# COL351: Analysis and Design of Algorithms

Tutorial Sheet - 3

August 24, 2022

**Question 1** Let $G = (V, E)$ be a directed graph. Let $C_1, C_2, \ldots, C_k$ denote the strongly connected components of $G$ (each $C_i$ is a subset of vertices).

We construct another directed graph $H$ from $G$ as follows: There are $k$ vertices in $H$, namely, $v_1, \ldots, v_k$. There is a directed edge $(v_i, v_j)$ in $H$ iff there is an edge in $G$ from a vertex in $C_i$ to a vertex in $C_j$.

i) Give a linear time algorithm to construct $H$ from SCCs $C_1, C_2, \ldots, C_k$ (without using DFS).

ii) Prove that $H$ is acyclic.
(Hint: Show that if there is a cycle in $H$, say on the first $t$ vertices, i.e. $(v_1, \ldots, v_t)$ is a cycle in $H$, then vertices in set $C_1 \cup \cdots \cup C_t$ must be strongly connected to each other. This is a contradiction.)

iii) Consider a DFS traversal of $G$. For $i \in [1, k]$, let $r_i$ be a vertex of highest finish time in $S_i$. If FINISH-TIME$(r_1) < \cdots <$ FINISH-TIME$(r_k)$, then prove that the list $(v_k, \ldots, v_1)$ forms a topological ordering of vertices of $H$.

**Question 2** Prove that any $n$ vertex undirected graph contains at most $n - 1$ bridge-edges and at most $n - 2$ cut-vertices. Also present examples to show that these bounds are tight.
(Hint: Prove that in any graph $G$ there are at least two vertices that are not cut-vertices.)

**Question 3** A graph $G = (V, E)$ is said to be **bipartite** if vertex set $V$ can be partitioned into two sets $X$ and $Y$ such that every edge $e \in E$ connects a vertex in $X$ to one in $Y$.

i) Show that any graph having an odd length cycle cannot be bipartite.

ii) Let $G = (V, E)$ be a connected graph and $T$ be a BFS tree of $G$. Show that $G$ is bipartite iff for each edge $(a, b) \in E$, $|Level(a) - Level(b)| = 1$.

iii) Devise an $O(n+m)$ time algorithm to check if a connected/unconnected graph $G$ is bipartite.

**Question 4** 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 $G = (V, E)$.

```
1  VISITED[v] ← True;
2  HIGH-POINT[v] ← LEVEL[v];
3  foreach w ∈ N(v) do
4  |  if (VISITED[w] = False) then
5  |  |  Set PARENT[w] ← v  and  LEVEL[w] = 1 + LEVEL[v];
6  |  |  _____;
7  |  |  _____ ← _____;
8  |  |  if _____ ≥ _____ then
9  |  |  |  _____;
10 |  |  end
11 |  else if (w ≠ PARENT[v]) then
12 |  |  HIGH-POINT[v] ← _____;
13 |  end
14 end
```

**Procedure** DFS(v)

```
1  Let (v_1, ..., v_n) be any ordering of vertices of G;
2  for i = 1 to n do
3  |  VISITED[v_i] ← False   and  IS-CUT-VERTEX[v_i] ← False ;
4  end
5  for i = 1 to n do
6  |  if (VISITED[v_i] = False)  then
7  |  |  LEVEL[v_i] ← 0;
8  |  |  Invoke DFS(v_i);
9  |  |  if (v_i has one child)  then _____;
10 |  end
11 end
```

**Procedure** Compute-Cut-vertices(G)

**Question 5** Let $y$ and $z$ be any two vertices in a directed acyclic graph $G$. If there is a path from $y$ to $z$, then prove that the following holds true for each DFS traversal carried out in $G$.

1. FINISH-TIME$(z)$ < FINISH-TIME$(y)$.
2. If START-TIME$(y)$ < START-TIME$(z)$, then $z$ must be a descendant of $y$ in the DFS tree.