COMP 340 **Analysis of Algorithms**

Problem Set 3

Assigned: February 21, 2020 *Due: March 02, 2020*

Always provide explanations and show as much work as possible. Solutions to TADM's odd-numbered exercises are available at `http://www.algorist.com/algowiki/index.php/The_Algorithms_Design_Manual_(Second_Edition)`. Designing algorithms often involves some creativity, so start early and work consistently. If you are stuck on a problem, move on and come back to it. If you get stuck again, discuss it with your classmates and/or come see me in office hours.

1. In the **reduction** design technique[1], you solve a problem by seeing that it is actually some other (solved) problem expressed in a different way. You use the algorithm for the solved problem, and perhaps a few other simple steps, to get the solution to the original problem.

   In the order statistics problem you are given a sequence of n (unsorted) values and an integer $k \in [1, n]$ and must return the kth-smallest value. Give an algorithm for the order statistics problem by reduction to sorting.

2. Consider the following knapsack variants:

   (a) Given a set of integers $S = \{s_1, s_2, \ldots, s_n\}$ and a positive integer k, return a subset whose sum is *at least* k, or the empty set if no such set exists.

   (b) Given a set of integer $S = \{s_1, s_2, \ldots, s_n\}$ and a positive integer k, return a subset that sums to the highest possible value $\leq k$.

   You are given an algorithm A that solves the first variant in time $T_A(n)$. Use this algorithm as a black box[2] to solve the second variant in time $O(T_A(n) \cdot \log k)$. Analyze its running time and briefly explain (a formal proof is not necessary) its correctness.

3. The following questions involve the "Stock market" problem and comes from a parallel algorithms course at WUSTL[3]:

   > The problem with the stock market is that, while it is possible to make a great deal of money buying and selling stocks, it's easy to lose even more. The long-standing—if somewhat unhelpful—maxim to make more money than you lose is "buy low, sell high."
   >
   > The stock market problem is finding the best opportunity to follow this advice: for any sequence of integer prices, where the index in the sequence represents time, find maximum jump from an earlier price to a later price. For example, if the sequence of prices was
   >
   > $$\langle 40, 20, 0, 0, 0, 1, 3, 3, 0, 0, 9, 21 \rangle$$
   >
   > then the maximum jump is 21, which happens between the price at time 2 and time 11. More formally, the stock market problem is to compute
   >
   > $$\max\{s_j - s_i \mid 0 \leq i \leq j < |s|\}$$
   >
   > Note that this maximum is only well defined if there is at least one element in s.

[1] Not to be confused with the `reduce` algorithm.

[2] In other words, you may call a procedure/function $A(S, \ell)$ that runs algorithm A on S with $k = \ell$. You may call A as many times as you wish (provided you satisfy the time bound) and choose ℓ each time.

[3] Washington University in St. Louis

(a) Give a brute force algorithm for the Stock market problem. Analyze the runtime.

(b) Give a *parallel* divide-and-conquer algorithm for this problem.

(c) Analyze the work AND span of your divide and conquer algorithm.

(d) Prove your divide and conquer algorithm is correct.

4. You are given a sorted array A of n numbers. Give a *parallel* algorithm to remove duplicates — copy the unique numbers to a new array of size n', where n' is the number of unique numbers in A. Analyze the work and span of your algorithm. *Hint*: Use prefix sums.

   For full credit your algorithm should have work $O(n)$ and span $O(\log n)$ assuming a `scan` (prefix sums) algorithm with span $O(\log n)$.