CS315, Fall 2016
Homework 6
due: Oct. 26

Problem 1. Design an algorithm that, given two lists of integers, creates a list consisting of those integers that appear in both lists (each integer on the final list should appear only once). Describe your algorithm in terms of a high-level pseudocode focusing on main algorithmic tasks and not on low-level details.

Analyze the running time of your algorithm. You will get full credit only if your algorithm achieves an asymptotically better worst-case performance than $\Theta(n^2)$, where n is the sum of the lengths of the two input lists.

Problem 2. Give a high-level pseudocode for an algorithm that, given a list of n integers from the set $\{0, 1, \ldots, k-1\}$, preprocesses its input to extract and store information that makes it possible to answer any query asking how many of the n integers fall in the range $[a..b]$ (with a and b being input parameters to the query) in $O(1)$ time. Explain how yout algorithm works.

The preprocessing time should be $O(n + k)$ in the worst case. Provide an argument showing that your preprocessing algorithm meets that bound.

Problem 3. Describe an algorithm (high-level pseudocode) to sort a list of n integers, each in the range $[0..n^2 - 1]$, in $O(n)$ time. Justify the correctness and the running time of your algorithm.

Generalize to an arbitrary *constant integer* k. That is, describe an algorithm to sort a list of n integers, each in the range $[0..n^k - 1]$, in $O(n)$ time.

Problem 4. A sequence of integers $x_1, x_2, \ldots, x_n$ is *unimodal* if for some $1 \leq i \leq n$ we have $x_1 < x_2 < \ldots < x_i$ and $x_i > x_{i+1} > \ldots > x_n$. For instance, the sequence

$$2, 4, 5, 8, 9, 16, 28, 23, 22, 17, 15, 8, 6, 3, 0$$

is unimodal.

Describe (in high-level pseudocode) an algorithm to find the maximum element in a unimodal sequence of integers $x_1, x_2, \ldots, x_n$. The running time should be $O(\log n)$. Show that your algorithm meets the bound.

Problem 5. Describe an algorithm to merge k sorted lists containing altogether n elements into one sorted list. Give a pseudo-code. The algorithm must run in time $O(n \log k)$. Show that your algorithm meets the bound.

Hint: Build on the ideas behind the mergesort algorithm.

Solutions must be typed (wordprocessed), and submitted as a single pdf file to the canvas