COL 702

Homework I

Due on Sept. 4, 2021

In all cases, prove (succintly) why your algorithm is correct. Do NOT show any examples. Needlessly long arguments will fetch negative marks.

1. Let G 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 now construct another directed graph H as follows: there are k vertices in H, denoted $v_1, \ldots, v_k$. There is a directed edge (v_i, v_j) in H if and only if there is an edge in G from a vertex in C_i to a vertex in C_j. Prove that H is a DAG. Give a linear time algorithm to construct H.

2. Call a directed graph G to be *weakly* connected if for every pair of vertices u, v , either there is a directed path from u to v or a directed path from v to u. Give a linear time algorithm to check if a directed graph is weakly connected.

3. Describe an efficient algorithm to find the second minimum shortest path between vertices u and v in a weighted graph without negative weights. The second minimum weight path must differ from the shortest path by at least one edge and may have the same weight as the shortest path.

4. Given a directed acyclic graph that has maximal path length k, design an efficient algorithm that partitions the vertices into $k+1$ sets such that there is no path between any pair of vertices in a set.

5. Suppose you are given an undirected graph G and a tree T on the same set of vertices. We would like to know whether it is possible to have an adjacency list representation of G (note that there are mutliple options here as the adjacency list can arrange the neighbours of a vertex in any order) such that running Breadth First Search on G with this adjacency list will result in T being the BFS Tree. Give an efficient algorithm to solve this problem (note that it is not enough for T to have the shortest path property, why?).

6. Solve the above problem, but replace "BFS" by "DFS" everywhere.