This homework will not count as one of the graded assignments. Completion, however, is mandatory. Completion, for this assignment, means that you must hand in:

- Problems 1 and 3;
- Problems 4 and 5;
- One of Problems 6 and 7.

Please answer the questions in simple, concise prose and when appropriate, simple, concise pseudocode. Please do not use any outside resources, including your textbook, in answering these questions. Don’t worry if you are uncertain about an answer or if you do not know an answer or if something looks unfamiliar. The goal is to check your familiarity with some of the concepts and tools that we’ll need for the semester. Oh, and some of the problems are fun, too....

**Question 1.** For each of the following, answer with the tightest upper bound from this list: $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(2^n)$.

a) number of leaves in a complete\(^1\) binary tree of height $n$

b) depth of a complete binary tree having $n$-nodes

c) number of edges in an $n$-node tree

d) worst-case run time to sort $n$ items using merge sort

e) number of distinct subsets of a set of $n$ items

f) number of bits needed to represent the positive integer $n$

g) time to find the closest pair of points among $n$ points in Euclidean space by enumerating all possibilities

h) time to insert $n$ items into an empty (binary) heap

i) time to find the second largest number in a set of $n$ (not necessarily sorted) numbers

**Question 2.** How many times can you repeatedly halve 32 before falling below 1? What is the common mathematical name for the operation that computes this value?

In this course, we will not use the syntax of any particular programming language in this course to specify algorithms. Instead, we will use well-written prose and pseudocode—an intuitive set of instructions that are appropriate to the problem at hand. For example, Algorithm 1 below gives some pseudocode for finding the smallest integer in an array.

**Algorithm 1 FINDMIN($A$, $n$)**

**Require:** An array of integers $A$ of length $n \geq 0$. Empty arrays return $+\infty$.

1: $m \leftarrow +\infty$
2: for $i \leftarrow 1$ to $n$ do
3: \hspace{1em} $m \leftarrow \min(m, A[i])$
4: end for
5: return $m$

\(^1\)Complete: Every leaf has same depth and every non-leaf has two children.
**Question 3.** Are you comfortable with the initialization line \( m \leftarrow +\infty \)? How could you most simply modify the code if a "+\infty" or "MAX_INT" value isn’t available in the language you might be using to implement this algorithm?

**Question 4.** Prove by induction that \( \sum_{i=1}^{n} i = \frac{n^2 + n}{2} \).

In the problems below, be careful not to use any assumptions that you cannot prove!

**Question 5.** Prove by induction that any tree with \( n \) vertices has exactly \( n - 1 \) edges.

**Question 6.** A diagonal of a polygon \( P \) is a segment connecting two vertices of \( P \) that is contained completely within \( P \). A triangulation of \( P \) is a set of non-crossing diagonals that divides the interior of \( P \) into triangles. Prove by strong induction that every triangulation of an \( n \)-sided polygon consists of exactly \( n - 3 \) diagonals.

**Question 7.**

a) Write some pseudocode to find the smallest integer value in a binary tree \( T \). Assume that each node \( x \in T \) has three components: left child \( x.left \), right child \( x.right \), and integer value \( x.value \). You can assume that \( x.left \) (respectively \( x.right \)) is NULL if \( x \) doesn’t have a left (respectively right) child.

b) Prove by strong induction that the algorithm you designed for part a) above works correctly.

**Question 8.** Challenge Problem ² You are asked to test the resistance to breakage of identical glass spheres by finding the minimum height, in meters, at which they will break if dropped from that height onto a concrete surface. We will call this height the breakage threshold of the sphere. You can only drop from integral heights: 1 meter, 2 meters, and so on. If you have only one sphere, the only method that is guaranteed to succeed is to drop from 1 meter, then drop from 2 meters, and so on, until the sphere breaks. Thus, if the breakage threshold of the sphere is \( n \), you’ll need to drop the sphere \( n \) times to determine that fact. [Note: spheres are not "weakened" by repeated drops; if a sphere’s breakage threshold is \( k \) meters, no number of drops from lesser heights will change that fact.]

a) Design an algorithm that, given two identical spheres, will find their (common) breakage threshold in less than a linear (in \( n \)) number of drops, and show that your algorithm is correct. What’s "smaller than linear"? Well, any \( n^c \), where \( c < 1 \), for example, or \( \log n \), are all smaller than linear. Hint: Values of \( n \), such as 4, 16, 25, 36, 49, and so on, are helpful cases to consider.

b) For spheres with breakage threshold at most \( n \), what do you think is the minimum number, \( k \), of drops needed to guarantee that you can find the breakage threshold of the spheres? How many spheres would you need to achieve this minimum? A complete answer would include proofs of all claims made.

---

²Challenge problems are optional.