CS 6515: Homework 1

Due on Sunday, September 3, 11:59pm via Gradescope.
Late submission with 10% penalty until Monday, September 4, 11:59pm.

Professor Brito

Instructions.

For the graded problems, when asked to design an algorithm you must describe it in plain English. Do not use pseudocode! Always address the runtime of your design. You should aim for the fastest (in big-O notation).

Suggested reading.

From Algorithms by Dasgupta, Papadumitriou, and Vazirani, ([DPV] from now on) chpater 3 covers DFS and SCC, and chapter 4 covers BFS.

Practice Problems.

There are multiple problems in chapter 3 that cover fundamentals and simple applications of DFS and SCC. You are welcome to explore those! We recommend 3.1-3.8, 3.11, 3.15, 3.18, 3.22, 3.23, 3.24. Do not turn these problems in.

Problem 1

We saw in class how to find a `topological sorting` of the vertices of a directed acyclic graph (a.k.a. DAG) using `DFS`. Recall the definition of *source and sink vertex*. In this problem we explore when a DAG has a unique topological sorting.

(a) Give an explicit example of a DAG with exactly one source vertex and one sink vertex but multiple topological sorting.
We expect: a clear description of your graph (a picture is OK), and two distinct topological sorting.

(b) Part (a) shows that uniqueness of the source and sink is not sufficient to have a unique topological sorting. Design an algorithm that takes as input a DAG and output YES if it has a unique topological sorting, and outputs NO otherwise. Your algorithm should run in linear time.
We expect: a clear description of your design in plain English, a brief justification of its correctness, and a analysis of its runtime (in big-O notation).

Problem 2

We know that finding a long cycle is a hard problem, how about a short one? Let $G = (V, E)$ be an undirected graph such that every vertex has degree greater or equal than three. Design an algorithm that outputs a cycle of length $O(\log(n))$, if such cycle exits, and returns NO otherwise. Your algorithm should run in linear time.
We expect: a clear description of your design in plain English, a justification of its correctness where the upper bound on the length of the cycle is rigurously justified, and a analysis of its runtime (in big-O notation).