San Skulrattanakulchai
Gustavus Adolphus College Oct 30, 2014

Study Questions for Dynamic Programming

Recurrence

1. A quantity $m(i)$ has the recurrence

$$m(i) = \max\{\, m(j) + c(i,j) : i < j \le n \,\} \qquad 1 \le i \le n.$$

The base case is not shown, and $c(\cdot,\cdot)$ is a given cost function.

(i) To specifiy the base case we will supply the value of $m(0)$ or $m(n+1)$. Which one does *not* work? Explain your answer.

(ii) Assume we solve this problem by dynamic programming. What is the running time for the table-filling step? Explain your answer.

2. A certain problem has the following recurrence, where $c(\cdot,\cdot)$ is a given cost function.

$$m(i) = \begin{cases} 0 & \text{if } i = 1 \\ \min\{\, c(k,i) + \sum_{j=1}^{k} m(j) : 1 \le k < i \,\} & \text{if } 1 < i \le n \end{cases}$$

(i) Explain why a straightforward implementation of the recurrence results in an $O(n^3)$ time bound for the table-filling step.

(ii) Show how to fill the table in time $O(n^2)$.

Flight Problem

We will use the backward version of recurrence.

$$m(i) = \begin{cases} 0 & \text{if } i = n \\ \min\{\, c(i,k) + m(k) : i < k \le n \,\} & \text{if } 1 \le i < n \end{cases}$$

1. The following recursive procedure ROUTE(i) prints out the route from city i to city n. E.g.,

1 5 10

gives the route from city 1 to city 10 that goes through city 5. Complete the pseudocode for ROUTE(i) by giving 1 statement for each of **a–c**. Assume that the $M[\cdot]$ and $K[\cdot]$ tables have been filled in; here $K[i]$ is the minimizer corresponding to $M[i]$.

```
/* Precondition: 1 ≤ i ≤ n */
ROUTE(i) {
  if i = n then {
    ________a________
  } else {
    ________b________
    ________c________
  }
}
```

2. Here are the values of $c(i, k)$:

	$k : 2$	3	4	5
$i : 1$	3	4	11	116
2		3	8	112
3			6	111
4				110

   Give the 2 missing entries in the table below. Here $K[i]$ is the minimizer for $M[i]$. Show your work.

i	1	2	3	4	5
$M[i]$	?	112	111	110	0
$K[i]$	?	5	5	5	–

3. We want to solve the flight problem, but instead of the cost function $c(\cdot, \cdot)$, we are given the functions $t(\cdot)$ and $\ell(\cdot)$, where,
   $t(i)$ is the cost of taking off from city i; and
   $\ell(i)$ is the cost of landing at city i.
   (i) Explain how to use the original recurrence to solve this problem.
   (ii) Modify the recurrence for this problem.

4. We are given two cities a, b where $1 < a < b < n$, and we never want to fly directly from a to b.
   (i) Modify the recurrence for this problem.
   (ii) Instead of changing the original recurrence, modify the cost function $c(\cdot, \cdot)$.

Longest Common Subsequence Problem

1. Professor Dull proposes the following algorithm for printing out the LCS of $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$.

```
/* print LCS of x_1 x_2 ... x_i and y_1 y_2 ... y_j */
PRINTLCS(i, j) {
        if i = 0 or j = 0 then
                return
        if C[i, j] = C[i − 1, j − 1] + 1 then {
                PRINTLCS(i − 1, j − 1)
                print x_i
        } else if C[i − 1, j] > C[i, j − 1] then
                PRINTLCS(i − 1, j)
        else /* x_i ≠ y_j and C[i − 1, j] ≤ C[i, j − 1] */
                PRINTLCS(i, j − 1)
}
```

Give input strings $X = x_1x_2 \ldots x_m$ and $Y = y_1y_2 \ldots y_n$ such that PRINTLCS(m, n) will print out an incorrect LCS.

2. Write a correct pseudocode for PRINTLCS procedure.

3. Write a procedure PRINTX-LCS(i, j) that will print out the string $x_1x_2 \ldots x_i$ minus the LCS. E.g., if X is **stamp** and Y is **tame**, then PRINTX-LCS(5, 4) will print **sp** since LCS(**stamp**, **tame**) is **tam**, so **stamp** with the LCS deleted from it is **sp**.

4. Suppose we execute our LCS algorithm on input strings $X = x_1x_2 \ldots x_n$ and $Y = y_1y_2 \ldots y_n$ of equal length. Give the running time for each step. Explain.

5. Here is a backward recurrence for the LCS problem.

$$c(i,j) = \begin{cases} 0 & \text{if } i = m+1 \text{ or } j = n+1 & \text{[base case]} \\ c(i+1, j+1) + 1 & \text{if } 1 \le i \le m,\ 1 \le j \le n,\ x_i = y_j & \text{[match case]} \\ \max\{\, c(i, j+1), c(i+1, j) \,\} & \text{if } 1 \le i \le m,\ 1 \le j \le n,\ x_i \ne y_j & \text{[unmatch case]} \end{cases}$$

(i) What is the quantity we are seeking?

(ii) Fill in the following dynamic programming table for the input strings **PAPAL** and **APPLY**.

	A	P	P	L	Y	
P						0
A						0
P						0
A						0
L						0
	0	0	0	0	0	0

Matrix-Chain Multiplication Problem

1. Here is the recurrence for the problem.

$$m(i,j) = \begin{cases} 0 & \text{if } i = j \\ \min\{\, m(i,k) + m(k+1,j) + p_{i-1}p_kp_j : i \le k < j \,\} & \text{if } i < j. \end{cases}$$

   Rewrite the recurrence using the base case $i = j - 1$ instead of $i = j$.

2. Matrix A has dimension 2 x 5; matrix B has dimension 5 x 2; and matrix C has dimension 2 x 10. Find the optimal way to fully parenthesize the product ABC. Show your work.