COMP 3711
Spring 2019

Answer to Midterm DP Question

6. **Dynamic Programming** [22pts]
You are given an input array $A[1 \dots n]$. Recall that the *maximum-contiguous subarray (MCS)* is a subarray $A[i \dots j]$ such that $\sum_{k=i}^{j} A[k]$ is maximum among all subarrays. As an example, $A[4 \dots 7]$ is a MCS of the array below

k	1	2	3	4	5	6	7	8	9	10
A[k]	−3	3	−5	18	−1	2	8	−50	−30	5

and has value $18 - 1 + 2 + 8 = 27$.

The problem is now modified so that for given $x > 0$ you need to find the *x-discounted MCS*. The x-discounted *cost* of $A[i \dots j]$ is defined as

$$C(i, j : x) = A[j] + xA[j-1] + x^2 A[j-2] + \cdots + x^{j-i} A[i] = \sum_{k=0}^{j-i} A[j-k] x^k.$$

In the example array above,

$$C(2, 7 : 2) = 8 + 2 \cdot 2 - 2^2 \cdot 1 + 2^3 \cdot 18 - 2^4 \cdot 5 + 2^5 \cdot 3 = 168$$

Given $x > 0$, the *x-discounted* MCS is the subarray $A[i \dots j]$ such that $C(i, j : x)$ is maximum among all subarrays.

By definition, if $x = 1$, the *x-discounted MCS* is exactly the MCS. If $x \neq 1$ it might be different.

In the array above, for example, if $x = 2$, then $A[2 \dots 7]$ is the *x-discounted* MCS of the full array.

The full problem is, given array $A[1 \dots n]$ and $x > 0$, to design an $O(n)$ time dynamic programming algorithm that calculates the cost of the x-discounted MCS.

(A) Prove that, for every i, j with $1 \le i < j \le n$ and $x > 0$,

$$C(i, j : x) = A[j] + xC(i, j-1 : x).$$

$$C(i,j:x) = A[j] + xA[j-1] + x^2A[j-2] + \cdots + x^{j-i}A[i] = \sum_{k=0}^{j-i} A[j-k]x^k.$$

$$\begin{aligned}
C(i,j:x) &= \sum_{k=0}^{j-i} A[j-k]x^k \\
&= A[j] + \sum_{k=1}^{j-i} A[j-k]x^k \\
&= A[j] + \sum_{t=0}^{j-1-i} A[j-(t+1))]x^{t+1} \\
&= A[j] + x\sum_{t=0}^{j-1-i} A[j-1-t]x^t \\
&= A[j] + xC(i,j-1:x)
\end{aligned}$$

Note that this follows directly from the definition!

Part (A) was meant to provide you with a pathway for finding the recurrence relation.

(B) Define

$$V_j = \max_{1 \le i \le j} C(i, j : x).$$

Give a recurrence relation (write it in the space below) for V_j in terms of V_i with $i < j$ and the values in the array.

Tools:

(i) $V_1 = A[1]$

(ii) from part (A), for $i < j$,

$$C(i, j : x) = A[j] + x\, C(i, j-1 : x)$$

$$
\begin{aligned}
V_j &= \max_{1 \le i \le j} C(i, j : x) \\
&= \max\left(A[j], \max_{1 \le i \le j-1} C(i, j : x)\right) \\
&= \max\left(A[j], \max_{1 \le i \le j-1} \left(A[j] + xC(i, j-1 : x)\right)\right) \\
&= \max\left(A[j], A[j] + x \max_{1 \le i \le j-1} C(i, j-1 : x)\right) \\
&= \max\left(A[j], A[j] + xV_{j-1}\right)
\end{aligned}
$$

$$
V_j = \begin{cases} A[j] & if\ j = 1 \\ MAX(A[j], A[j] + xV_{j-1}) & if\ j > 1 \end{cases}
$$

(C) Give documented pseudocode for your DP algorithm to calculate the cost of the x-discounted MCS (based on the recurrence from part (B)), and explain why it is correct.

$$V_j = \begin{cases} A[j] & if\ j = 1 \\ MAX(\ A[j], A[j] + xV_{j-1}\ ) & if\ j > 1 \end{cases}$$

The problem asks you to calculate

$$XDMCS = \max_{1 \le i \le j} C(i, j : x).$$

But this is just

$$XDMCS = \max_{1 \le i \le j} C(i, j : x) = \max_{1 \le j \le n} \left(\max_{1 \le i \le j} C(i, j : x) \right) = \max_{1 \le j \le n} V_j.$$

Given $V_j = \begin{cases} A[j] & if\ j = 1 \\ MAX(\ A[j], A[j] + xV_{j-1}\ ) & if\ j > 1 \end{cases}$, calculate $\max_{1 \le j \le n} V_j$

% Initialize

1. $V[1] = A[1]$;

% Calculate the V_j

2. For $j = 1$ *to* n *do*

3. $\quad V[j] = \max(A[j], A[j] + xV_{j-1})$

A simple O(n) loop solves the problem

% Return $\max_{1 \le j \le n} V_j$.

4. $XDMCS = V[1]$

2. For $j = 2$ *to* n *do*

3. $\quad$ *If* $V[j] > XDMCS$ *then*

3. $\quad\quad XDMCS = V[j]$