CS 584/684 Algorithm Design and Analysis

Homework 4

Portland State U, Winter 2021 *01/26/21*
Lecturer: Fang Song *Due: 02/02/21*

Instructions. This problem set contains 4 pages (including this cover page) and 3 questions. A random subset of problems will be graded.

- Your solutions will be graded on *correctness* and *clarity*. You should only submit work that you believe to be correct, and you will get significantly more partial credit if you clearly identify the gap(s) in your solution. It is good practice to start any long solution with an informal (but accurate) summary that describes the main idea. You may opt for the "I take 15%" option.
- You need to submit a PDF file before the deadline. Either a clear scan of you handwriting or a typeset document is accepted. You will get 5 bonus points for typing in LaTeX (Download and use the accompany TeX file).
- You may collaborate with others on this problem set. However, you must **write up your own solutions** and **list your collaborators and any external sources** for each problem. Be ready to explain your solutions orally to a course staff if asked.
- For problems that require you to provide an algorithm, you must give a precise description of the algorithm, together with a proof of correctness and an analysis of its running time. You may use algorithms from class as subroutines. You may also use any facts that we proved in class or from the book.

1. (Semi-connected graphs) A directed graph G is *semi-connected* if, for every pair of vertices u and v, either u is reachable from v or v is reachable from u (or both).
   (a) (5 points) Give an example of a DAG with a unique source (a source is a vertex with no entering edges) that is **not** semi-connected.
   (b) (10 points) Give an algorithm to determine whether a given DAG is semi-connected.
   (c) (10 points) Give an algorithm to determine whether an arbitrary directed graph is semi-connected.

2. (Component graph) Given a directed graph $G = (V, E)$, we define another graph $G^{\text{SCC}} = (V^{\text{SCC}}, E^{\text{SCC}})$ called the *component graph* as follows. Suppose that G has strongly connected components $C_1, C_2, \ldots, C_k$. The vertex set V^{SCC} is $\{v_1, \ldots, v_k\}$ where $v_i \in C_i$. There is an edge $(v_i, v_j) \in E^{\text{SCC}}$ if G contains a directed edge $x \to y$ for some $x \in C_i$ and some $y \in C_j$. Alternatively, imagine contracting all edges whose incident vertices are within the same strongly connected component of G, and the resulting graph will be G^{SCC}.

   (a) (10 points) Prove or disprove that G^{SCC} is a DAG.

   (b) (10 points (bonus)) Give an $O(|V| + |E|)$-time algorithm to compute the component graph of a directed graph $G = (V, E)$.

3. (20 points) (Longest forward-backward contiguous substring) Describe and analyze an efficient algorithm to find the length of the longest *contiguous* substring that appears both *forward* and *backward* in an input string $T[1, \ldots, n]$. The forward and backward substrings must NOT overlap. Here are several examples.

   - Given the input string `ALGORITHM`, your algorithm should return 0.
   - Given the input string `RECURSION`, your algorithm should return 1, for the substring `R`.
   - Given the input string `REDIVIDE`, your algorithm should return 3, for the substring `EDI`. (The forward and backward substrings must not overlap!)
   - Given the input string `DYNAMICPROGRAMMINGMANYTIMES`, your algorithm should return 4, for the substring `YNAM`. (It should not return 6, for the subsequence `YNAMIR`, because it's not contiguous.).