To receive credit for a problem, you must both have sufficient narrative description of your algorithm and a sentence stating what the intended meaning of each array entry is (e.g. $A(i, j)$ is 1 if there is a subset of the first i items with aggregate sum j).

23. (2 points) The input to this problem consists of an ordered list of n words, and an ideal line length L. The length of the ith word is w_i, that is the ith word takes up w_i spaces. (For simplicity assume that there are no spaces between words.) The goal is to break this ordered list of words into lines, this is called a layout. Note that you can not reorder the words. The length of a line is the sum of the lengths of the words on that line. No line may be longer than L, although it may be shorter. The discrepency for having a line of length K is $(L - K)$. The penalty for having a line of length K is the cube of the discrepency, namely $(L - K)^3$. The penalty of a layout is the sum of the penalties for the lines in that layout.

    (a) Give a dynamic programming algorithm to find the minimum penalty of any layout with running time is $O(nL)$.

    (b) Then give an algorithm to find the actual minimum penalty layout in time $O(nL)$.

    Fact: LaTeXuses this objective, the sum of the cubes of line discrepancies, as its objective when determining line breaks.

24. (4 points) Informally in this problem we consider how to divide the chapters in a story into volumes (think the 7 volumes of Harry Potter or however many Game of Thrones volumes there will be) so as to equalize the size of the volumes. The input consists of positive integers $x_1, \ldots, x_n$ and k. Here x_i is the number of pages in chapter i, and k is the desired number of volumes. The problem is determine which chapters go into each of the k volumes so as to minimize the difference between the most number of pages in any volumes, and the least number of pages in any of the volumes. Of course you can not reorder the story. Give a dynamic programming algorithm whose running time is bounded by a polynomial in n. You running time should not depend on the number of pages in the chapters.

25. (6 points) Consider the following problem. The input is a tree $T = (V, E)$ with positive edge lengths, and a positive integer k. A feasible solution is a subcollection C of k vertices from V. There are called the centers. The objective is to minimize $\sum_{v \in V} \min_{c \in C} d(c, v)$. Here $d(c, v)$ is the aggregate length of the path from c to v in T. So we want to minimize the aggregate distance from the vertices to their closest center. Give a dynamic programming algorithm to solve this problem that runs in time polynomial in n and k.

    Hint: As a warmup, first assume that each vertex in T has at most three neighbors, and all edge lengths are 1. Then next remove the assumption that all edge lengths are 1. Then remove the assumption on the degree bound of the vertices.