1 (100 PTS.) Find kth smallest number.

This question asks you to design and analyze a *randomized incremental* algorithm to select the kth smallest element from a given set of n elements (from a universe with a linear order).

In an *incremental* algorithm, the input consists of a sequence of elements $x_1, x_2, \ldots, x_n$. After any prefix $x_1, \ldots, x_{i-1}$ has been considered, the algorithm has computed the kth smallest element in $x_1, \ldots, x_{i-1}$ (which is undefined if $i \leq k$), or if appropriate, some other invariant from which the kth smallest element could be determined. This invariant is updated as the next element x_i is considered.

Any incremental algorithm can be *randomized* by first randomly permuting the input sequence, with each permutation equally likely.

1.A. (5 PTS.) Describe an incremental algorithm for computing the kth smallest element.

1.B. (5 PTS.) How many comparisons does your algorithm perform in the worst case?

1.C. (10 PTS.) What is the expected number (over all permutations) of comparisons performed by the randomized version of your algorithm? (Hint: When considering x_i, what is the probability that x_i is smaller than the kth smallest so far?) You should aim for a bound of at most $n + O(k \log(n/k))$. Revise (A) if necessary in order to achieve this.

2 (100 PTS.) Majority tree

Consider a uniform rooted tree of height h (every leaf is at distance h from the root). The root, as well as any internal node, has 3 children. Each leaf has a boolean value associated with it. Each internal node returns the value returned by the majority of its children. The evaluation problem consists of determining the value of the root; at each step, an algorithm can choose one leaf whose value it wishes to read.

2.A. Show that for any deterministic algorithm, there is an instance (a set of boolean values for the leaves) that forces it to read all $n = 3^h$ leaves. (hint: Consider an adversary argument, where you provide the algorithm with the minimal amount of information as it request bits from you. In particular, one can devise such an adversary algorithm.).

2.B. (10 PTS.) Consider the recursive randomized algorithm that evaluates two subtrees of the root chosen at random. If the values returned disagree, it proceeds to evaluate the third sub-tree. If they agree, it returns the value they agree on.

Write an explicit exact formula for the expected number of leaves being read, for a tree of height $h = 1$, and height $h = 2$.

2.C. (5 PTS.) Using (B), prove that the expected number of leaves read by the algorithm on any instance is at most $n^{0.9}$.

3 (100 PTS.) Sorting Random Numbers

Suppose we pick a real number x_i at random (uniformly) from the unit interval, for $i = 1, \ldots, n$.

3.A. (5 PTS.) Describe an algorithm with an expected linear running time that sorts $x_1, \ldots, x_n$.

To make this question more interesting, assume that we are going to use some standard sorting algorithm instead (say merge sort), which compares the numbers directly. The binary representation of each x_i can be generated as a potentially infinite series of bits that are the outcome of unbiased coin flips. The idea is to generate only as many bits in this sequence as is necessary for resolving comparisons between different numbers as we sort them. Suppose we have only generated some prefixes of the binary representations of the numbers. Now, when comparing two numbers x_i and x_j, if their current partial binary representation

can resolve the comparison, then we are done. Otherwise, the have the same partial binary representations (upto the length of the shorter of the two) and we keep generating more bits for each until they first differ.

3.B. Compute a tight upper bound on the expected number of coin flips or random bits needed for a single comparison.

3.C. Generating bits one at a time like this is probably a bad idea in practice. Give a more practical scheme that generates the numbers in advance, using a small number of random bits, given an upper bound n on the input size. Describe a scheme that works correctly with probability $\geq 1 - n^{-c}$, where c is a prespecified constant.