CS 473 (Spring 2025)
Homework 2 (due Feb 13 Thu 10am)

Instructions: As in Homework 1.

Problem 2.1:

(a) Consider a function $C(\cdot,\cdot)$ defined by the following recursive formula: for all $i, j \geq 1$,

$$C(i,j) \;=\; \max\Big\{C(i-1,j-1)+f(i,j),\; C(i+2,j-2)+g(i,j),\; C(i+j,j)+h(i,j)\Big\},$$

with the base case $C(i,j) = 0$ if $i \leq 0$ or $j \leq 0$ or $i > n$. Here, f, g, h are given functions that can be evaluated in constant time.

(i) Write pseudocode to evaluate $C(n,n)$ using dynamic programming, and (ii) analyze the running time and space as a function of n.

(b) Consider a function $C(\cdot,\cdot)$ defined by the following recursive formula: for all $i, j \in \{1,\ldots,n-1\}$,

$$C(i,j) \;=\; \min_{k\in\{j+1,\ldots,n\}:\; g(i,j,k)\geq 0} \Big(C(i-1,k)+C(i+1,k)+f(i,j,k)\Big),$$

with base cases $C(i,j) = 0$ if $i = 0$ or $i = n$, and $C(i,j) = 0$ if $j = n$. Here, f, g are given functions that can be evaluated in constant time.

(i) Write pseudocode to evaluate $C(\lfloor n/2 \rfloor, 1)$ using dynamic programming, and (ii) analyze the running time and space as a function of n.

Problem 2.2: Recall the following formulation of the *line break* problem from class: given a sequence of positive integers $\langle a_1,\ldots,a_n\rangle$ and a number L, we want to split the sequence into contiguous subsequences $\langle a_1,\ldots,a_{i_1}\rangle$, $\langle a_{i_1+1},\ldots,a_{i_2}\rangle$, …, $\langle a_{i_{k-1}+1},\ldots,a_n\rangle$, for some $0 = i_0 < i_1 < \cdots < i_{k-1} < i_k = n$, such that each subsequence sums to at most L, while minimizing the *penalty* $\sum_{j=1}^{k} f(L - a_{i_{j-1}+1} - \cdots - a_{i_j})$. Here, f is a given function that can be evaluated in constant time.

(a) Consider a variant of the problem in which we impose an additional constraint that each subsequence has at most 10 elements and the last subsequence has at most 5 elements (i.e., $i_j - i_{j-1} \leq 10$ for all $j = 1,\ldots,k-1$ and $i_k - i_{k-1} \leq 5$).

Describe a modified dynamic programming algorithm to solve this problem. Include the following steps: (i) define your subproblems and indicate how the final answer can be obtained, (ii) derive the recursive formula (including base cases) with brief justifications, (iii) specify a valid evaluation order, and (iv) analyze the running time and space (as a function of n). For this problem, you do not need to write pseudocode if your recursive formula and evaluation order are described clearly. And you do not need to write pseudocode to output the optimal solution; just the optimal value suffices.

(b) Consider a variant of the problem in which we are given an additional input parameter m and we impose an additional constraint that the number of subsequences used (denoted k above) is exactly m. (But we don't impose the constraint from part (a).)

Describe a modified dynamic programming algorithm to solve this problem. Include the following steps: (i) define your subproblems, (ii) derive the recursive formula (including base cases) with brief justifications, (iii) specify a valid evaluation order, and (iv) analyze the running time and space (as a function of n and m). For this problem, you do not need to write pseudocode if your recursive formula and evaluation order are described clearly. And you do not need to write pseudocode to output the optimal solution; just the optimal value suffices.

Problem 2.3: We have an ordered list of n customers, all arriving at time 0 (i.e., the time when the store opens). We have 3 servers. For each $i = 1, \ldots, n$, customer i requires ℓ_i minutes to serve, where ℓ_i is a positive integer at most L. We want to assign customers to servers, so as to minimize the sum of the waiting times of the customers. The customers need to be served in the given order, and *all* customers must be served. Describe a dynamic programming algorithm to solve this problem in time polynomial in n and L.

Example: for $\ell_1 = 5$, $\ell_2 = 3$, $\ell_3 = 10$, $\ell_4 = 20$, $\ell_5 = 6$, $\ell_6 = 15$, and $L = 20$, one (not necessarily optimal) solution is to assign customers 1,3 to server 1, and customers 2,4,5 to server 2, and customer 6 to server 3. Then customer 3 has to wait 5 minutes, customer 4 has to wait 3 minutes, and customer 5 has to wait $3 + 20 = 23$ minutes. The total waiting time is $5 + 3 + 23 = 31$ minutes.

Include *all* of the following steps: (i) first define your subproblems precisely, (ii) then derive the recursive formula (including base cases) with brief justifications, (iii) specify the evaluation order, (iv) write pseudocode to output the optimal value (i.e., the minimum total waiting time), (v) write pseudocode to output the optimal solution (i.e., the assignment of customers to servers), and (vi) analyze the running time and space as a function of n and L.

[Hint: define a class of subproblems with 4 parameters (the number of subproblems may be a function of both n and L)...]