HW08 Solutions

Data Structure and Algorithm (New York University)

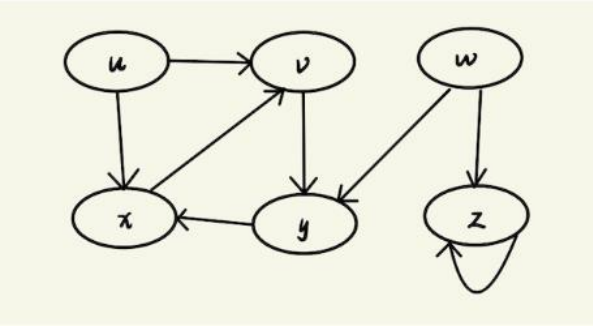
EL9343 Homework 8

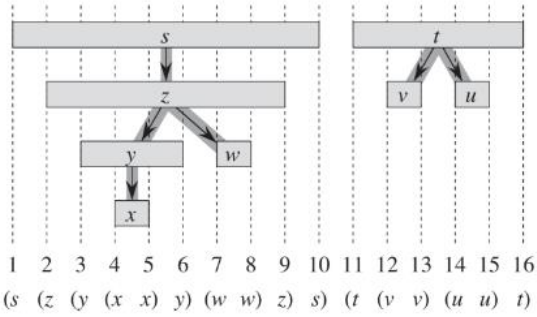
Due: Nov. 9th 8:00 a.m.

1. What is the running time of DFS if the graph ($G = (V, E)$, where there are $|V|$ nodes and $|E|$ edges) is given as an adjacency list and adjacency matrix? Justify your running time.
   Solution:
   For adjacency list: $O(|V| + |E|)$; for adjacency matrix: $O(|V|^2)$.
   In DFS, we will visit every node once and visit those nodes are adjacent to them. In both case, we first need $O(|V|)$ time to visit all nodes.
   If we use adjacency list, we only need to visit every edge in each list, which is $O(|E|)$. So, the total running time is $O(|V|) + O(|E|) = O(|V| + |E|)$.
   If we use adjacency matrix, although there are only $|E|$ edges, once we visit a node (i.e. the i-th node), we will visit the element in the i-th row of the adjacency matrix. In total, we will visit every element in the adjacency matrix (no matter it is 0 or 1), which is $|V|^2$. So the total running time will be $O(|V|^2) + O(|V|) = O(|V|^2)$.

2. Write a method that takes any two nodes u and v in a tree T whose root node is s, and quickly determines if the node u in the tree is a *descendant* or *ancestor* of node v.
   Solution:
   We can simply run DFS on a tree T from root node s and record the discover time and finish time of every node. Because T is a tree, the number of edges is $|E| = |V| - 1$. So, the total running time is $O(|V| + |E|) = O(|V|)$. As we have shown in the lecture, given any 2 nodes u, v in the DFS tree, if $u.d < v.d < v.f < u.f$, u is the ancestor of v. Else if $v.d < u.d < u.f < v.f$, u is the descendant of v.
   Because we only need $O(|V|)$ time to complete DFS and store the discover time and finish time, and other operations are in constant time, the total running time is $O(|V|)$. More specifically, we need $O(|V|)$ time to initialize and $O(1)$ time to determine the relationship between 2 nodes.

3. Draw the parenthesis structure of the DFS of bottom left figure (start from u, assume that DFS considers vertices in alphabetical order) and see the example parenthesis structure as is shown in the bottom right.





   Solution:
   (u (v (y (x x) y) v) u) (w (z z) w)

4. **Bipartiteness** Given a undirected graph $G = (V, E)$, it is *bipartite* if there exist U and W such that $U \cup W = V$, $U \cap W = \emptyset$, and every edge has one endpoint each in U and W.
   (a) Prove: G is bipartite only if G has no odd cycle. (Hint: proof by contradiction)
   (b) In fact, G is bipartite if and only if G has no odd cycle. Suppose this is given, consider the Algorithm 1 and briefly describe why it is correct.
   (c) Analyze the time complexity (worst-case, big-O) of the algorithm, in terms of $|V|$ and $|E|$. (Hint: can we check that there is no edge inside any layer in $O(|E|)$? Why?)

   Solution:
   (a) Proof by contradiction, we assume G is bipartite when there is an odd cycle in G.
   Let this cycle be $u_1 \to u_2 \to u_3 \to \cdots \to u_{2k+1} \to u_1$. Since G is bipartite, let the bipartition be U and W, and we can assign $u_1, \ldots, u_{2k+1}$ to either U or W.
   Without loss of generality, we let $u_1 \in U$. There is an edge between u_1 and u_2. Because of bipartiteness, u_2 must be in W. Similarly, $u_3 \in U, u_4 \in W, \ldots$.
   Finally, $u_1, u_3, \ldots, u_{2k+1} \in U, u_2, u_4, \ldots, u2k \in W$, and there is an edge between u_{2k+1} and u_1.

Algorithm 1 Testing bipartiteness of graphs

Do BFS starting at any node u
Let $L_0, L_1, L_2, \ldots, L_k$ be layers in the resulting breadth-first tree ($L_0 = \{u\}$, $L_i, i = 1, 2, \ldots, k$ contains the vertices at distance i from u)
if There is no edge inside any layer L_i **then**
  Declare G to be bipartite, and $U = L_0 \cup L_2 \cup L_4 \cup \ldots, W = L_1 \cup L_3 \cup L_5 \cup \ldots$ are the bipartition
else
  Declare G to be non-bipartite
end if

However, u_{2k+1} and u_1 are all in set U, which contradicts to the assumption that G is bipartite (every edge has one endpoint each in U and W).
Thus, G is bipartite only if G has no odd cycle.

(b) In BFS, it is impossible that there is an edge between a node in L_i and a node in L_j, where $|i - j| \geq 2$. And because if there is no edge inside any layer L_i, there will not be odd cycle, thus G will be bipartite.

(c) The time complexity is $O(|V| + |E|)$. Doing the BFS is $O(|V| + |E|)$, checking that there is no edge inside any layer is $O(|E|)$, and anything else is $O(1)$. Therefore, it is $O(|V| + |E|)$ in total.
We can check each edge in set E on the two layer index of its two endpoints (for each node, its layer index is i if the node is in layer L_i), which is $O(|E|)$. If every pair of the two endpoints are in different layers, we are done. This is faster than checking if an edge exists in every pair of nodes inside a layer, which in worst case is $O(|V|^2)$.