1. In our analysis of QuickSort in class, we showed that if $M_j$ is the sum of squares of the problem sizes at level $j$ of the recursion tree, then $E[M_j] \leq (2/3)^j n^2$. Use this fact to show that the expected depth of the recursion tree is $O(\log n)$.

2. You are given $n$ numbers, each of bit length $\ell$. Using Karsuba’s algorithm as a subroutine, show how to compute the product of these numbers in time $O((n\ell)^{\log_2 3})$.

3. Suppose we have an abstract data type that represents sets of items. There are three operations: init$(S)$, which initializes an empty set $S$, copy$(S, T)$, which creates a set $S$ that is a copy of $T$, and insert$(S, a)$, which inserts the item $a$ into the set $S$.

Design an algorithm for the following problem. The input is a list of $n$ distinct items $a_1, \ldots, a_n$. The output is a list of $n$ sets $S_1, \ldots, S_n$, where $S_i = \{a_1, \ldots, a_n\} \setminus \{a_i\}$.

Your algorithm is only allowed to use the operations init, copy, and insert, described above, and should use $O(n \log n)$ such operations.

4. Consider the following recursive, probabilistic algorithm $A$, which takes as input a finite set $S$ of items.

   Algorithm $A(S)$:
   
   | if $|S| \leq 1$
   | return $(0, 0, 0)$
   | else
   | let $R$ be a randomly chosen subset of $S$
   | $(v_1, v_2, v_3) \leftarrow A(R)$
   | $(w_1, w_2, w_3) \leftarrow A(S \setminus R)$
   | return $(\max\{v_1, w_1\} + 1, v_2 + w_2 + |S|, v_3 + w_3 + |S|^2)$

   Let $(u_1, u_2, u_3)$ denote the output of $A$ on input $S$, and let $n := |S|$. Show that $E[u_1] = O(\log n)$, $E[u_2] = O(n \log n)$, and $E[u_3] = O(n^2)$.

5. (a) It takes $n - 1$ comparisons to find the largest of $n$ numbers. Why?

   (b) It takes only $[3n/2]$ comparisons to find both the largest and smallest of $n$ numbers. How can this be done?

   (c) Show that any comparison-based algorithm for finding both the largest and smallest of $n$ items must make at least $[3n/2] - 2$ comparisons in the worst case.

6. You are given $d$ sequences of items such that each sequence is already sorted, and there is a total of $n$ items. Design an $O(n \log d)$ algorithm to merge all the sequences into one sorted sequence.

7. You are given $n$ items, and you want to determine if there are any duplicates among them.

   (a) Show how to solve this problem in expected linear time using universal hashing (make any reasonable assumptions you like about the hash functions).

   (b) Show how to solve this problem using only comparisons in time $O(n \log n)$.

   (c) Prove an $\Omega(n \log n)$ time lower bound for this problem in the comparison model.

   You may wish to proceed as follows.

   - For a permutation $\pi$ on $\{1, \ldots, n\}$, let $v(\pi)$ denote the leaf in the decision tree reached on any input $(a_1, \ldots, a_n)$ satisfying $a_{\pi(1)} < a_{\pi(2)} < \cdots < a_{\pi(n)}$.

   - Show that $v(\pi) \neq v(\pi')$ if $\pi \neq \pi'$.

     Hint: for any permutation $\pi$, the comparisons along the path from the root to $v(\pi)$ naturally define a partial ordering on $\{1, \ldots, n\}$, and any partial ordering can always be extended to (at least) one total ordering.