Homework 5

Handed out Monday, August 9. Due at the start of class Friday, August 13. Late homeworks will not be accepted.

Problem 1. Design an algorithm that runs in time $O(nK)$ that takes as input a set $S = \{x_1, x_2, \ldots, x_n\}$ of integers, and a value $K$ and returns a subset $S'$ of $S$ that adds exactly to $K$. In other words, if such a subset exists, your algorithm should output it.

Problem 2. We are going on a trip along the Appalachian trail. We have a list of all possible campsites that we can camp in along the way (say $n$). We want to do this trip in exactly $K$ days, stopping $K-1$ nights to camp. Our goal is to plan this trip so that we minimize the maximum amount of walking done in a single day. In other words, if our trip involves 3 days of walking, and we walk 11, 14, 12 miles on each day respectively, the cost is 14. Another schedule that involves walking 11, 13, 13 miles on each day has cost 13. Note that the location of the campsites is specified in advance, and we can only camp at a campsite. Your algorithm should be based on dynamic programming and run efficiently.

Problem 3. In the alignment problem we are given two strings $X, Y \in \Sigma^*$ and we want to find a sequence of delete/insert/substitute operations with minimum cost that takes us from $X$ to $Y$. The cost of a substitution is given by the function $D : \Sigma \times \Sigma \rightarrow R^+$. Remember in class we extended this function with a new character $\_\_$ to capture insertions and deletions.

Now we want to consider block insertions and deletions. Instead of deleting characters one at the time we are allowed to delete chunks of $l$ consecutive characters. The cost of such operation will be $\alpha + \beta l$, where typically $\alpha > \beta$. Insertions are modeled in the same fashion, if we insert $l$ consecutive characters we pay $\alpha + \beta l$. For example take $\alpha = 10$ and $\beta = 3$, in the following alignment:

\[
\begin{array}{cccccc}
A & C & T & G & T & A
\end{array}
\]

\[
(10 + 3 \times 1) + 0 + (10 + 3 \times 2) + 0 + D(T, A) + (10 + 3 \times 3) = 46 + D(T, A)
\]

The different operations are separated with vertical bars. There are two deletions and one insertion. Two characters remain the same, these can be regarded as “substitutions” with 0 cost ($D(x, x) = 0$ for all $x \in \Sigma$), and finally we substitute $T \rightarrow A$ with cost $D(T, A)$.

Derive a DP formulation for the new version of the problem, analyze the running time of your solution.
Problem 4. We are given a collection of $n$ books, which must be placed in a library book case. Let $h[1..n]$ and $w[1..n]$ be arrays, where $h[i]$ denotes the height of the $i$-th book and $w[i]$ denotes the width of the $i$-th book. Assume that these arrays are given by the books’ catalogue numbers (and the books MUST be shelved in this order), and that books are stacked in the usual upright manner (i.e., you CANNOT lay a book on its side). All book cases have width $W$, but you may have any height you like. The books are placed on shelves in the book case. You may use as many shelves as you like, placed wherever you like (up to the height of the book case), and you may put as many books on each shelf as you like (up to the width of the book case). Assume for simplicity that shelves take up no vertical height.

The book-shelver’s problem is, given $h[1..n]$, $w[1..n]$, and $W$, what is the minimum height book case that can shelve all of these books. Below shows an example of a shelving. The height and width of the 6th book are shown to the right. (Notice that there is some unnecessarily wasted height on the third shelf.)

(a) Write down a recurrence for book shelver’s problem.
(b) Use your recurrence to develop an efficient dynamic programming for solving the book shelver’s problem.
(c) Analyze the running time of your algorithm.