Problem 1

Consider the union-find problem, but with an additional operation *DELETE(x)* that removes element x from its current set and places it in a set by itself. Note that later union operations might apply to this new set. The goal of this question is to modify the solutions given in class so that a sequence of m UNION, FIND and DELETE operations on n elements takes $O(m \log^* n)$ time.

(a) A first idea is to actually remove the tree node associated with x, and link its subtrees back to the root. Show that this is a bad approach.

(b) Give a solution using the idea of "delayed deletion," where we do not actually remove the tree node of the element that should be deleted, and instead we just mark it dead. Then, when the number of dead nodes goes above half the total number of nodes we do a big cleanup and remove all the dead nodes.

Problem 2

In this problem we will work out some splaying counterexamples;

(a) Single rotations "don't work" for splay trees. To show this, consider a degenerate n-node "linked-list shaped" binary tree where each node's right child is empty. Suppose that the (only) leaf is splayed to the root by *single* rotations. Show the structure of the tree after this splay. Generalize this to argue that there is a sequence of $n/2$ splays that each take at least $n/2$ work.

(b) Now from the same starting tree, show the final structure after splaying the leaf with (zig-zig) double rotations. Explain how this splay has made much more progress than single rotations in "improving" the tree.

(c) Given the theorem about access time in splay trees, it is tempting to conjecture that splaying does not create trees in which it would take a long time to find an item. Show this conjecture is false by showing that for large enough n, it is possible to restructure any binary tree on n nodes into any other binary tree on n nodes by a sequence of splay requests. Conclude that it is possible to make a sequence of requests that cause the splay tree to achieve any desired shape.

Hint: start by showing how you can use splay requests to make a specified node into a leaf, then recurse.

Problem 3

1. Show that given n numbers in $[0, 1]$ it is impossible to estimate the value of the median within say 1.1 multiplicative approximation factor with $o(n)$ samples.

   Hint: to show an impossibility result you show two different sets of n numbers that have very different medians but which generate, with high probability, identical samples of size $o(n)$.

2. Now calculate the number of samples needed (as a function of t) so that the following is true: with high probability, the median of the sample has at least $n/2 - t$ numbers (from the given set of n numbers) less than it and at least $n/2 - t$ numbers (from the given set of n numbers) more than it.

PROBLEM 4

The simplest model for a random graph consists of n vertices, and tossing a fair coin for each pair $\{i, j\}$) to decide whether this edge should be present in the graph. Call this model $G(n, 1/2)$. A triangle is a set of 3 vertices with an edge between each pair.

1. What is the expected number of triangles?
2. What is the variance?
3. Show that the number of triangles is concentrated around the expectation and give an expression for the bound in the decay of probability.

PROBLEM 5

Consider again the experiment in which we toss m labeled balls at random into n labeled bins, and let the random variable X be the number of empty bins. We have seen that $\mathbb{E}[X] = n \cdot \left(1 - \frac{1}{n}\right)^m$.

(a) By writing $X = \sum_i X_i$ for suitable random variables X_i, show how to derive the following exact formula for the variance of X:

$$Var[X] = n \cdot \left(1 - \frac{1}{n}\right)^m + n(n-1) \cdot \left(1 - \frac{2}{n}\right)^m - n^2 \cdot \left(1 - \frac{1}{n}\right)^{2m}$$

(b) What is the asymptotic value (as $n \to \infty$) of $Var[X]$ in the cases $m = n$ and $m = n \ln(n)$?

(c) When throwing n balls into n bins, what is the expected number of bins with *exactly* one ball? Compute an exact formula and its asymptotic approximation.

PROBLEM 6

Let us consider the coupon collector problem: we toss $m = n \log n + cn$ balls into n bins, where c is a constant, and we are interested in the probability that there is no empty bin. We saw in class that

$$Pr[\text{some bin is empty}] \le n \cdot \left(1 - \frac{1}{n}\right)^m \sim n \cdot \frac{1}{e^{m/n}} = \frac{1}{e^c}.$$

Prove that

$$Pr[\text{some bin is empty}] = \Omega\left(\frac{1}{e^c} - \frac{1}{2e^{2c}}\right).$$