CSCI 2300 — Introduction to Algorithms
Exam 1 Prep and Sample Questions (document version 1.0)

- Exam 1 is scheduled for 6:00-7:50PM on Thursday 2/20 in West Hall Auditorium
- Be sure to bring your RPI ID to the exam
- Exam 1 is open book(s), open notes; given that you will be in a seat with a small fold-up "desk," be sure you prepare and consolidate your notes
- For extra-time accommodations, watch your RPI email for an earlier start time and location
- Make-up exams are given only with an official excused absence (`http://bit.ly/rpiabsence`)
- Exam 1 covers everything up to and including Tuesday 2/18 lecture
- Exam 1 covers: Homeworks 1, 2, and 3; Labs 1, 2, and 3; and Recitations 1 and 2
- Key topics are graph algorithms, runtime efficiency using $O()$, and greedy algorithms
- All work must be your own; do not even think of copying from others
- Be as concise as you can in your answers; long answers are difficult to grade

Sample problems

Work on the problems below as practice problems as you prepare for the exam. **Also work on the "Warm-up problems" and graded problems from our first three homeworks.**

Feel free to post your solutions in the Discussion Forum (except for graded Homework 3 problems); and reply to posts if you agree or disagree with the proposed approaches/solutions.

Some selected solutions will be posted on Wednesday 2/19 (for your lab/recitation sections).

1. Given an adjacency list representation, write an algorithm to reverse all edges of given directed graph G. Make sure your algorithm is correct and runs in linear time.

   Repeat the above with an adjacency matrix representation.

2. A binary tree is a tree with 0, 1, or 2 child nodes. Let B_n denote the number of binary trees possible with n nodes. Therefore, $B_1 = 1$ and $B_2 = 2$.

   (a) Determine B_3 and B_4 by drawing all possible binary trees in each case.

   (b) Generalize and define a recurrence relation for B_n. As a hint, consider how many nodes there are in the left subtree and therefore how many nodes there are in the right subtree.

   (c) Solve the recurrence relation and show using Big $O()$ notation.

3. Construct an undirected graph G with five nodes and seven edges such that the `pre` and `post` numbers (from the DFS algorithm) for all but one of the nodes differ by at least 3 (i.e., for each node u in G, $\texttt{post}(u) > \texttt{pre}(u) + 2$).

4. Given directed acyclic graph G, write an efficient algorithm to identify whether there exists a node that can be reached by every other node in G. Show the runtime complexity of your algorithm.

5. Write an algorithm to find a path that traverses all edges of directed graph G exactly once. You may visit nodes multiple times, if necessary. Show the runtime complexity of your algorithm.

6. Draw a connected weighted undirected graph with exactly six nodes that contains exactly three cycles; further, all edge weights must be distinct.

   Repeat the above, but this time a minimum spanning tree of the graph must contain the largest weighted edge.

7. Design an algorithm that finds the largest weighted spanning tree of a given undirected graph. Show the runtime complexity of your algorithm.

8. Give an example graph in which Dijkstra's algorithm fails to find the shortest path.

9. The *diameter* of a graph is the largest distance between any pair of nodes. Design an algorithm to find the diameter of a graph. Show the runtime complexity of your algorithm.

10. DPV Problem 3.13 (all parts)

11. DPV Problem 3.14 (all parts)

12. DPV Problem 4.4

13. DPV Problem 4.8