There are two parts to this problem set. The first part is due on Monday, May 10 at the usual time. The second part is due Thursday, May 13, at the usual time. Solutions to the first part will be posted after it is due and before the second part is due.

You will need to submit this via GradeScope. Late problem sets are not accepted beyond a very short grace period.

For this assignment, each answer must be contained within a single piece of paper, although you are allowed to use multiple pages for the assignment. When you submit to GradeScope, you will need to inform the system which page of your scanned PDF contains the answer. *Do this even if your submission is a single page. Failure to do so may cost you points.*

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. Remember that you may not seek help from any source where not all respondents are subject to UC Irvine's academic honesty policy.

Each question except the first requires a **dynamic programming** algorithm for the solution; for each problem, (1) write out the base case and recurrence expressions; (2) give an English specification of the underlying recursive sub-problems 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.

I plan to make the same requirements for exam questions that ask you to design an algorithm with this technique. While requirement (2) may seem like a burden, for what it's worth, many professors who teach algorithms classes have found that this requirements *improves grade averages.*

1 Part One

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.

Part One continues on the next page

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.

   Give an efficient 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.

   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 quarter.

   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 immediate previous assignment at all: neither the easy nor the hard version (and thus are more relaxed, de-stressed, etc). You are allowed to do the hard version of assignment one if you want. Your goal is to maximize the number of points you get from homework assignments over the course of the quarter. Give an efficient dynamic programming algorithm to determine the largest number of points possible for a given quarter's homework choices.

Part Two appears on the next page.

[1] This is not the actual policy.

2 Part Two

This part is due a few days after part one.

When their respective sport is not in season, UCI's student-athletes are very involved in their community, helping people and spreading goodwill for the school. Unfortunately, NCAA[2] regulations limit each student-athlete to at most one community service project per semester, so the athletic department is not always able to help every deserving charity. For the upcoming quarter, we have S student-athletes who want to volunteer their time, and B buses to help get them between campus and the location of their volunteering[3]. There are F projects under consideration; project i requires s_i student-athletes and b_i buses to accomplish, and will generate $g_i > 0$ units of goodwill for the university.

Use dynamic programming to produce an algorithm to determine which projects the athletic department should undertake to maximize goodwill generated. Note that each project must be undertaken entirely or not done at all – we cannot choose, for example, to do half of project i to get half of g_i goodwill. Give the running time of your algorithm. For full credit, your algorithm should have runtime $\mathcal{O}(FBS)$.

If you are having trouble solving this problem, look at the **0-1 Knapsack Problem** in the textbook. Remember, just because you have to be the sole provider of answering this question for yourself doesn't mean you can't ask for help on related problems!

3 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 the dynamic programming chapters in the textbooks, although the specific algorithms from lecture vary from the book.

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

[2]Regulations mentioned in this problem are not necessarily accurate to reality and should be considered parody.

[3]The NCAA also won't permit student-athletes to use their own vehicles for these purposes. Furthermore, each bus can only be used for one project.