Assignment 1
CSC373 Winter 2009
Due Date: Feb 5th, Noon, BA2220 Drop Box 7

Problem 1

Given two sorted arrays, A[1…n] and B[1…k], we would like to find the ith largest element in the union of the two arrays.

a) Design an algorithm to solve the problem. Your description should be accurate and concise—pseudo-code is recommended. Justify why it is correct.

b) Describe the running time of your algorithm using a recurrence relation, and solve it. (NOTE: To get more than half the points, your algorithm should run in O(log(n) + log(k)) time—otherwise you could get at most half the points for this problem)

Problem 2

You are given the array A[1…n]. The elements of this array are either positive or negative integers. You are asked to find an interval of this array A[1< i…j < n] such that the sum of the elements in the interval $\sum_{k=i\ldots j} A[k]$ is maximum.

a. How many intervals are there?

b. Design a simple, non-recursive algorithm that runs in time at most equal to the total number of intervals.

c. Design a recursive, divide and conquer algorithm for this problem that runs in time O(n log n). (Hint: Given the array and a position in the array, the optimal interval either contains that position, or is entirely contained within the left sub-array, or is entirely contained within the right sub-array)

Problem 3

Your friend is working as a camp counselor who is in charge of organizing activities for a set of junior-high-school-age campers. One of the plans is the following mini-triathlon exercise: each contestant must swim 20 laps of a pool, then bike 10 kilometers, then run 3 kilometers. The plan is to send the contestants out in a staggered fashion, via the following rule: the contestants must use the pool one at a time. In other words, first one contestant swims the 20 laps, gets out, and starts biking. As soon as this first person is out of the pool, a second contestant begins swimming the 20 laps; as soon as he or she is out and starts biking, a third contestant begins swimming, and so on. Note that participants can bike simultaneously and can run simultaneously, but at most one person can be in the pool at any time; also, contestants must complete the events in the same order—swimming, biking, running—to avoid giving anyone an unfair advantage.

Each contestant has a projected swimming time (the expected time it will take him or her to complete the 20 laps), a projected biking time (the expected time it will take him or her to complete the 10kms of bicycling), and a projected running time (the expected time it will take him or her to complete the 3kms of running).

Your friend wants to decide on a schedule for the triathlon: an order in which to schedule the contestants. Let's say that the completion time of a schedule is the earliest time at which all contestants will be finished with all three legs of the triathlon, assuming they each spend exactly their projected swimming, biking, and running times on the three parts.

You are asked to give an algorithm that specifies the order in which to send people out in a way that minimizes the completion time.

Consider three different greedy strategies:

- Always send out the person with the longest total projected time first.
- Always send out the person with the shortest projected swimming time.
- Always send out the person whose sum projected time for biking and running is biggest.

For each strategy, either prove or disprove that it always produces the optimal solution.

Problem 4

Consider the problem of making change for n cents using as few coins as possible.

a) Describe a Greedy Algorithm to make change consisting of quarters, dimes, nickels and pennies. Prove that your algorithm yields an optimum solution.

b) Is the optimal solution always unique? (prove or disprove)

c) Give a set of coin denominations for which your Greedy Algorithm does not yield an optimum solution. Your set should include a penny, so that there is a solution for every value of n.

Problem 5

When considering Huffman Codes in class, we emphasized that the code has to be prefix-free and minimal: that is, no code for one symbol can be a prefix for the code for another symbol, and no code for one symbol could be made shorter while maintaining the prefix-free property. These prefix codes were represented by full binary trees, with each symbol corresponding to a leaf. We consider two codes distinct if there is at least one symbol which is coded by a different number of bits.

a. How many possible distinct codes are there for an alphabet of 3 symbols? What about 4 and 5? Justify your answer.

b. Write down a recurrence that describes the total number of distinct codes for an alphabet of n symbols.

c. Prove that the number of distinct prefix codes is $2^{\Omega(n)}$