Homework 3
Design and Analysis of Algorithms
Sections BCS-4C, BCS-6A, Spring 2021

Due Monday, May 22, 2021 at 1159pm

1. **a.** Given a knapsack of capacity 50, what is the maximal value obtainable with three items of $60, $100 and $120 with weights 10, 20 and 30 respectively?

   Show how the exhaustive algorithm will obtain this value. Recall that the exhaustive algorithm checks all possible candidate solutions and then picks the one which maximizes the accumulated value.

   b. Write pseudo-code for the Fractional Knapsack algorithm. The running time of the algorithm should be O(nlgn) for a problem over n items. The inputs to the algorithm are: arrays v and w of sizes n each containing the values and weights of the n items; a number W which is the capacity of the knapsack; and the number n.

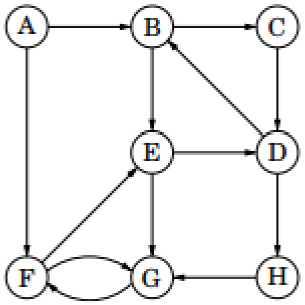
2. We know that the activity selection algorithm works by picking non-overlapping activities in the increasing order of their finish times. Which of the following two strategies will also produce an optimal solution? Explain your answer:

   a. **Picking non-overlapping activities in the decreasing order finish times.**

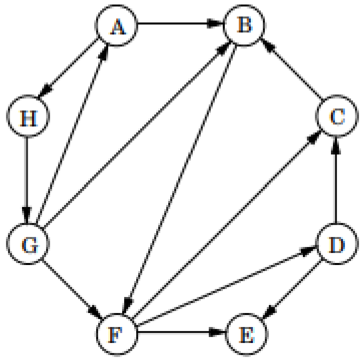
   b. **Picking non-overlapping activities in the decreasing order of start times.**

3. Run the DFS algorithm on the following graph. Draw the DFS tree (or forest). Label each node with pre and post numbers. Indicate the backward, forward and cross edges. Wherever multiple options are available, visit the nodes in the alphabetic order.

A



B

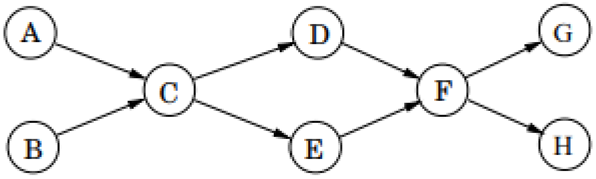


4. Recall **Algorithm I** for the linearization of a DAG, as explained in “Lecture 19 Graphs II”. Following is a broad outline of the algorithm:

    - Create a data structure D, based on the in-degrees of the graph nodes.
    - Extract nodes from D based on lowest in-degree first.
    - For each node x extracted from D, decrement the in-degrees of nodes y, for each edge (x, y).
    - Add each extracted node to another data structures S. (S should be such a data structure that one can retrieve the nodes from S in linearized order.)

Convert this broad outline into a precise pseudo-code, specifying data structures D and S and their exact functions which you will need to use. Make sure that the running time of your algorithm is no more than O((|V|+|E|)lg|V|).

5. Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.



(a) Indicate the pre and post numbers of the nodes.
(b) What are the sources and sinks of the graph?
(c) What topological ordering is found by the algorithm?
(d) How many topological orderings does this graph have?

6. You are given a DAG G=(V, E) of the courses in your curriculum (nodes), with the edges between them showing the pre-requisite relationship, i.e. an edge going from A to B means that A must be studied before B: so you take the course A in one semester and then B must come in the following semester.

    Imagine that other than the prerequisite relationship, there is no bound on the number of courses you can take in a semester. Design an O(|V|+|E|) algorithm to find the minimum number of semesters needed to graduate.

    Hint: notice that this problem asks you to find the longest path in the DAG, since there will be at least that many semesters needed to graduate.

7. The reverse of a directed graph G = (V, E) is another directed graph $G^R = (V, E^R)$ on the same vertex set, but with all edges reversed; that is, $E^R = \{(v, u): (u, v) \in E\}$. Give a linear-time algorithm for computing the reverse of a graph in the adjacency list format. Give a clearly specified pseudo-code.