cs473: Algorithms Assigned: *Thu., Sep. 17, 2020*

Updated: *Wed., Sep. 23, 2020*

Problem Set #3 (v2)

Prof. Michael A. Forbes Due: *Thu., Sep. 24, 2020 (5:00pm)*

Some reminders about logistics.

- **Submission Policy:** See the course webpage for how to submit your pset via gradescope.
- **Collaboration Policy:** For this problem set you are allowed to work in groups of up to three. Only one copy should be submitted per group on gradescope. See the course webpage for more details.
- **Late Policy:** Late psets are not accepted. Instead, we will drop several of your lowest pset problem scores; see the course webpage for more details.

All problems are of equal value.

1. Erickson Chapter 9, #8 (`http://jeffe.cs.illinois.edu/teaching/algorithms/book/09-apsp.pdf`).

2. Consider the *subset-sum problem*: given n positive integers $a_1, \dots, a_n > 0$ and a target integer T, decide whether there exists a subset $S \subseteq \{a_1, \dots, a_n\}$ that sums to exactly T. It is easy to adapt the dynamic program presented in lecture for the knapsack problem to solve this problem in $O(nT)$-time and $O(nT)$ space, even if we want to output the desired subset (or otherwise declare that no subset exists).

   In lecture we saw a space-saving trick for computing not only the optimal edit-distance between two strings in smaller space, but *also* to output an optimal alignment within the same space bound. Apply this trick to the subset-sum problem, to obtain a $O(nT)$-time, $O(n+T)$-space algorithm that outputs a subset summing to exactly T (or otherwise declares that no subsets exists).

3. In the selection problem we are given an array A of n numbers (not necessarily sorted) and an integer k, and the goal is to output the rank k element of A. Consider a randomized algorithm where we pick a number x uniformly at random from A and use it as a pivot as in quick sort to partition A into numbers less than equal to x and numbers greater than x. The algorithm recurses on one of these arrays depending on k and the size of the two arrays.

   (a) Write down a description of randomized quick selection in pseudocode. Show that the expected depth of the recursion of randomized quick selection is $O(\log n)$, and that the expected running time is $O(n)$.

   Hint: Write a recurrence for the depth of the recursion, as well as for the runtime.

   Remark: This algorithm has the advantage of being quite simple when compared to deterministic algorithms with the same runtime, such as algorithms that compute median of medians.

   (b) Let $A_1, A_2, \dots, A_h$ be h *sorted* arrays where A_i has n_i elements, and let $n = \sum_{i=1}^{h} n_i$. Assume that the arrays have distinct elements. Describe a randomized algorithm that given integer k finds the k-th smallest element in the combined set of arrays in $O(h \log^2 n)$ expected time.

Hint: Adapt the randomized quick selection algorithm and analysis from the first part.

Remark: It is possible to deterministically achieve a time bound of $O(h \log n)$.