CS208 Theory Assignment 6

12312110 李轩然

DDL: May. 20th 20:50

Chapter 6 Exercise 2

Description

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.

Analysis

(a) A counterexample:

Values	Week 1	Week 2	Week 3
l_i	1	1	1
h_i	1	10	100

The optimal solution is: Choose low-stress job in week 1, no job in week 2, high-stress job in week 3. The total value will be 101.

But in this program, we will choose no job in week 1, high-stress job in week 2, and low-stress job in week 3. The total value will be 11.

The program is that we ignore the effect for next week if we choose high-stress job.

(b) Suppose array $dpl[i]$ means the maximum value if we choose low-stress job at week i, array $dph[i]$ means the maximum value if we choose high-stress job at week i.

If we choose low-stress job at week i, week $i-1$ can be any job (including no job, but choosing low-stress job is always better than no job, so just ingore this case), and we take the bigger value, thus we have:

$$dpl[i] = \max(dpl[i-1], dph[i-1]) + l[i]$$

If we choose high-stress job at week i, week $i-1$ must be no job, and week $i-2$ can be any job, thus we have:

$$dph[i] = \max(dpl[i-2], dph[i-2]) + h[i]$$

Consider the boundary. For $dpl[i]$, we just let $dpl[1] = l[1]$; for $dph[i]$, obviously $dph[1] = h[1]$, when $i = 2$, week 1 must be no job, so $dph[2] = h[2]$.

Choosing low-stress job is always better than no job in week n, so the answer is $\max(dpl[n], dph[n])$.

The pseudocode:

```
l[1...n] h[1...n] dpl[1...n] dph[1...n]
dpl[1] = l[1]
dph[1] = h[1] dph[2] = h[2]
for i: 2 to n
    dpl[i] = max(dpl[i-1], dph[i-1]) + l[i]
    if i >= 3: dph[i] = max(dpl[i-2], dph[i-2]) + h[i]
return max(dpl[n],dph[n])
```