Advanced Algorithms Assignment II

Matthew Bennett

Dynamic Programming and Greedy Algorithms HW Draft date February 2, 2006

Exercise 15.2-1 Optimal Matrix Composite Product for $< 5, 10, 3, 12, 5, 50, 6 >$

Using a computer to get the M values, we have:

j/k	A	B	C	D	E	F
A	0	150	330	405	1655	2010
B		0	360	330	2430	1950
C			0	180	930	1770
D				0	3000	1860
E					0	1500
F						0

The corresponding s values are:

j/k	A	B	C	D	E	F
A	0	1	2	2	4	5
B		0	2	2	2	2
C			0	3	4	4
D				0	4	4
E					0	5
F						0

One optimal parenthetization is: $((A_1A_2)((A_3A_4)(A_5A_6)))$

```
#include <iostream>

using namespace std;

 const int numMatr = 7;
 const int p[numMatr] = {5,10,3,12,5,50,6};
 const int n = numMatr - 1;

void PRINT_OPTIMAL_PARENS(int s[n][n], int i, int j);

int main() {
 int i, j, k, l, q;
 int m[n][n];
 int s[n][n];

 for (i = 1; i <= n; i++)
 {
  m[i][i] = 0;
 }
 for (l = 2; l <= n; l++)       //l is the chain length.
 {
    for (i = 1; i <= n - l + 1; i++)
    {
     j = i + l - 1;
     m[i][j] = 999999999;
     for (k = i; k <= j - 1; k++)
     {
      q = m[i][k] + m[k + 1][j] + p[i-1]*p[k]*p[j];
      if (q < m[i][j])
      {
            m[i][j] = q;
            s[i][j] = k;
      }
     }
  }
 }

 PRINT_OPTIMAL_PARENS(s, 1, n);

 system("pause");
}



void PRINT_OPTIMAL_PARENS(int s[n][n], int i, int j) {
 if (i == j) cout << "A_" << i;
 else
 {
  cout << "(";
  PRINT_OPTIMAL_PARENS(s, i, s[i][j]);
  PRINT_OPTIMAL_PARENS(s, s[i][j] + 1, j);
  cout << ")";
 }
}
```

Exercise 15.4-1 Determine an LCS of $< 1, 0, 0, 1, 0, 1, 0, 1 >$ and $< 0, 1, 0, 1, 1, 0, 1, 1, 0 >$.

10111110 from:
11**10111110** and
1011110100

Exercise 15.4-3 Give a memoized version of LCS-LENGTH that runs in $O(m \times n)$ time.

```
int LCS_LENGTH(int X[m], int Y[n], int m, int n)
{
 //c is the global memoization table, X and Y are strings

 if (c[m][n]) > -1 return c[m][n];
 if (m == 0 || n == 0) //empty row or col array
    c[m][n] = 0
    else
    {
     if (X[m] = Y[m]) c[m][n] = LCS_LENGTH(X, Y, m - 1, n - 1) + 1;
     else c[m][n] = max(
       LCS_LENGTH(X, Y, m, n - 1), // whichever is
       LCS_LENGTH(X, Y, m - 1, n)  //  smaller
      );
    }
 return c[m][n];
}
```

Exercises 16.2-4 Professor Midas drives an automobile from Newark to Reno along Interstate 80. His car's gas tank, when full, holds enough gas to travel n miles, and his map gives the distances between gas stations on his route. The professor wishes to make as few gas stops as possible along the way. Give an efficient method by which Professor Midas can determine at which gas stations he should stop, and prove that your strategy yields an optimal solution.

If the professor stays on Interstate 80 without deviating, his only choice will be in how far down the road he chooses to refill. The optimal strategy to minimize the number of stops is to go as far as possible on each tank of petrol without refilling.

Show that the optimal substructure property holds Suppose there are n places for Midas to refuel. Consider an optimal solution of s stops, the one being at k. The rest of the optimal solution must be also be an optimal solution to the subproblem of the remaining n - k stations. How can we be sure that this is true? Assume a better subproblem solution exists. It must therefore have fewer than s - 1 stops, we could use it to come up with a solution with fewer than s stops for the full problem, contradicting our original claim. So the optimal substructure property holds.

Show that the greedy choice property yields an optimal solution First we will assume that the method outlined above is the "greedy choice", as it is trivial to show. Now, assume that there are stations along the way, and Midas makes the Greedy choice at a given time by choosing to visit station k. If Midas brother, Jonas chooses station k-1 at that step (same initial conditions), then he could have chosen station k instead, which means he made a suboptimal choice. So Midas' Greedy Strategy is an optimal strategy.