Problem Set 3: Dynamic Programming

Due: Friday 10/29 11:59PM

Feel free to work in groups of at most 4 for these - if you have a group of more than 4, please run it by me first. If you do work in a group, please include the names of those that you worked with. **However: each student should submit a separate copy where the solutions have been written by yourself.**

1. Consider the following packing problem: items of varying weights $w_1, ..., w_n$ arrive over time (so the item of weight w_1 arrives before the item of weight w_2, and so on) and we must place them into boxes upon arrival. For each item, we can either place it into the current box we are filling if it fits, or we can close the current box and start a new box. Each box can hold at most W weight, and we cannot put an item into a box that is already closed. Define the slack of a box to be W minus the sum of the weights of the items in the box (so the amount of extra weight the box could still hold). Your goal is to give a polynomial time algorithm that packs boxes so that the sum of the **squares** of the slacks of all boxes (including the last one) is minimized. As an example, suppose $W = 5$, and $w_1 = w_2 = 2$ and $w_3 = w_4 = 3$. Then one possible solution might be packing the first two items in the first box and giving each of the last items their own box, giving slacks 1, 2, and 2 for a total cost of $1^2 + 2^2 + 2^2 = 9$. Another solution might be packing the first item in its own box, the next two into a second, and the last item in its own box, giving slacks 3, 0, and 2 for a total cost of $3^2 + 0^2 + 2^2 = 13$.

   (a) (5 points) Convince me that the algorithm from the first problem of the first problem set won't work for this problem, namely the algorithm that packs boxes as long as the weight will fit and starts a new box only when the next item won't fit in the current box.

   (b) (10 points) Define an algorithm for this problem and convince me of its correctness.

2. Assume that you are given a collection $B_1, ..., B_n$ of boxes. You don't know the individual weights, but you know that each box weighs some integer value between 1 and W pounds. You have also been given a pan balance: this device contains two platforms on which you can place as many boxes as you would like. After placing the boxes, it can tell you which side is heavier. Your goal is to determine if the boxes can be divided into two subcollections (with no overlap) such that the two subcollections have equal weight.

   (a) (5 points) Consider the algorithm that tries every subcollection against every other disjoint subcollection using the pan balance. Is this a polynomial number of uses of the pan balance? Why or why not?

   (b) (10 points) Determine a strategy to determine whether there are two disjoint subcollections of equal weight that uses the pan balance at most $O(n^2W)$ times and convince me of its correctness.