

## Homework Assignment 2

September 21, 2004

Due Date: October 5

## Core Problems

1. (25 pts) Here we consider the problem of optimally deciding where to put line breaks when formatting text. 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:

- (a) (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  $w_1, \dots, w_n$ .
  - (b) (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.
  - (c) (15 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. NOTE: If you feel you would learn more by solving another problem instead of doing the implementation you can use one of the two extra credit problems (or problem 5 if you are a 441 student) in place of this.
2. (10 pts) Here we return to problem 3b of homework 1. 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.

3. (10 points) Give a dynamic programming solution to the following problem. The input to this problem is a complete binary tree  $T$  with root  $r$  where each edge in  $T$  is directed towards the leaves and has a real-valued weight which may be negative, zero, or positive. You are to give a linear time algorithm to find a maximum weight directed path in  $T$ . This path need not start at the root or end at a leaf. Any directed path in  $T$  is to be considered.

Note: A complete binary tree is one in which every internal node has exactly two children. Here's some notation for you to use.

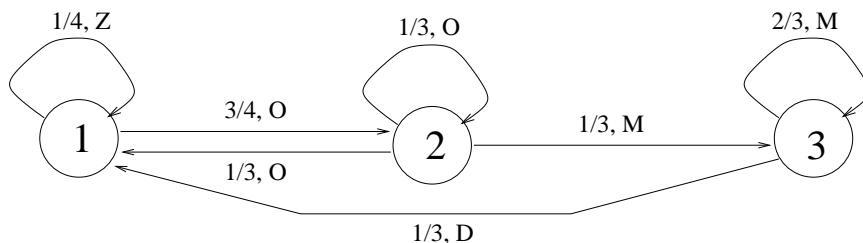
- For  $x$  a node in the tree, let  $T(x)$  be the subtree rooted at node  $x$ . So  $T = T(r)$ .
- For  $x$  a node in the tree, let  $\text{Leaf}(x)$  be a boolean that is true exactly when  $x$  is a leaf.
- For  $x$  an internal node in the tree, let  $\text{Left}(x)$  be the node that is  $x$ 's left child, and let  $\text{LeftEdgeWeight}(x)$  be the weight on the edge from  $x$  to  $\text{Left}(x)$ .
- For  $x$  an internal node in the tree, let  $\text{Right}(x)$  be the node that is  $x$ 's right child, and let  $\text{RightEdgeWeight}(x)$  be the weight on the edge from  $x$  to  $\text{Right}(x)$ .

You are required to solve this problem using **dynamic programming**. (While you could formulate this as a shortest path problem, you will get a more efficient solution using dynamic programming). Try to make your algorithm as efficient as you can. Be sure to clearly describe the algorithm, prove it is correct, and analyze the asymptotic time complexity.

4. (15 pts) We return to problem 1b of Homework 1. We have  $n$  jobs, where only one job can be processed at a time. Job  $i$  must start at time  $s_i$ , end at time  $e_i$ , and if run will result in a profit of  $p_i$ . The goal is to find a schedule that results in the maximum total profit. Give the most efficient algorithm to compute an optimal solution to this problem. Be sure to clearly describe the algorithm, prove it is correct, and analyze the asymptotic time complexity.

### Advanced Problem, required for CSE 541T (extra credit for CS 441T)

5. (15 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  $\Sigma$ . 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 asymptotic time complexity.

## Challenge Problems (extra credit for everyone)

Note: These points will not be added into your score but rather kept separately to be used in picking the final grade in borderline cases. So you should be sure to do all you can on the other problems before working on either of these.

6. (15 pts) You are given a sequence of  $n$  songs where the  $i$ th song has length  $\ell_i$  minutes. You plan to release a series of five CDs (CD 1, ..., 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. (10 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.