CPSC 320: Intermediate Algorithm Design	2016 Summer Term 1

Tutorial 8

Prof. Nick Harvey — *University of British Columbia*

Recurrences describing the value of optimal solutions to subproblems — Dynamic Programming

1. Let A be an array of n distinct integers. In this problem, we are interested in finding the *longest increasing subsequence* of A. That is, we want to find elements $A[i_1], A[i_2], ..., A[i_t]$ such that

$$i_1 < i_2 < ... < i_t$$
$$A[i_1] < A[i_2] < ... < A[i_t]$$

   and t is as big as possible.

   For instance, consider the array

$$A = (1, 9, 17, 5, 8, 6, 4, 7, 12, 3).$$

   In this example, $(1, 9, 17)$ and $(1, 5, 6, 7, 12)$ are two increasing subsequences. Note that the subsequence given by the greedy algorithm is $(1, 9, 17)$, which is not the longest one.

   In order to find the longest increasing subsequence in the array, you can compute for each position i the length $L[i]$ of the longest subsequence that ends with element $A[i]$.

   (a) Give a recurrence relation that expresses $L[i]$ as a function of $L[j]$ for values of j that are smaller than i. **Hint:** you need to consider the position of the previous element of the longest increasing subsequence.

   (b) Write pseudo-code for an algorithm that finds the longest increasing subsequence of an array with n elements.

2. You are managing a consulting team of computer experts, and each week you have to choose a job for them to undertake. Now, as you can well imagine, the set of possible jobs is divided into those that are *low-stress* (e.g., setting up a Web site for a class at the local elementary school) and those that are *high-stress* (e.g., protecting the nation's most valuable secrets). The basic question, each week, is whether to take on a low-stress job or a high-stress job.

   If you select a low-stress job for your team in week i, then you get a revenue of $L_i > 0$ dollars; if you select a high-stress job, you get a revenue of $H_i > 0$ dollars. The catch, however, is that in order for the team to take on a high-stress job in week i, it is required that they do no job (of either type) in week $i - 1$; they need a full week of preparation to get ready for the crushing stress level. On the other hand, it is okay for them to take a low-stress job in week i even if they have done a job (of either type) in week $i - 1$.

   So, given a sequence of n weeks, a *plan* is specified by a choice of "low-stress", "high-stress", or "none" for each of the n weeks, with the property if that "high-stress" is chosen for week $i > 1$ then "none" has to be chosen for week $i - 1$ (choosing a high-stress job in week 1 is acceptable). The *value* of the plan is determined in the natural way: for each i, you add L_i to the value if you choose "low-stress" in week i, you add H_i to the value if you choose "high-stress" in week i, and you add 0 if you choose "none" in week i.

   The problem: Given sets of values $L_1, L_2, \ldots, L_n$ and $H_1, H_2, \ldots, H_n$, find a plan of maximum value (such a plan will be called *optimal*).

   Example: Suppose $n = 4$ and the values of L_i and H_i are given by the table:

	Week 1	Week 2	Week 3	Week 4
L_i	10	1	10	10
H_i	20	50	20	15

Then the plan of maximum value would be to choose "none" in week 1, a "high-stress" job in week 2, and "low-stress" jobs in weeks 3 and 4. The value of this plan would be $0 + 50 + 10 + 10 = 70$.

(a) Show that the following greedy algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer:

$i \leftarrow 1$
while $i \leq n$ **do**
  if $i < n$ and $H_{i+1} > L_i + L_{i+1}$ **then**
    output "choose no job in week i"
    output "choose a high-stress job in week i+1"
    $i \leftarrow i + 2$
  else
    output "choose a low-stress job in week i"
    $i \leftarrow i + 1$
  end if
end while

Write down both the correct answer for your instance, and the answer incorrectly returned by the algorithm.

(b) Solve this problem using DP approach.

(c) What is the running time of your algorithm?

(d) **(Optional)** Rewrite your DP algorithm so that it uses a memoization technique.