COT5405: Homework 3 (Fall 2017)

Instructions: study Solved Exercises 1 and 2 on pages 307 – 311. Complete the following exercises by defining the recurrence formulae, designing DP algorithms, and analyzing their running time and space requirements. The first two's analyses have been done for you.

1. Exercise 8(b) on pages 319-320.

8. The residents of the underground city of Zion defend themselves through a combination of kung fu, heavy artillery, and efficient algorithms. Recently they have become interested in automated methods that can help fend off attacks by swarms of robots.

   Here's what one of these robot attacks looks like.

   - A swarm of robots arrives over the course of n seconds; in the i^{th} second, x_i robots arrive. Based on remote sensing data, you know this sequence $x_1, x_2, \ldots, x_n$ in advance.
   - You have at your disposal an *electromagnetic pulse* (EMP), which can destroy some of the robots as they arrive; the EMP's power depends on how long it's been allowed to charge up. To make this precise, there is a function $f(\cdot)$ so that if j seconds have passed since the EMP was last used, then it is capable of destroying up to $f(j)$ robots.
   - So specifically, if it is used in the k^{th} second, and it has been j seconds since it was previously used, then it will destroy $\min(x_k, f(j))$ robots. (After this use, it will be completely drained.)
   - We will also assume that the EMP starts off completely drained, so if it is used for the first time in the j^{th} second, then it is capable of destroying up to $f(j)$ robots.

   The problem. Given the data on robot arrivals $x_1, x_2, \ldots, x_n$, and given the recharging function $f(\cdot)$, choose the points in time at which you're going to activate the EMP so as to destroy as many robots as possible.

   Example. Suppose $n = 4$, and the values of x_i and $f(i)$ are given by the following table.

i	1	2	3	4
x_i	1	10	10	1
$f(i)$	1	2	4	8

   The best solution would be to activate the EMP in the 3^{rd} and the 4^{th} seconds. In the 3^{rd} second, the EMP has gotten to charge for 3 seconds, and so it destroys $\min(10, 4) = 4$ robots; In the 4^{th} second, the EMP has only gotten to charge for 1 second since its last use, and it destroys $\min(1, 1) = 1$ robot. This is a total of 5.

> **(b)** Give an efficient algorithm that takes the data on robot arrivals $x_1, x_2, \ldots, x_n$, and the recharging function $f(\cdot)$, and returns the maximum number of robots that can be destroyed by a sequence of EMP activations.

Hint: Let OPT(j) be the maximum number of robots that can be destroyed for the instance of the problem just on $x_1, \ldots, x_j$. Clearly, if the input ends x_j, there is no reason not to activate the EMP then (you are not saving it for anything), so the choice is just when to last activate it before step j. Thus OPT(j) is the best of these choices over all i.

2. Exercise 27 on pages 333-334.

> **27.** The owners of an independently operated gas station are faced with the following situation. They have a large underground tank in which they store gas; the tank can hold up to L gallons at one time. Ordering gas is quite expensive, so they want to order relatively rarely. For each order,

Hint: A solution is specified by the days on which orders of gas arrive, and the amount of gas that arrives on those days. In computing the total cost, we will take into account delivery and storage costs, but we can ignore the cost for buying the actual gas, since this is the same in all solutions. (At least all those where all the gas is exactly used up.)

Consider an optimal solution. It must have an order arrive on day 1, and it the next order is due to arrive on day i, then the amount ordered should be $\sum_{j=1}^{i-1} g_j$. Moreover, the capacity requirements on the storage tank say that i must be chosen so that $\sum_{j=1}^{i-1} g_j \leq L$.

What is the cost of the storing this first order of gas? We pay g_2 to store the g_2 gallons for one day until day 2, and $2g_3$ to store the g_3 gallons for two days until day 3, and so forth, for a total of $\sum_{j=1}^{i-1}(j-1)g_j$. More generally, the cost to store an order of gas that arrives on day a and lasts through day b is $\sum_{j=a}^{b}(j-a)g_j$. Let us denote this quantity by S(a,b).

Let OPT(a) denote the optimal solution for days a through n, assuming that the tank is empty at the start of day a, and an order is arriving. We choose the next day b on which an order arrives: we pay P for the delivery, S(a,b-1) for the storage, and then we can behave optimally from day b onward. Thus we know how to compute OPT(a).

3. Exercise 2(a) & (b) on pages 313-314.

2. Suppose you're managing a consulting team of expert computer hackers, 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, or helping a desperate group of Cornell students finish a project that has something to do with compilers). 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 $\ell_i > 0$ 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's required that they do no job (of either type) in week $i - 1$; they need a full week of prep time to get ready for the crushing stress level. On the other hand, it's 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 that if "high-stress" is chosen for week $i > 1$, then "none" has to be chosen for week $i - 1$. (It's okay to choose a high-stress job in week 1.) 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, and you add h_i to the value if you choose "high-stress" in week i. (You add 0 if you choose "none" in week i.)

The problem. Given sets of values $\ell_1, \ell_2, \ldots, \ell_n$ 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 following table. 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 $0 + 50 + 10 + 10 = 70$.

	Week 1	Week 2	Week 3	Week 4
ℓ	10	1	10	10
h	5	50	5	1

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

```
For iterations i = 1 to n
  If 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"
    Continue with iteration i+2
  Else
    Output "Choose a low-stress job in week i"
    Continue with iteration i+1
  Endif
End
```

To avoid problems with overflowing array bounds, we define $h_i = \ell_i = 0$ when $i > n$.

In your example, say what the correct answer is and also what the above algorithm finds.

(b) Give an efficient algorithm that takes values for $\ell_1, \ell_2, \ldots, \ell_n$ and $h_1, h_2, \ldots, h_n$ and returns the *value* of an optimal plan.