Homework 5

CS6033 Design and Analysis of Algorithms I

Fall 2024

(Sec. B, Prof. Yi-Jen Chiang)

Due: Wed. 11/27 by 1pm
(submit online on NYU Brightspace; one submission per group)
Maximum Score: 106 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 counter-example to the conjecture that if a directed graph G contains a **path** from a vertex u to another vertex v, and if $u.d < v.d$ in a depth-first search of G (where $x.d$ means the discovery time of vertex x), then v is a descendant of u in the depth-first forest produced.

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 Smith 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. (22 points)
Given an undirected graph $G = (V, E)$ with at most one cycle, your task here is to assign each edge a direction so that every vertex has in-degree at most one.
(a) First consider a simpler case. Suppose we know that G has **no** cycle. Design and analyze an algorithm to carry out the task in worst-case time $O(V + E)$. **(7 points)**

(b) Now we only know that G has **at most one** cycle. Design and analyze an algorithm to carry out the task in worst-case time $O(V + E)$. **(15 points)**

6. (30 points)
Let $G = (V, E)$ be a **directed** graph. Each vertex u has a **real-number** key value $Key(u)$. For a vertex u, define $Max\_Key(u)$ to be

$$Max\_Key(u) = \max\{Key(v) | v \text{ can be reached from } u\},$$

where v can be reached from u if v is u itself or if there is a path from u to v in G. Your task here is to compute $Max\_Key(u)$ for all vertices $u \in V$ in worst-case time $O(V + E)$.
(**Note:** You should **not** sort the key values since it will take more than linear time.)

(a) Consider the special case where G is a directed **acyclic** graph (DAG). Design and analyze an algorithm to carry out the task in worst-case time $O(V + E)$. **(15 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)**.) **(15 points)**