Sample Dynamic Programming Problems

CIS315

The following quote and problems are from the Dasgupta et al text:

“We now turn to the two sledgehammers of the algorithms craft, dynamic programming and linear programming, techniques of very broad applicability that can be invoked when more specialized methods fail. Predictably, this generality often comes with a cost in efficiency.”

I. A contiguous subsequence of a list $S$ is a subsequence made up of consecutive elements of $S$. For instance, if $S$ is $5, 15, -30, 10, -5, 40, 10$, then $15, -30, 10$ is a contiguous subsequence but $5, 15, 40$ is not. Give a linear-time algorithm for the following task:

**Input:** A list of numbers, $a_1, a_2, \ldots, a_n$.

**Output:** The contiguous subsequence of maximum sum (a subsequence of length zero has sum zero).

For the preceding example, the answer would be $10, -5, 40, 10$, with a sum of 55. (Hint: For each $j \in \{1, 2, \ldots, n\}$, consider contiguous subsequences ending exactly at position $j$)

II. Yuckdonald’s is considering opening a series of restaurants along Quaint Valley Highway (QVH). The $n$ possible locations are along a straight line, and the distances of these locations from the start of QVH are, in miles and in increasing order, $m_1, m_2, \ldots, m_n$. The constraints are as follows:

- At each location, Yuckdonald’s may open at most one restaurant. The expected profit from opening a restaurant at location $i$ is $p_i$, where $p_i > 0$ and $i = 1, 2, \ldots, n$.
- Any two restaurants should be at least $k$ miles apart, where $k$ is a positive integer.

Give an efficient algorithm to compute the maximum expected total profit subject to the given constraints.

III. A subsequence is palindromic if it is the same whether read left to right or right to left. For instance, the sequence $A, C, G, T, G, T, C, A, A, A, A, A, T, C, G$ has many palindromic subsequences, including $A, C, G, C, A$ and $A, A, A, A$ (on the other hand, the subsequence $A, C, T$ is not palindromic). Devise an algorithm that takes a sequence $x[1 \ldots n]$ and returns the (length of the) longest palindromic subsequence. Its running time should be $O(n^2)$.

IV. (HARD) Cutting cloth. You are given a rectangular piece of cloth with dimensions $X \times Y$, where $X$ and $Y$ are positive integers, and a list of $n$ products that can be made with parts of the cloth.
using the cloth. For each product \( i \in [1, n] \) you know that a rectangle of cloth of dimensions \( a_i \times b_i \) is needed and that the final selling price of the product is \( c_i \). Assume the \( a_i \), \( b_i \), and \( c_i \) are all positive integers. You have a machine that can cut any rectangular piece of cloth into two pieces either horizontally or vertically. Design an algorithm that determines the best return on the \( X \times Y \) piece of cloth, that is, a strategy for cutting the cloth so that the products made from the resulting pieces give the maximum sum of selling prices. You are free to make as many copies of a given product as you wish, or none if desired.

**Now two problems from the Kleinberg and Tardos text:**

**V.** 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). 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’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 of 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 \( l_1, l_2, ..., l_n \) and \( h_1, h_2, ..., h_n \), find a plan of maximum value.

**example:** Suppose \( n=4 \), and the \( l_i \) and \( h_i \) are \( L=(10, 1, 10, 10) \) and \( H=(5, 50, 5, 1) \). Then the plan of maximum value would be (“none”, “high-stress”, “low-stress”, “low-stress”). The value of this plan would be \( 0+10+10+10 = 70 \).

**VI.** 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 look 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 $x_1, x_2, ..., x_n$ 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 $f(-)$ 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 $X=(x_1, x_2, ..., x_n)$, and given the recharging function $f(-)$, 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 $X=(1, 10, 10, 1)$ and $f(-)=(1, 2, 4, 8)$. The best solution would be to activate the EMP in the $3^{rd}$ and $4^{th}$ 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.