Assignment 2

COMP3121/9101 21T3

Released September 29, due October 13

In this assignment we apply divide and conquer (including multiplication of large integers and the Fast Fourier Transform) and the greedy method. There are *five problems*, for a total of 100 marks.

Your solutions must be typed, machine readable PDF files. *All submissions will be checked for plagiarism!*

For each question requiring you to design an algorithm, you *must* justify the correctness of your algorithm. If a time bound is specified in the question, you also *must* argue that your algorithm meets this time bound.

Partial credit will be awarded for progress towards a solution.

1. (20 points) You are given n stacks of blocks. The ith stack contains $h_i > 0$ identical blocks. You are also able to move any number of blocks from the ith stack to the $(i+1)$th stack. You want to know if the sizes of the stacks can be made *strictly* increasing. For example $\langle 1, 3, 6, 8\rangle$ is acceptable, but $\langle 1, 4, 4, 7\rangle$ is not.

   Design an $O(n)$ algorithm that determines whether it is possible to make the sizes of the stacks strictly increasing.

2. (20 points) Alice has n tasks to do, the ith of which is due by the day d_i. She can work on one task each day, and will complete each task in one day. Morever, Alice is a severe procrastinator and wants to accomplish every task as close as possible to its due date. If Alice finishes the ith task on day j, her rage will increase by $d_i - j$.

   Design an $O(n \log n)$ algorithm that determines whether all tasks can be completed by their deadlines, and if so, outputs the minimum total rage that Alice can accumulate.

3. (20 points) Define the *separation* of an array of integers to be the smallest difference between any two integers in the array.

   You are given an array A of n distinct positive integers, each no larger than m. For a given positive integer k satisfying $2 \leq k \leq n$, you wish to select a length k subarray of A with the largest possible separation. This subarray need not be contiguous.

   Design an $O(n \log m)$ algorithm to select such a subarray.

4. (20 points) You are given a set of real numbers $S = \{t_1, t_2, \ldots, t_n\}$, where $n = |S|$ is a positive integer. Your task is to construct a polynomial P of degree n and leading coefficient 1, i.e.

$$P(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}\,x^{n-2} + \ldots + a_2\,x^2 + a_1\,x + a_0,$$

such that $P(t_1) = P(t_2) = \ldots = P(t_n)$.

Design an $O(n \log^2 n)$ algorithm to construct such a polynomial and evaluate its coefficients.

5. (20 points) Aleks received an offer from UNSW and he wants to graudate as soon as possible. His program requires him to complete n courses in an order of his choice. The courses are labelled $1, 2, \ldots, n$, where course i takes t_i weeks to complete.

   However, some courses are extensions of other courses. If course j is an extension of course i, then a student who has already completed course i can complete course j in fewer than t_j weeks.

   Aleks provides you with a set S consisting of m ordered pairs of courses, as well as a helper function f which he has conveniently produced from the UNSW handbook. For a pair $(i, j) \in S$, $f(i, j)$ calculates in *constant time* the number of weeks required to complete course j if course i has already been completed. Note that $f(i, j) \leq t_j$, with equality if $i = j$ or if course j is not an extension of course i. Note also that the function f only accepts pairs from S.

   Design an $O((n + m) \log(n + m))$ time algorithm that finds the minimum number of weeks required to complete all n courses.