COMP 360 Algorithm Design Techniques

Assignment 1: Dynamic Programming

We hope to hold tutorials September 16 and 17. Watch Minerva and the COMP 360 announcements web page, available from the menu at www.cs.mcgill.ca/ sue/360

Due date: Thurs. Sept. 18

1. *Counting Paths on a Grid.*

   (a) Consider a grid (like the one from class) that has diagonal shortcuts as well edges directed to the right or down. Show how to calculate the total number of paths from the origin to point (i, j) on a grid using dynamic programming. [*Hint.* Slightly adapt the method we used to find the shortest path from the origin to (i, j).]

   (b) Prove that the recurrence you used in part a) is correct.

   (c) Apply your method to find the total number of disjoint paths from the origin to the point $(3, 2)$.

2. *Longest Common Subsequence Problem.*

   (a) Show how the longest common subsequence problem can be solved using a dynamic program for the shortest path problem on a grid with shortcuts. [*Hint.* Label the rows and columns of the grid by the elements of X_n and Y_m respectively. Then show how each path from the origin to (n, m) can be identified with a common subsequence of X and Y. Finally assign lengths to each arc so that a minimum length path corresponds to a longest common subsequence.]

   (b) Use this (or the dynamic program seen in class) to find the longest common subsequence of $X =$ `ACTAGGACCT` and $Y =$ `CGTAGATC`.

3. *Planning a Company Party.* (CLRS textbook, Question 15-4).

   Suppose that a company with a hierarchical structure (the supervisor relation forms a tree rooted at the president) is having a party. The personnel office has ranked each employee with a conviviality rating which is a real number (possibly negative). In order to make the party fun for all attendees the president of the company does not want both an employee and his or her immediate supervisor to attend.

(a) Design a dynamic program to make up the guest list. The goal should be to maximise the sum of the conviviality ratings of the guests.

(b) Prove that that the recurrence you used in your dynamic program is correct.

(c) Analyse the running time of your algorithm.

4. *Longest Increasing Subsequence.*

   Given a sequence $X = (x_1, x_2, \ldots, x_n)$, we say that $Z = (z_1, z_2, \ldots, z_k)$ is an increasing subsequence of X if $(z_1, z_2, \ldots, z_k)$ is a subsequence of X *and* $z_1 < z_2 < \cdots < z_k$. For example, $(1, 4, 7, 8, 9)$ is an increasing subsequence of $(1, 4, 7, 2, 8, 3, 5, 7, 9, 6)$.

   (a) Let f_i be the length of the longest increasing subsequence of $X = (x_1, x_2, \ldots, x_n)$ *ending in* x_i. Calculate $f_1, f_2, \ldots, f_{10}$ for the subsequence $(1, 4, 7, 2, 8, 3, 5, 7, 9, 6)$. Use this to verify that $(1, 2, 3, 5, 7, 9)$ is a longest increasing subsequence.

   (b) Set up a recursive definition of f_i and prove its validity. [*Hint.* The longest subsequence ending in x_i cannot contain any element of $x_1, x_2, \ldots, x_{i-1}$ that is at least as large as x_i. It follows that your definition need not depend on any f_j with $j < i$ and $x_j \geq x_i$.]

   (c) Use your recurrence to find a longest increasing subsequence of $(5, 8, 3, 9, 4, 8, 1, 2, 9, 6)$.

5. *Ruler Folding.*

   Given a sequence of segments of positive integer lengths $l_1, \ldots, l_n$ connecting joints $A_0, \ldots, A_n$ of a carpenter's ruler, determine the length k of the shortest line segment onto which the ruler can be folded. In class, we showed that $M \leq k \leq 2M$, where $M = max\{l_i\}$, $1 \leq i \leq n$. One way to determine k is to build a sequence of tables to test if the ruler folds into length M, length $M + 1$, etc., until k is determined.

   (a) Propose a way to solve the Ruler Folding problem by building only one table.

   (b) Is your dynamic programming solution (or the one given in class) a polynomial time solution to the problem? Why or why not?