Assignment 2 - Hints and Clarifications

COMP3121/9101 21T3

Released September 29, due October 13

This document provides some hints to help you solve the problems in Assignment 2. You are *not* required to follow these hints, and there may be alternate solutions which are equally correct.

Also included are the clarifications listed in the Assignment 2 FAQ on the Ed forum. Further clarifications may be added after this document is released.

General clarifications:

- **Am I required to use divide and conquer or greedy for each problem, or can I use other methods?**

  All questions can be done with divide and conquer or greedy. We recommend sticking within those topics, but if you have a valid solution that doesn't use them it won't be penalised.

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.

   Clarifications:

   - **What is given as input?**

     An array of size n containing the number of blocks in each stack, i.e. $A[i] = h_i$, the height of stack i.

   - **What should I return?**

     YES or NO. You do *not* need to give the moves required, or the final sequence.

   - **Can a stack ever be empty?**

     No, not in the final sequence or any intermediate step.

   Hint: Use the greedy method. What is the fewest number of blocks required in the ith stack of a strictly increasing sequence?

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.

   Clarifications:

   - **What is given as input?**
     An array of size n containing the deadlines of each task, i.e. $A[i] = d_i$, the deadline of task i.
   - **Can multiple tasks be due on the same day?**
     Yes.
   - **Does Alice have to do a task every day?**
     No.

   Hint: Use the greedy method. If there is a unique task with the latest deadline, when should Alice do that task? What if there are two or more tasks with the equal latest deadline?

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.

   Clarifications:

   - **What is given as input?**
     The array A of size n, as well as the integers m and k.
   - **What should I return?**
     You can return either the indices or the values of the selected elements.
   - **Does non contiguous subarray just mean subset?**
     Yes.
   - **What if multiple subarrays of size k have the largest separation?**
     Pick any.

   Hint: For some fixed s, can you determine whether there is a subarray of length k and separation at least s in $O(n)$ time?

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.

Clarifications:

- **What is given as input?**
  The set S of cardinality n.

Hint: Use divide and conquer.

5. (20 points) Aleks received an offer from UNSW and he wants to graduate 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. Aleks gives you these values in an array A.

However, some pairs of courses *overlap*. If courses i and j overlap, then a student who has already completed either course can complete the other in a number of weeks less than both t_i and t_j.

Using the UNSW handbook, Aleks has produced another array B with m entries. Each entry consists of an *unordered* pair of *distinct* courses which overlap (say $p = \{i, j\}$), as well as the number of weeks t_p required to complete either course if the other has already been completed. For each such pair, you are guaranteed that $t_p < \min(t_i, t_j)$.

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

Clarifications:

- **What is given as input?**
  An array A of size n containing the number of weeks required to complete each course, i.e. $A[i] = t_i$, the time required to finish course i.
  An array B of size m, each entry comprised of an *unordered* pair of *distinct overlapping* courses and the time taken to complete either of them if the other is already completed.
- **Can Aleks do multiple courses simultaneously?**
  No.
- **Do courses overlap each other? For example if i overlaps with j (i.e. i can be completed faster if j is already completed) does j also overlap with i in the same way?**
  Yes.

- **Can a course overlap with multiple other courses?**

  Yes, but you can only save time due to one of them. For example, if k overlaps with i and j, then the time taken to do course k is either t_k, $t_{\{i,k\}}$ or $t_{\{j,k\}}$, not any combination of them.

- **Can multiple courses overlap with the same course?**

  Yes.

- **What pairs can appear in B?**

  Each pair $\{i, j\}$ contained in an entry of B are distinct courses (i.e. $i \neq j$). All of these pairs are overlapping courses, so $t_{\{i,j\}} < t_i$ and $t_{\{i,j\}} < t_j$. Note that there are no longer any extraneous pairs provided.

Hint: Construct a graph where each course is represented by a vertex, and each pair of overlapping courses is represented by an edge. How can you account for the times t_i and $t_{\{i,j\}}$?