Written Problem Set 4: Dynamic Programming

The first thing you should do in ps4.tex is set up your name as the author who is submitting by replacing the line, `\submitter{TODO: your name}`, with your name and UVA email id, e.g., `\submitter{Grace Hopper (gmh1a)}`.

Next, put your response for problem 1 into problem1.tex, problem 2 into problem2.tex, etc.

Before submitting , also remember to:

– List your collaborators and resources, replacing the TODO in `\collaborators{TODO: replace ...}` with your collaborators and resources. (Remember to update this before submitting if you work with more people.)

– Replace the second line in ps4.tex, `\usepackage{algo}` with `\usepackage[response]{algo}`.

Collaborators: TODO: replace this with your collaborators (if you did not have any, replace this with *None*)

Resources: TODO: replace this with your resources (if you did not have any, replace this with *None*)

Problem 1: Stranger Things

The town of Hawkins, Indiana is being overrun by interdimensional beings called Demogorgons. The Hawkins lab has developed a Demogorgon Defense Device (DDD) to help protect the town. The DDD continuously monitors the inter-dimensional ether to perfectly predict all future Demogorgon invasions.

The DDD allows Hawkins to predict that i days from now a_i Demogorgons will attack. The DDD has a laser that is able to eliminate Demogorgons, but the device takes time to charge. In general, charging the laser for j days will allow it to eliminate c_j Demogorgons (it is not necessarily the case that the c_j values are increasing). Demogorgon attacks and therefore laser activation will always happen at the end of the day, so when activating we consider that day's charging to occur before the activation.

Example: Suppose $(a_1, a_2, a_3, a_4) = (1, 10, 10, 1)$ and $(c_1, c_2, c_3, c_4) = (1, 2, 4, 8)$. The best solution is to activate the laser on days 3 and 4 in order to eliminate 5 Demogorgons.

1. Construct an instance of the problem on which the following "greedy" algorithm returns the wrong answer:

   `BADLASER`$((a_1, a_2, a_3, \ldots, a_n), (c_1, c_2, c_3, , \ldots, c_n))$:

   Compute the smallest j such that $c_j \geq a_n$, Set $j = n$ if no such j exists

   Shoot the laser at time n

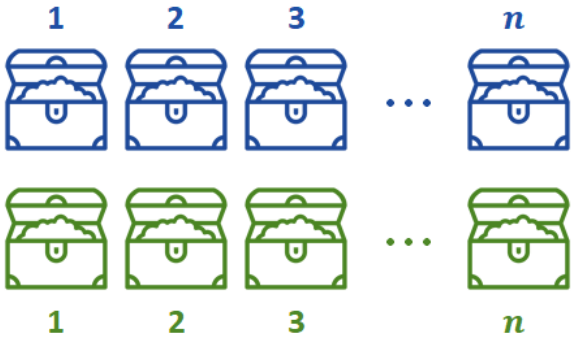
   if $n > j$ then `BadLaser`$((a_1, \ldots, a_{n-j}), (c_1, ..., c_{n-j}))$

   Intuitively, the algorithm figures out how many days (j) are needed to eliminate all the Demogorgons in the last time slot. It activates during that last time slot, and then accounts for the j days required to recharge for that last slot, and recursively considers the best solution for the smaller problem of size $n - j$. In the example, we will charge for 1 day to eliminate the demogorgon who arrives on day 4, and charge for 3 days before that to eliminate 4 of the 10 Demogorgons that arrive on day 3.

2. Given an array holding a_i and c_j, devise a dynamic programming algorithm that eliminates the maximum number of Demogorgons. Provide and justify the running time of your solution.

Problem 2: Game Show

You are a contestant on a new game show, on which you are presented a row of n blue chests and a row of n green chests. The chests in each row are numbered from left to right with the numbers 1 through n:



Each chest has a label displaying the amount of prize money (possibly negative!) contained within the chest. For $i = 1, \ldots, n$, let b_i denote the prize money in the i^{th} blue chest and let g_i denote the prize money in the i^{th} green chest. Your goal is to choose the best combination of chests to open to maximize your total prize money (i.e., the sum of the prize money in the chests you opened). The rules of the game are as follows:

– You must open chests in a left-to-right order and you can open *at most* one chest in each column. You are *not* required to open a chest in every column.

– You must alternate between opening blue chests and green chests. In other words, no consecutively-*chosen* chests can belong to the same row. For example, if you chose chest b_i, skipped columns $i+1$ and $i+2$, and chose a chest from column $i+3$, then you must have selected g_{i+3}. You may choose *any* chest to be the first chest you open.

Give a dynamic programming algorithm for computing the maximum prize money you can earn. The running time of your algorithm must be a polynomial in n.

An example problem instance with accompanying solution (27) is illustrated below:

i	1	2	3	4	5	6	7
b_i	1	$\boxed{-1}$	-2	3	3	$\boxed{10}$	-2
g_i	$\boxed{8}$	1	4	$\boxed{10}$	2	6	-4

Recursive Structure. Explain the optimal substructure for this problem. This should make it clear what choices are available for solving a large problem and which subproblems result from each choice.

Memory. Describe the structure of the memory you'll be using to keep track of solutions to subproblems. Be sure to mention, asymptotically, the size of this memory.

Algorithm. Describe your dynamic programming algorithm using the recursive structure and the memory you provided above.

Running Time. Analyze, asymptotically, the worst-case running time of your algorithm.

Problem 3: Facebook Trolling

People love being smug on Facebook. I've seen a lot of posts on Facebook (mostly posted by Aunts and Uncles) who try to trip people up with ambiguous arithmetic expressions. For example, my Aunt once posted "90% of Millenials will get this wrong! $4-3+2=?$" in hopes that I'd choose to evaluate this as $(4-3)+2$ and answer 3 so that she could counter with "No! PEMDAS! The answer is -1", alluding to the (faulty, in my opinion) order of operations taught in school suggesting that addition always be done before subtraction, obtaining $4-(3+2)=-1$. Inspired by my Aunt's facebook trolling, I want to determine, for some parentheses-free arithmetic expression, the "facebook-worthiness" of that expression. We will say that its facebook-worthiness is the distance between the maximum value and minimum value the expression could evaluate to by changing the order of operations. For instance, my Aunt's expression would have a facebook-worthiness of 4 since $3-(-1)=4$.

You are given an arithmetic expression containing n integers and the only operations are additions $(+)$ and subtractions $(-)$. There are no parenthesis in the expression. For example, the expression might be: $1+2-3-4-5+6$.

You can change the value of the expression by choosing the order of operations:

$$((((1+2)-3)-4)-5)+6=-3$$
$$(((1+2)-3)-4)-(5+6)=-15$$
$$((1+2)-((3-4)-5))+6=15$$

Give a **dynamic programming** algorithm that computes the maximum and minimum possible values of the expression to find its facebook-worthiness. You may assume that the input consists of two arrays: `nums` which is the list of n integers and `ops` which is the list of $n-1$ operations (each entry in `ops` is either `'+'` or `'-'`), where `ops[0]` is the operation between `nums[0]` and `nums[1]`. The running time of your algorithm should be $O(n^3)$. **Hint:** Use a similar strategy to our algorithm for matrix chain multiplication.

Recursive Structure. Identify the optimal substructure for this problem. This should make it clear what choices are available for solving a large problem and which subproblems result from each choice.

Memory. Describe the structure of the memory you'll be using to keep track of solutions to subproblems. Be sure to mention, asymptotically, the size of this memory.

Algorithm. Describe your dynamic programming algorithm using the recursive structure and the memory you provided above.

Running Time. Analyze, asymptotically, the worst-case running time of your algorithm.

