Problem 1. Assume, we pick a pivot for Quicksort that always gives us a 9-to-1 split.

1. Write the recurrence equation.
2. What is the height of the recursion tree?
3. Solve the recurrence equation using constructive induction.

Problem 2. The Quicksort algorithm described in class, is called recursively until a base case of leaves (elements) of size 1. However, on small input sequences, insertion sort works really well. For this question you would modify the Quicksort algorithm so that after some threshold, m, the sorting is performed by using insertion sort (without sentinel) instead of Quicksort all the way to the leaves. In other words, instead of going all the way up to $r - p > 0$ in the algorithm, you would use Quicksort up to some threshold, m, such that $r - p > m$ and insertion sort (without sentinel) after that.

1. Write the pseudo code for the modified algorithm.
2. Write the recurrence equation for an average case of comparisons for this algorithm.
3. What is the average number of comparisons? Show your work.

Problem 3. You are given twelve coins that are identical in appearance ; either all are genuine or exactly one of them is fake. We do not know whether the fake coin is lighter or heavier than the genuine ones. You have a two-pan balance scale without weights. Each weighing would yield either the coins are $=, <,$ or, $>$ each other in weight. You may assume $<$ means lighter in weight and $>$ means heavier in weight. You are required to find whether all the coins are genuine and, if not, find the fake coin and establish whether it is lighter or heavier than the genuine ones. Use decision tree (display it) model to solve the problem with the minimum number of weighings. Find the lower bound on the number of weighings using a proper inequality just like we did in class.

Problem 4. Suppose I want to find the k-th largest number in an array of size n. I could sort the array and look at the k-th value from the end. This could be an $O(nlg\,n)$ runtime algorithm. We would like to improve it. Write an algorithm in English or in pseudo-code to find the k-th largest value in $O(klg\,n)$ runtime for large k. As an example, the 3^{rd} largest value in, $A = [4, 2, 3, 1, 6, 8]$, is 4.

Note: For smaller values of k, the runtime is obviously linear. For this problem we are seeking the runtime when k is closer to n than to 1, such that, $1 << k \leq n$.