - 1$, hence the running time is $O(|V|)$. The correctness is based on the simple observation that given an end point v of a longest path, the other endpoint is simply the node that is farthest from v. It now remains to show that the vertex v found in Step 2 is indeed an endpoint of a longest path in T. Consider any longest path with endpoints x and y. If $x = v$ or $y = v$ we are done. So let's assume that $x \neq v$ and $y \neq v$. Without loss of generality, assume that x is at least as far from u than y. Since v is farthest from u, the path $v \rightsquigarrow lca(x, y) \rightsquigarrow x$ must be at least as long as the path $x \rightsquigarrow y$.

5. Give an efficient algorithm that takes as input a directed acyclic graph $G = (V, E)$, and two vertices s, $t \in V$, and outputs the number of different directed paths from s to t in G.

Solution.

NumDirectedPaths($G = (V, E), s, t$)

for each $v \in V$ **do**
  $numPaths(s, v) \leftarrow 0$

```
3     numPaths(s, s) ← 1
4     Order the vertices in V in topological order
5     for each vertex u in the topological ordering starting from s do
6         for each neighbor v of u do
7             numPaths(s, v) ← numPaths(s, u) + numPaths(s, v)
8     return numPaths(s, t)
```

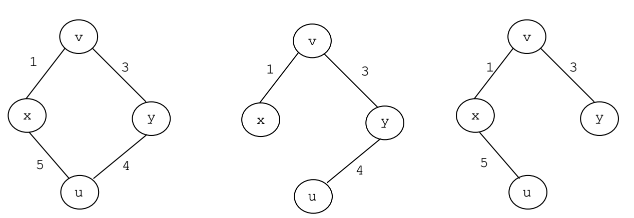
The running time of the algorithm is $O(|V| + |E|)$. The correctness follows from the following observation. Since the vertices are topologically ordered, all vertices on paths from s to u appear before u in the ordering. Thus, by the time vertex u is reached in Line 5, the number of paths from s to all predecessors of u in s-u paths would have been computed and added to give the value of $numPaths(s, u)$.

6. Consider a weighted, directed acyclic graph $G = (V, E)$ in which the edges that leave the source vertex s may have negative weights and all other edge weights are non-negative. Does Dijkstra's algorithm, started at s, correctly compute the shortest paths from s to every other vertex in the graph? Prove your answer.

Solution. Yes, Dijkstra's algorithm will compute the shortest paths correctly. Consider any vertex v and let P be the path from s to v as computed by our algorithm. Let u be the predecessor of v in the path P. Consider the time just before the vertex v was added to the set S (the set of vertices to which the shortest path is computed correctly). Let P' be any other path from s to v. We will show that the length of path P' is at least as much as the length of path P. Path P' must leave the set S via some edge, say (w, t). Dijkstra's algorithm chose to include v to S instead of t because $d(s, v) \leq d(s, t)$. Furthermore, all edges in P' between vertex t and v are non-negative and hence length of P' cannot be less than length of P.

7. Prof. Midas postulates that if every edge in an undirected graph has a unique positive weight, then the shortest path tree rooted at v in that graph is always the same as the minimum spanning tree found by Prim's algorithm when seeded initially with the vertex v. Is this correct? If so, prove it. If not, give a counter-example.

Solution. Below is the counter-example.



Graph, G   MST   Shortest Path Tree