ArsDigita University
Month 5: Algorithms - Professor Shai Simonson

Problem Set 5 – Dynamic Programming Algorithms

1. World Series Odds

The chance of team A winning a World Series (a best 4 out of 7 competition for you non-baseball fans) in a match with an evenly matched team B, is denoted as $P(a,b)$, where a is the number of games already won by team A, and b is the number of games already one by team B. Write a dynamic programming solution to calculate $P(a,b)$. Your solution should work for any series of odd length $2n+1$, not just for 7.

2. The Electoral College

Modify the Knapsack algorithm to solve the Partition problem. The Partition problem gives a set of integers and asks if the set can be partitioned into two parts so that the sums of the integers in each part are equal.

a. Write code for your algorithm and use it to check whether or not it is possible to have a tie vote in our electoral college.
b. Enhance your algorithm to print out the actual values whose sums are equal.

3. Carnival Games

Consider an n by n array of positive integers (a_{ij}), $1 \leq i, j \leq n$, rolled into a cylinder, so that the top and bottom rows are glued together.

A path is to be threaded from the entry side of the cylinder to the exit side, subject to the restriction that from the given square (i,j) it is possible to move to $(i+1, j)$, $(i+1, j-1)$ or $(i+1, j+1)$. The path may begin at any position on the entry side and end at any position on the exit side. The cost of such a path is the sum of the integers in the squares through which it passes. You should figure out the minimum cost path.

a. Write a recursive solution to this problem and analyze its complexity.
b. Write a dynamic programming solution and show its complexity to be $\theta(n^2)$ (Note, this is really a special case of the shortest path problem on an acyclic graph, so it can be solved by conventional techniques in $\theta(e)$ time but I don't want you to do it this way).

4. Applications of the Knapsack Problem

a. Write a recursive solution to determine how many different ways there are to make n cents in change using any coins from among pennies, nickels, dimes, quarters and half dollars. (e.g. there are 6 ways to make 17 cents in change).
b. Write a dynamic programming implementation of your algorithm and analyze its complexity.

Big hint: Let $C(i,j)$ be the number of ways to make change for j cents using coins a_1 through a_i, where coin a_1 is a penny coin a_2 is a nickel etc. Note that this hint solves the problem independent of the actual denominations of the coins.

5. The Liquid Knapsack

Consider a version of the knapsack problem (liquid version) where you are allowed to place fractional amounts of each object into the knapsack.

a. Describe a greedy style algorithm that solves this problem.
b. Analyze how much time your algorithm requires and explain why it works.

6. Simple Parsing

Modify the CYK algorithm by storing more information, and writing a procedure to help recover and print the actual grammar productions that generate the string in question. (Hint: For each non-terminal in $V(i,j)$ store the k which made that non-terminal appear. Then use these stored values to backtrack from $V(1,n)$.)

7. (Optional) Approximate String Matching

String matching algorithms are an important application used in editors and word processors. Spell-checker programs must flag a word and give suggestions for its correct spelling, hence these programs need to match strings approximately. That is, they must be able to check how far apart two strings are from each other. We say that two strings are distance n apart if there is a way to change one into another by a combination of n insertions, deletions or changes of single letters.

Let $s = s_1s_2\ldots s_m$ be a search string of length m and $t = t_1t_2\ldots t_m$ be text string of length p. Let $C(i, j)$ be the minimum distance between $s_0s_1\ldots s_i$ and a segment of t ending at t_j.

a. Explain why: $C(0, j) = 0$, $C(i, 0) = i$, and when $i>0$ and $j>0$
   $C(i, j) = min\,\{C(i\text{-}1, j)+1, C(i, j\text{-}1)+1, C(i\text{-}1, j\text{-}1)\}$ if $s_i = t_j$
   $C(i, j) = min\,\{C(i\text{-}1, j)+1, C(i, j\text{-}1)+1, C(i\text{-}1, j\text{-}1)+1\}$ if $s_i \neq t_j$
b. Write code for a dynamic programming algorithm based on part (a), that takes a search string and a text string, and finds the j for which $C(i, j)$ is minimum.