Homework 5

CS6033 Design and Analysis of Algorithms I
Fall 2023
(Sec. B, Prof. Yi-Jen Chiang)

Due: Wed. 11/29 by 1pm
(submit online on NYU Brightspace; one submission per group)
Maximum Score: 102 points

Note: This assignment has 2 pages.

1. (14 points)
(a) Consider the directed graph in the textbook Figure 20.6 (page 571). Copy the figure, and run DFS on it starting with vertex q. Mark **tree edges** on the picture (like this: put a "$\times$" on an edge $\rightarrow$) and give DFS numbers (discovery/finish times) for each vertex.
Note: Whenever there is more than one option during the search, always follow the **alphabetically increasing** order. **(7 points)**

(b) Consider the DAG in the textbook Figure 20.8 (page 575). Copy the figure; run the topological sorting algorithm on it and give a topological sorting order of the vertices.
Note: Whenever there is more than one option in constructing the order, always follow the **alphabetically increasing** order. **(7 points)**

2. (9 points)
Give a (simple) example to explain how a vertex u of a directed graph G can end up in a depth-first-search tree containing only u, even though u has both incoming and outgoing edges in G.

3. (19 points)
Textbook Exercise 20.4-5 (page 576). We assume that the graph G is given by an adjacency-list representation. We refine this question into the following three sub-questions.

(a) Design and analyze an algorithm to compute the in-degrees of all vertices in worst-case time $O(V + E)$. **(9 points)**

(b) Design and analyze an algorithm to perform topological sorting in worst-case time $O(V + E)$ by implementing the idea described in this question, given that the in-degrees of all vertices are already available (as computed in **part (a)**). **(7 points)**

(c) Discuss what happens to this algorithm (as developed in **part (b)**) if G has cycles. **(3 points)**

4. (12 points)
Professor Goodhead came up with a new approach to simplify the algorithm for strongly connected components: in the second DFS, perform it on the **original graph** G in the order of **increasing finish time** (rather than performing DFS on the transpose graph G^T in the order of decreasing

finish time) — this new approach is simpler since the computation of the transpose graph G^T is avoided. Is this new algorithm correct? You should either prove that this algorithm is correct, or give a counter-example to argue that this algorithm is incorrect.

5. (18 points)
Let $G = (V, E)$ be a connected, **undirected** graph. Your task here is to compute a **cycle** in G that traverses each edge in E **twice, exactly once in each direction.**

(a) First consider the special case where G is a tree. Design and analyze an algorithm to carry out the task in worst-case time $O(V + E)$.
(**Hint:** Observe the traversal of DFS.) **(8 points)**

(b) Now consider the general case (where G may contain cycles). Design and analyze an algorithm to carry out the task in worst-case time $O(V + E)$.
(**Hint:** Extend the algorithm in **part (a)** and consider edge classifications.) **(10 points)**

6. (30 points)
Let $G = (V, E)$ be a directed graph, where each vertex v has a **real-number** label $L(v)$. For vertices $u, v \in V$, define $D_L(u, v)$ as

$D_L(u, v) = L(v) - L(u)$ if there is a path from u to v in G, and
$D_L(u, v) = -\infty$ else.

Your task here is to find vertices $u^*, v^* \in V$ such that $D_L(u^*, v^*)$ is maximum over all pairs of vertices in V.

(a) Consider the special case where G is a directed **acyclic** graph (DAG). To carry out the task, we split the work into two parts:

(i) For each vertex v, define $min\_reaching\_L(v) = \min\{L(u) | \text{there is a path from } u \text{ to } v \text{ in } G\}$. Design and analyze an algorithm that computes, for **each vertex** v, the value $min\_reaching\_L(v)$ and the the corresponding vertex u that realizes $min\_reaching\_L(v)$, in worst-case time $O(V+E)$. **(12 points)**

(ii) Given that you already have, for each vertex v, the value $min\_reaching\_L(v)$ and the the corresponding vertex u that realizes $min\_reaching\_L(v)$ (as computed in **part (i)**), discuss how you can carry out the task in overall $O(V + E)$ worst-case time. **(6 points)**

(b) Now consider the general case where G is a (general) directed graph. Design and analyze an algorithm to carry out the task in worst-case time $O(V + E)$.
(**Hint:** Consider what you need to do to apply the algorithm of **part (a)**.) **(12 points)**