COL 351 : Analysis and Design of Algorithms

Tutorial Sheet - 4

Question 1 Complete the pseudo-code below to obtain an $O(|V| + |E|)$ time algorithm for computing cut-vertices of an n-vertex undirected (possibly unconnected) graph $G = (V, E)$.

```
VISITED[v] ← True;
HIGH-POINT[v] ← LEVEL[v];
foreach w ∈ N(v) do
    if (VISITED[w] = False) then
        Set PARENT[w] ← v  and  LEVEL[w] = 1 + LEVEL[v];
        ________________;
        ________________ ← ______________________________________;
        if ________________ ≥ ___________ then
            ___________________________;
        end
    else if (w ≠ PARENT[v]) then
        HIGH-POINT[v] ← _________________________________;
    end
end
```

Procedure DFS(v)

```
Let (v_1, ..., v_n) be any ordering of vertices of G;
for i = 1 to n do
    VISITED[v_i] ← False  and  IS-CUT-VERTEX[v_i] ← False ;
end
for i = 1 to n do
    if (VISITED[v_i] = False) then
        LEVEL[v_i] ← 0;
        Invoke DFS(v_i);
        if (v_i has one child) then _____________________________;
    end
end
```

Procedure Compute-Cut-vertices(G)

Question 2 Prove that any n vertex undirected graph contains at most $n-1$ bridge-edges and at most $n-2$ cut-vertices.
(Hint: Prove that in any graph G there are at least two vertices that are not cut-vertices.)

Question 3 A *topological ordering* of a directed acyclic graph G is a linear order of its vertices such that for every directed edge (x, y) in G, x appears before y in the ordering.

(a) Let G be a DAG on n vertices. Prove that following holds for each DFS traversal of G.

- For any directed edge (x, y), $\text{FINISH-TIME}(y) < \text{FINISH-TIME}(x)$.
- If $L = (v_1, \ldots, v_n)$ is a list of vertices of G in decreasing order of finish time, that is, $\text{FINISH-TIME}(v_1) > \text{FINISH-TIME}(v_2) > \cdots > \text{FINISH-TIME}(v_n)$, then, L is a topological ordering of G.

(b) Prove that a DAG always contains a vertex of in-degree zero.

Question 4 Let $A = (a_1 \ldots a_n)$ be a sequence of integers. Devise an $O(n^2)$ time algorithm to compute a **longest increasing subsequence (LIS)** of A.

Question 5 Let $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_m)$ be two sequences, where a_i's and b_j's are positive integers. Devise an $O(mn)$ time algorithm to compute a **longest common increasing subsequence (LCIS)** of A and B.