INSTRUCTIONS

Homework should be done in groups of one to three people. You are free to change group members at any time throughout the quarter. Problems should be solved together, not divided up between partners. A single representative of your group should submit your work through Gradescope. Submissions must be received by 11:59pm on the due date, and there are no exceptions to this rule.

Homework solutions should be neatly written or typed and turned in through Gradescope by 11:59pm on the due date. No late homeworks will be accepted for any reason. You will be able to look at your scanned work before submitting it. Please ensure that your submission is legible (neatly written and not too faint) or your homework may not be graded.

Students should consult their textbook, class notes, lecture slides, instructors, TAs, and tutors when they need help with homework. Students should not look for answers to homework problems in other texts or sources, including the internet. Only post about graded homework questions on Piazza if you suspect a typo in the assignment, or if you don’t understand what the question is asking you to do. Other questions are best addressed in office hours.

Your assignments in this class will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should always explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

For questions that require pseudocode, you can follow the same format as the textbook, or you can write pseudocode in your own style, as long as you specify what your notation means. For example, are you using “=” to mean assignment or to check equality? You are welcome to use any algorithm from class as a subroutine in your pseudocode. For example, if you want to sort list A using InsertionSort, you can call InsertionSort(A) instead of writing out the pseudocode for InsertionSort.

REQUIRED READING Rosen 10.4 through Theorem 1, 11.1, 11.2 through Theorem 2

KEY CONCEPTS DAGs and topological orderings, graph search and reachability, rooted and unrooted trees, basic counting principles (sum and product rules)
1. In this scheduling problem, you are given a set of tasks (shown in bold) that all need to be performed as you are getting ready in the morning. However, for practical reasons, certain tasks cannot be started until other tasks have been completed. These constraints are:

- Leaving the House cannot occur until Putting on Shoes, Packing Lunch, Combing Hair, Taking Vitamins and Brushing Teeth are done.
- Packing Lunch, Showering, Making Breakfast, Putting on Socks, and Putting on Clothes cannot occur until Getting Out of Bed is done.
- Combing Hair cannot occur until Showering is done.
- Taking Vitamins and Brushing Teeth cannot occur until Eating Breakfast is done.
- Eating Breakfast cannot occur until Making Breakfast is done.
- Getting Out of Bed cannot occur until Waking Up is done.
- Putting on Shoes cannot occur until Putting on Socks and Putting on Clothes are done.
- Putting on Socks and Putting on Clothes cannot occur until Showering is done.

(a) (5 points) Draw a graph that models this situation. Is your graph directed or undirected, and why? Does it contain cycles, and why?

(b) (5 points) Use the graph to help you list all the tasks in an order in which they can actually be performed, given the above set of constraints. Explain how you used the graph to find your answer.

2. Let $G$ be a DAG with vertices labeled 1, 2, . . . , $n$. Define the adjacency matrix and adjacency list representations of $G$ as in Rosen, page 669.

(a) (3 points) Let $A$ be the adjacency matrix that represents $G$. Write pseudocode that takes $A$ as an input and outputs the array `InDegree[]` for the graph $G$.

(b) (2 points) Let $A$ be the adjacency matrix that represents $G$. Write a high-level English description of an algorithm that takes as inputs $A$ and a source vertex $v$ and outputs the adjacency matrix that represents $G - v$.

(c) (3 points) Let $L$ be the adjacency list that represents $G$. Write pseudocode that takes $L$ as an input and outputs the array `InDegree[]` for the graph $G$.

(d) (2 points) Let $L$ be the adjacency list that represents $G$. Write a high-level English description of an algorithm that takes as inputs $L$ and a source vertex $v$ and outputs the adjacency list that represents $G - v$.

3. In this puzzle, a traveler needs to transport a lion, a poodle, and a giant chocolate bar across a river, but his boat can only fit himself plus one of the other three. A configuration of this puzzle can be described as a length 4 binary string, where each bit from left to right describes the status of the traveler, the lion, the poodle, and the chocolate, respectively. A 0 means that the item is on one side of the river (the starting side) and a 1 means that the item is on the other side of the river. For example the configuration 1100 means that the traveler and the lion are not on the starting side of the river, and the poodle and the chocolate are on the starting side of the river.

(a) (4 points) Which of the 16 possible configurations can actually be reached from the starting configuration (0000) if the following rules apply?
- It is forbidden to leave the lion and the poodle on a shore together without the traveler, because the lion will eat the poodle.
- It is forbidden to leave the poodle and the chocolate on a shore together without the traveler, because the poodle will eat the chocolate.

Justify your answer using an appropriate graph.
(b) (4 points) How can the traveler, the lion, the poodle, and the chocolate all make it across the river in the fewest number of boat trips possible? Say how many boat trips are required, and justify why this is the minimum amount by referring to the graph used in part (a).

(c) (2 points) Now suppose that the traveler must pay a toll of one dollar every time he crosses the river with an animal. How can they all make it across the river while paying the least money possible? Say how much money is required, and justify why this is the minimum amount by referring to the graph used in part (a).

4. A binomial tree is a special kind of rooted tree used for various data structures in computer science. A degree $d$ binomial tree can be defined recursively as follows. A degree 0 binomial tree is a single vertex with no edges. A degree $d$ binomial tree has a root vertex with out-degree $d$. The first (that is, leftmost) subtree is a degree $d - 1$ binomial tree. The second (that is, second to left) subtree is a degree $d - 2$ binomial tree. Continue on in this way so that the last (rightmost) subtree is a degree 0 binomial tree.

(a) (2 points) Draw binomial trees of degree 1, 2, 3, and 4.

(b) (3 points) What is the height $h(d)$ of a degree $d$ binomial tree? Prove your result by induction on $d$.

(c) (2 points) Write a recurrence for the number of nodes $n(d)$ in a binomial tree of degree $d$.

(d) (3 points) Use the guess-and-check method to guess a formula for $n(d)$. Prove that your formula holds by induction on $d$.

5. A sorting algorithm that uses a binary tree to sort a list of positive integers $a_1, a_2, \ldots, a_n$ from largest to smallest can be described by the following $n$ steps.

**Step 1: Construct the binary tree. Output the root value.**
The elements $a_1, a_2, \ldots, a_n$ are the leaves of the tree, and we build up the tree one level at a time from there. From left to right, compare the elements in pairs, and put the larger of the two as the parent vertex. Do this at each level until reaching the root, which will be the largest element. Output the value at the root. For example, if the list to be sorted is 22, 8, 14, 17, 3, 9, 27, 11, the tree would look like this:

```
      27
     /  \
   22   27
  /  \
22   17
 /  \
8   17
 /    \
3   9
     /  \
   27  11
```

**Step 2: Recompute labels. Output the root value.**
In the second step, remove the leaf corresponding to that largest element, and replace it with a leaf labeled 0, which is defined to be smaller than all the other list elements. Recompute the labels of all vertices on the path from this 0 to the root. That is, relabel all vertices on the path from the 0 to the root by choosing the larger of the values of their two children. Then the root will become the second-largest element. Output the value at the root. In our example, the tree would now look like this:
Steps 3 through \( n \): Recompute labels. Output the root value.
Repeat the same process as described in step 2. At the end, we will have output the entire list in decreasing order.

(a) (3 points) Trace through the algorithm as applied to the list 17, 4, 1, 5, 13, 10, 14, 6. Show the tree at each step.

(b) (3 points) If \( n \) is a power of two, as in the example, how many comparisons are done at step 1? How many comparisons are done in each of the other steps?

(c) (4 points) If \( n \) is a power of two, as in the example, how many comparisons are done throughout the entire algorithm? What is the order of the algorithm in \( \Theta \) notation?