Due date: Sunday, November 10, 11:59 PM You will need to submit this as a PDF via GradeScope.

Please review the syllabus and course reference for the expectations of assignments in this class. Remember that problem sets are not online treasure hunts. You are welcome to discuss matters with classmates, but remember the Kenny Loggins rule.

Questions 2 and 3 each requires a **dynamic programming** algorithm for the solution; for each problem, (1) write out the base case and recurrence expressions; (2) explain (briefly) why their expressions are the correct things to compute; and (3) analyze the running time for the iterative version, explaining any important implementation details. You do not need to actually write out the iterative version unless explicitly stated. However, you might find it useful to do so.

It is strongly encouraged that you attempt each of these problems before lecture on Tuesday, November 5.

1. Suppose I am going to choose an integer between 1 and n, inclusive, according to some probability distribution. For each integer i, I have written p_i, the probability that I select i as the chosen integer. You may assume that $\sum_{i=1}^{n} p_i = 1$.

   (a) Give an $\mathcal{O}(n^3)$ time algorithm to compute a $2D$-array X, where $X[i, j]$ is the probability that some integer in the range $[i, j]$ (inclusive) is chosen. You may assume that arithmetic operations take $\mathcal{O}(1)$ time each.

   (b) Give an $\mathcal{O}(n^2)$ time algorithm to solve the problem in part (a). If you are confident that your answer to this question is $\mathcal{O}(n^2)$, you may elect to skip the previous part and count this as your answer to both.

2. College students get a lot of free food at various events. Suppose you have a schedule of the next n days marked with those days when you get a free dinner, and those days on which you must acquire dinner on your own. On any given day you can buy dinner at the cafeteria for \$6. Alternatively, you can purchase one week's groceries for \$20, which will provide dinner for each day that week. However, because you don't have a fridge, the groceries will go bad after seven days (including the day of purchase) and any leftovers must be discarded. Due to your very busy schedule, these are your only three options for dinner each night.

   Write a dynamic programming algorithm to determine, given the schedule of free meals, the minimum amount of money you must spend to make sure you have dinner each night and to print the dates on which you should purchase groceries. Explain why your algorithm is correct and has running time polynomial in n, the number of days on the schedule.

   Hint: Start by writing a recursive procedure to determine how you should eat dinner on the last night.

3. Shindler gives lots of homework assignments, each of which have an easy version and a hard version[1]. Each student is allowed, for each homework, to submit either their answer to the easy version (and get $e_i > 0$ points) or the hard version (and get $h_i > 0$ points, which is also guaranteed to always be more than e_i) or to submit neither (and get 0 points for the assignment). Note that e_i might have different values for each i, as might h_i. The values for all n assignments are known at the start of the semester.

   The catch is that the hard version is, perhaps not surprisingly, more difficult than the easy version. In order for you to do the hard version, you must have not done the previous assignment at all: neither the easy nor the hard version (and thus are more relaxed, de-stressed, etc). Your goal is to maximize the number of points you get from homework assignments over the course of the semester. Give an efficient dynamic programming algorithm to determine the largest number of points possible for a given semester's homework choices.

Not Collected Questions

These questions will not be collected. Please do not submit your solutions for them. However, these are meant to help you to study and understand the course material better. You are encouraged to solve these as if they were normal homework problems.

This homework (approximately) covers Chapter 12, although the specific algorithms from lecture vary from the book. One of the strengths of this book is that it has a good variety and quality of practice problems.

If you need help deciding which problems to do, consider trying R-12.7, R-12.8, C-12.1, C-12.9, C-12.16, A-12.1, A-12.2, A-12.3, A-12.4, A-12.6, A-12.10

[1]This is not the actual policy.