Problem 1. [10 points]
One of the basic motivations behind the Minimum Spanning Tree Problem is the goal of designing a spanning network for a set of nodes with minimum total cost. Here we explore another type of objective: designing a spanning network for which the most expensive edge is as cheap as possible.

Specifically, let $G = (V, E)$ be a connected graph with $n$ vertices, $m$ edges, and positive edge costs that you may assume are all distinct. Let $T = (V, E')$ be a spanning tree of $G$; we define the bottleneck edge of $T$ to be the edge of $T$ with the greatest cost.

A spanning tree $T$ of $G$ is a minimum-bottleneck spanning tree if there is no spanning tree $T'$ of $G$ with a cheaper bottleneck edge.

(a) Is every minimum-bottleneck tree of $G$ a minimum spanning tree of $G$? Prove or give a counterexample.
(b) Is every minimum spanning tree of $G$ a minimum-bottleneck tree of $G$? Prove or give a counterexample.

Problem 2. [10 points - Proof problem]
Your friend is working as a camp counselor, and he is in charge of organizing activities for a set of junior-high-school-age campers. One of his plans is the following mini-triathlon exercise: each contestant must swim 20 laps of a pool, then bike 10 miles, then run 3 miles.

The plan is to send the contestants out in a staggered fashion, via the following rule: the contestants must use the pool one at a time. In other words, first one contestant swims the 20 laps, gets out, and starts biking. As soon as this first person is out of the pool, a second contestant begins swimming the 20 laps; as soon as he or she is out and starts biking, a third contestant begins swimming . . . and so on.

Each contestant has a projected swimming time (the expected time it will take him or her to complete the 20 laps), a projected biking time (the expected time it will take him or her to complete the 10 miles of bicycling), and a projected running time (the time it will take him or her to complete the 3 miles of running).

Your friend wants to decide on a schedule for the triathlon: an order in which to sequence the starts of the contestants. Let’s say that the completion time of a schedule is the earliest time at which all contestants will be finished with all three legs of the triathlon, assuming they each spend exactly their projected swimming, biking, and running times on the three parts.

Note that participants can bike and run simultaneously, but at most one person can be in the pool at any time.

(a) What are the inputs and goal of this problem? Try to formalize your notations, to help you later with the proof.
(b) What’s the best order for sending people out, if one wants the whole competition to be over as early as possible?
(c) Prove that your proposed order is optimal.
**Problem 3.** [5 points]
Consider the Minimum Spanning Tree Problem on an undirected graph \( G = (V, E) \), with a cost \( c_e \geq 0 \) on each edge, where the costs may not all be different. If the costs are not all distinct, there can in general be many distinct minimum-cost solutions.

Suppose we are given a spanning tree \( T \subset E \) with the guarantee that for every \( e \in T \), \( e \) belongs to some minimum-cost spanning tree in \( G \). Can we conclude that \( T \) itself must be a minimum-cost spanning tree in \( G \)? Give a proof or a counterexample with explanation.

**Problem 4.** [10 points]
Consider an algorithm whose running time \( T(n) \) on an input of size \( n \) satisfies the following recurrence:
\[
T(n) = aT\left(\frac{n}{2}\right) + cn
\]
where we assume the recurrence holds when \( n > 1 \), and that \( T(1) = c \).

(a) How many nodes are there at level \( i \) of the recursion tree?
(b) What is the input size for a problem at level \( i \) of the recursion tree?
(c) How much work is done in a single function call at level \( i \) of the recursion tree? (Just as in class, count only the work done in the function itself, excluding recursive calls.)
(d) What is the total work done at level \( i \) of the recursion tree?
(e) How many levels are in the recursion tree?

**Problem 5.** [15 points]
Suppose you're consulting for a bank that’s concerned about fraud detection, and they come to you with the following problem. They have a collection of \( n \) bank cards that they’ve confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we’ll say that two bank cards are equivalent if they correspond to the same account.

It’s very difficult to read the account number off a bank card directly, but the bank has a high-tech “equivalence tester” that takes two bank cards and, after performing some computations, determines whether they are equivalent. Their question is the following: among the collection of \( n \) cards, is there a set of more than \( n/2 \) of them that are all equivalent to one another?

Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only \( O(n \log n) \) invocations of the equivalence tester. Write the algorithm, and explain your reasoning.