Homework 6: More Dynamic Programming

Version 2: corrected a typo in problem 2. There is no n in the problem; the variable was supposed to be a w.

Be sure to read the grading guidelines and style guidelines. Especially to see the suggested format for describing algorithms.

We sometimes describe how long are justifications or proofs are. These lengths are intended to help you estimate how much detail we're expecting, you should not take those estimates as hard length-limitations.

Our solutions for any individual problem will fit in approximately one page or less. **If you submit an answer substantially longer than 2 pages, the TAs are allowed to stop reading at the end of page 2.**

You are allowed (and encouraged!!) to collaborate with each other. Brainstorming is much easier to do in a group than alone! But you must follow the collaboration policy (which includes needing to write your submission on your own).

You will submit to gradescope; we have a different box for each problem, please give yourself time to submit.

1. A Much More Efficient Tea Party [10 points]

Recall in Homework 2 Problem 5, you designed a recursive algorithm to find the number of different sequences of t teas satisfying a set of constraints. Our solution for that problem took $\mathcal{O}\left((t+p)^k\right)$ time (we strongly recommend looking at our solution if you did something different for this problem).

(a) In our proof, we said "Our function learns, on input `TeaTastings`(v, i) the number of tastings of length i (i.e., tastings with i teas) which end at Earl Grey and start at v." That looks like an English description of a recurrence! Give a recurrence that will calculate this value (the recursive code in the solutions may also help). You'll need to make sure you're using the same graph description our solutions give.

(b) Now that we know DP, memoize it! What is your memoization structure?

(c) What's the filling order?

(d) Is it faster now?! What would the running time be (give 1-2 sentences of justification). WOW memoization makes a big difference!!

2. Dynamic Tea-Partying [25 points]

As a manager of a teahouse, you realize that to stay in business, you need to host at least k tea parties in every two week period to bring in enough customers. Looking at your schedule over the next w weeks, you note that you already have some tea parties planned. You wish to figure out the minimum number of additional tea parties that need to happen in order for the teahouse to stay in business.

More formally, you are given an integer k and an array `E[]` of length w, where the i^{th} element in `E` is the number of tea parties are already planned on the i^{th} week. The requirement to stay in business is that for all $1 \leq i < w$, if you add a_i tea parties to week i and a_{i+1} in week $i+1$ then it must be that $\texttt{E}[i] + \texttt{E}[i+1] + a_i + a_{i+1} \geq k$.

You need to return a number denoting the minimum number of additional tea parties.

(a) Clearly state **in English** what your recurrence(s) calculate. Be sure to mention any parameters you use.

(b) Write a recurrence (or multiple recurrences) that you will use to solve this problem.

(c) Give a brief explanation of your recurrence; we're not looking for a formal proof, but a brief explanation of what each of the cases represent, and why that set of cases is exhaustive.

(d) Describe a memoization structure.

(e) Describe a filling-order for your memoization structure. If you wrote (iterative) pseudocode to fill the structure, what would the running time be?

(f) What is your final answer going to be? (where in the memoization structure do we look?)

3. All-Hands Meeting [25 points]

You have just taken over as CEO of a large dysfunctional company, and you want to have an "all-hands" meeting, where you can get honest feedback from anyone about the current state of the company. To make sure the feedback is honest, though, you can't have an employee and their direct supervisor at the meeting.[1] Your company's supervisory relationships form a tree. Every vertex represents a person and every parent-child relationship means the parent is the supervisor of the child node.

To ensure you can hear from everyone, you will have exactly 3 meetings. Between the three, everyone needs to attend **exactly** one.

But, you need to ensure that no supervisor is in the same meeting as one of their direct reports (i.e., no parent and child in the tree can be at the same meeting). Subject to that condition, your goal is to have as few people as possible attend meeting 1. The biggest conference room was already booked at the time meeting 1 happens, and you could only get one of the smaller ones.

More formally, given a tree T, your goal is to assign exactly one of the labels $1, 2, 3$ to every vertex, so that no edge connects identical labels and the minimum number of '1' labels are used. As usual for our DP problems, you only need to return the number of vertices labeled 1, not the actual assignment.

(a) Give pseudocode for an algorithm to run for this problem (Hint: You don't need DP here! We only needed one line). Give a 1-2 sentence explanation for why your algorithm is correct.

(b) Unfortunately, it seems that some of the non-manager employees have conflicts with the times you've assigned them. Luckily this is only happening to "leaf" nodes of the tree[2]. You now have a tree where some leaves are pre-labeled with $1, 2$ or 3, which is the **only** meeting that works with their schedules. All other nodes (internal nodes and unlabeled leaves) can attend any of the three meetings.

We'll find a dynamic programming algorithm to solve the problem.

- Give an English description of what each of your recurrence(s) calculate and what each of the parameters means.
- Write a recurrence (or multiple recurrences) that describe the minimum number of 1 labels in a tree where you don't get to change any of the pre-labeled nodes.If there is no way to assign everyone to meetings (say one supervisor has children pre-labeled with all three possible labels), return ∞.
- What will your final answer be (e.g. where in the memoization structure should we look)?

(c) What will your evaluation order be?

(d) What is the running time to evaluate your recurrence? Let n be the number of vertices in your tree. Give 1-3 sentences of justification.

[1] Indirect supervision, like having an employee and their boss's boss, is totally fine. The boss's boss doesn't even know who their indirect reports are. That's one of the reasons the company is dysfunctional, actually.

[2] It seems they're the only ones getting real work done...

4. The More You Buy the More You Save [25 points]

At your local plant store, you have a special coupon where everything at the plant store is completely free! The only condition is that you can only carry out plants with a plant tray that can support at most W lbs. Looking around, you see n plants $(p_1, p_2, ..., p_n)$ with tags on them clearly stating the plants retail value v_i and its respective weight w_i. Given that you can only redeem the coupon once, you want to leave the store with the highest value combination of plants without exceeding the capacity of the tray. Describe a dynamic programming algorithm to determine the maximum value you can get out of your coupon.

(a) Clearly state **in English** what your recurrence(s) calculate. Be sure to mention any parameters you use.

(b) Write a recurrence (or multiple recurrences) that you will use to solve this problem.

(c) Give a brief explanation of your recurrence; we're not looking for a formal proof, but a brief explanation of what each of the cases represent, and why that set of cases is exhaustive.

(d) Describe a memoization structure.

(e) Describe a filling-order for your memoization structure. If you wrote (iterative) pseudocode to fill the structure, what would the running time be? Justify the running time with 1-2 sentences.

(f) What is your final answer going to be? (where in the memoization structure do we look?)