This homework covers developing dynamic programming algorithms. It is due in class Friday, May 2.

Write your solutions carefully — your work should be neat, readable, organized, and polished. As with writing a paper, you will likely need at least two drafts — the first draft can be rough (similar to what was written in class) as the focus is on going through the steps and finding a solution, while in the final draft the focus is on clarity of explanation and the quality of writing (similar to what was written after class). While it is not required, you are encouraged to type your work so that the revisions for the final draft are easier to make.

See the Policies page on the course website for information about revise-and-resubmit, late work, and academic integrity as it applies to homework.

For the following problems, develop a dynamic programming algorithm to solve the problem using the process discussed in class. Give each of the steps in the template — don't just give an algorithm. This is not a research task to look up the solution — the point here is to understand and be able to apply the process discussed in class to develop an algorithm yourself.

1. You are participating in a contest which consists of a series of challenges. The wrinkle is that completing a challenge takes time, and so doing challenge i will cause you to miss the next d_i challenges. (There's no way to go back to complete a missed challenge after the fact.) Given that you earn p_i points for completing challenge i (and that you complete any challenge you start), determine which challenges to do so as to maximize your score.

2. You have a paragraph consisting of n words that you wish to display on the screen. Word i is l_i characters long, and the display area is W characters wide. Since the text is most likely longer than the display area is wide, your task is to determine the best places to put in line breaks so that the text fits within the display area while also keeping the length of each line as even as possible and not using too many lines. This "niceness" criteria will be captured by assigning a cost of s^3 to each line of text where s is the number of spaces at the end of the line. The goal is to determine where to put line breaks so as to minimize the total cost. Line breaks can only occur between words.

3. A shuffle S of two strings A and B is formed by interleaving the characters of A and B without changing their order. For example, three possible shuffles of `hello` and `goodbye` are `hellogoodbye`, `hgeololdobye`, and `gohoeldblyoe`. However, `geoodlblohye` is not a shuffle because the h is out of order with respect to the other characters of A. Given strings A, B, and S, determine whether S is a shuffle of A and B. Note that the characters of A and B do not need to be unique —

as in the example of `hello` and `goodbye`, it is fine for the same letter to appear in both strings and for a letter to appear multiple times within one string. The total number of occurrences in S will be the sum of the occurrences in each of A and B.