

Homework Assignment 2

September 19, 2002

Due Date: October 3

Core Problems

- (20 pts) We now look at a problem which is really just a variation of the problem you had in homework one of laying out n songs in order on a set of CDs to minimize the number of CDs. Here the problem is to format a sequence of n words on a piece of paper (with the words in the provided order). Formally, the input is a sequence of n words w_1, w_2, \dots, w_n of lengths $\ell_1, \ell_2, \dots, \ell_n$, measured in characters. The maximum number of characters that can be placed on a line is M . (Assume $\ell_i \leq M$ for all i .) The goal is to find where to place the line breaks so that you minimize the sum, over all lines except the last, of the square of the number of extra spaces at the end of the line.

To help clarify the above, I will briefly explain how it was obtained. If a given line contains words i through j and we leave exactly one space between words, the number of extra spaces at the end of the line (or which must be blended into the line when the text is right justified) is $M - j + i - \sum_{k=i}^j \ell_k$. For the formatting to look as good as possible, you want the amount of variation of the white space at the end of the line to be as small as possible. Minimizing the sum, over all of the lines, of the number of extra spaces squared will achieve this goal. Finally, observe, that in considering what looks good, it does not matter if the last line is much shorter than the rest, and hence the last line is not included in sum.

Below are the key aspects of a dynamic programming solution for this optimization problem. Let $width(i, j) = \sum_{k=i}^j \ell_k + (j - i)$ denote the amount of space required for words i through j . We consider the following subproblems: let $m[i]$ be the minimum cost of formatting words i through n . To compute $m[i]$ we will consider all possible “first lines” for our layout of words i through n . Let k be the largest value for which words i through k can fit on a single line. Then

$$m[i] = \begin{cases} 0 & \text{if words } i \text{ to } n \text{ fit on one line} \\ \min_{i \leq j \leq k} (M - width(i, j))^2 + m[j + 1] & \text{otherwise} \end{cases}$$

Using this as a starting point you are to:

- (6 pts) Prove that the algorithm obtained by computing $m[n], m[n - 1], \dots, m[1]$ using the above recursive definition yields an optimal value for $m[1]$. That is, prove that the value of $m[1]$ is the minimum cost possible for any layout of words w_1, \dots, w_n .
- (4 pts) Analyze the time complexity of the resulting algorithm to construct an optimal layout. Be sure to include any optimizations needed to make your algorithm as efficient as possible.
- (10 pts) Implement the algorithm. You will find test data on the course web page. Along with submitting your code, you must also submit the layouts obtained for the test data and the cost for your solution to each.

2. (10 pts) You are traveling by a canoe down a river and there are n trading posts along the way. Before starting your journey, you are given for each $1 \leq i < j \leq n$, the fee $f_{i,j}$ for renting a canoe from post i to post j . These fees are arbitrary. For example it is possible that $f_{1,3} = 10$ and $f_{1,4} = 5$. You begin at trading post 1 and must end at trading post n (using rented canoes). Your goal is to minimize the rental cost. Give the most efficient algorithm you can for this problem. Be sure to prove that your algorithm yields an optimal solution and analyze the time complexity.

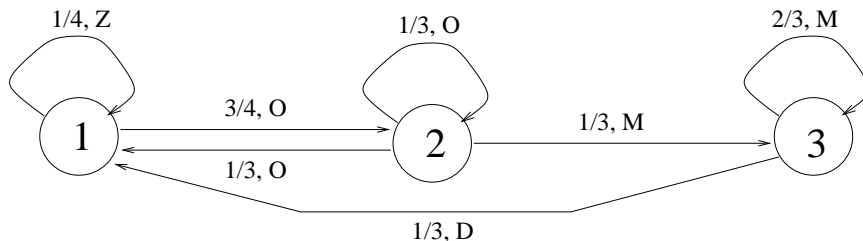
3. (10 pts) Here we return to the ski rental problem. There are m pairs of skis, where the length of the i th pair of skis is s_i . There are n skiers who wish to rent skis, where the height of the i th skier is h_i . Ideally, each skier should obtain a pair of skis whose height matches his own height as closely as possible. Your goal is to assign skis to skiers so that the sum of the absolute differences of the heights of each skier and his skis is minimized.
 Give the most efficient algorithm you can (analyzed as a function of n and m) to optimally solve this problem when $m \geq n$ (i.e. there could be more skies than skiers). You may use any facts that were proven in Homework 1 without repeating the proof here. Be sure to prove that your algorithm yields an optimal solution and analyze the time complexity.

4. (15 pts) For bit strings $X = x_1 \dots x_m$, $Y = y_1 \dots y_n$ and $Z = z_1 \dots z_{m+n}$, we say that Z is an *interleaving* of X and Y if it can be obtained by interleaving the bits in X and Y in a way that maintains the left-to-right order of the bits in X and Y . For example if $X = 101$ and $Y = 01$ then $x_1x_2y_1x_3y_2 = 10011$ is an interleaving of X and Y , whereas 11010 is not. Give the most efficient algorithm you can to determine if Z is an interleaving of X and Y . Prove your algorithm is correct and analyze its time complexity as a function $m = |X|$ and $n = |Y|$.

Advanced Problems, required for CS 539T (extra credit for CS 441T)

NOTE: You may pick one of problems 6 and 7. If you do both then the second will be treated as extra credit even for CS 539T students. However, if you do both you must specify which is to be counted as extra credit.

5. (10 pts) This problem (and variations of it) appear in speech-recognition applications. Consider a k -state Markov chain with states $1, \dots, k$ and a $k \times k$ transition matrix a_{ij} , where a_{ij} is the probability of the chain making a transition from state i to state j in one step. (Here $a_{j1} + \dots + a_{jk} = 1$ for all j .) When the $i \rightarrow j$ transition is made, the chain outputs a symbol s_{ij} which is drawn from some finite alphabet Σ . For example, a simple such Markov chain is shown (pictorially) below:



Consider a situation where the chain begins in state 1, makes n transitions which emit the symbols w_1, \dots, w_n . This models the process of vocalizing a word; the Markov chain can be used to model the choice of word, the speed of speech, variations in pronunciation, etc.

Describe an efficient algorithm that, given a description of the Markov chain and the string w_1, \dots, w_n , produces the most likely path through the Markov chain that could have produced this sequence of symbols. State the time complexity of your algorithm as a function of both k (the number of states) and n (the length of the output string). Note that the probability of a path through the chain is the product of the probabilities of the transitions taken. Give the most efficient algorithm you can for this problem. Be sure to prove that your algorithm yields an optimal solution and analyze the time complexity.

6. (15 pts) You are given a sequence of n songs where the i th song has length ℓ_i minutes. You plan to release a series of five CDs (CD 1, \dots , CD 5) with a selection of the n songs where each CD can hold m minutes of music. Your goal is to pick the maximum number of songs that can fit under the restrictions:
- (1) Songs must be recorded in the given order. That is for $i < j$ if s_i and s_j are both in the collection selected by the solution then s_i must precede s_j (either by having s_i on an earlier CD in the series or have s_i before s_j on the same CD).
 - (2) No song may be split across CDs.

Give the most efficient algorithm you can to find an optimal solution for this problem, prove the algorithm is correct and analyze the time complexity. *BIG HINT: First determine the minimum number of CDs needed to store any s songs. For example, 3.25 CDs needed means that 3 CDs have been completed (though may have some unused time at the end) and a 4th CD is being filled with $m/4$ minutes used so far.*

7. (15 pts) The *Euclidean traveling-salesman problem* is the problem of determining the shortest closed tour that connects a given set of n points in the plane. A closed tour is a path through the points which visits each point exactly once and finishes at the point where it began. In other words, each permutation of the points defines a closed tour, where the path starts at the first point in the permutation, visits the points in the order given by the permutation, and finally closes the tour by going from the last point in the permutation to the first one.

A simplification of this problem is obtained by considering *bitonic tours* which are tours that start at the leftmost point, go strictly left to right to the rightmost point, and then go strictly right to left back to the starting point. You may assume that no two points have the same x coordinate. Give the most efficient algorithm you can for this problem. Be sure to prove that your algorithm yields an optimal solution and analyze the time complexity.

Hint: Scan left to right, maintaining optimal possibilities for the two parts of the tour.