Justin Baraboo
Design and Analysis of Algorithms
Assignment 3

March 16, 2015

1 Problem 1

Find an optimal parenthesization of a matrix chain multiplication whose sequence of dimensions is (7, 10, 5, 16, 9, 22).

Solution We use the inductive formula:

$$M(i,j) = \min_{i \leq k < j} \{M(i,k) + M(k+1,j) + p_{i-1}p_k p_j\}$$
$$M(i,i) = 0$$

For this problem we have the following matrices:
$7 \times 10, 10 \times 5, 5 \times 16, 16 \times 9, 9 \times 22$
We will now build the M-matrix. When calling M(1,5), we will make subsequent calls until we reach M(i,i) for $i = 1, 2, 3, 4, 5$, giving the following matrix:

	1	2	3	4	5
1	0				
2	-	0			
3	-	-	0		
4	-	-	-	0	
5	-	-	-	-	0

Now we can compute M(1,2), which is:

$$M(1,2) = \min_{i \le k < j} \{M(1,k) + M(k+1,2) + p_0 p_k p_2\}$$

k can only take the value, 1, so:

$M(1,2) = M(1,1) + M(1,2) + 7 * 10 * 5 = 0 + 0 + 350 = 350; k = 1.$

Similarily, in M(2,3), M(3,4), M(4,5), k must take the lower value giving:

$$M(2,3) = M(2,2) + M(2,3) + 10 * 5 * 16 = 0 + 0 + 800 = 800; \quad k = 2$$
$$M(3,4) = M(3,3) + M(3,4) + 5 * 15 * 9 = 0 + 0 + 720 = 720; \quad k = 3$$
$$M(4,5) = M(4,4) + M(4,5) + 16 * 9 * 22 = 0 + 0 + 3168 = 3168; \quad k = 4$$

Updating our Matrix, note that the superscripts will show where the parenthesis are from each choice (which will be derived from our k matrix below it):

	1	2	3	4	5
1	0	350^{12}			
2	-	0	800^{23}		
3	-	-	0	720^{34}	
4	-	-	-	0	3168^{45}
5	-	-	-	-	0

Our K matrix:

	1	2	3	4	5
1	-	1			
2	-	-	2		
3	-	-	-	3	
4	-	-	-	-	4
5	-	-	-	-	-

For, M(1,3), k may be values 1 or 2. Giving

$$\begin{aligned} M(1,3) &= \min\{M(1,1) + M(2,3) + 7 * 10 * 16, M(1,2) + M(3,3) + 7 * 5 * 16\} \\ &= \min\{1920, 910\} \\ &= 910; k = 2 \end{aligned}$$

Similarily for the rest:

$$\begin{aligned} M(2,4) &= \min\{M(2,2) + M(3,4) + 10 * 5 * 9, M(2,3) + M(4,4) + 10 * 16 * 9\} \\ &= \min\{1170, 2240\} \\ &= 1170; k = 2 \\ M(3,5) &= \min\{M(3,3) + M(4,5) + 5 * 16 * 22, M(3,4) + M(5,5) + 5 * 9 * 22\} \\ &= \min\{4928, 1710\} \\ &= 1710; k = 4; \end{aligned}$$

Update the Matrices:

	1	2	3	4	5
1	0	350^{12}	$910^{(12)3}$		
2	-	0	800^{23}	$1170^{2(34)}$	
3	-	-	0	720^{34}	$1710^{(34)5}$
4	-	-	-	0	3168^{45}
5	-	-	-	-	0

Our K matrix:
And we continue:

	1	2	3	4	5
1	-	1	2		
2	-	-	2	2	
3	-	-	-	3	4
4	-	-	-	-	4
5	-	-	-	-	-

$$
\begin{aligned}
M(1,4) &= \min\{M(1,1)+M(2,4)+7*10*9, M(1,2)+M(3,4)+7*5*9, M(1,3) \\
&\qquad M(4,4)+7*16*9\} \\
&= \min\{1800, 1385, 1918\} \\
&= 1385; k=2 \\
M(2,5) &= \min\{M(2,2)+M(3,5)+10*5*9, M(2,3)+M(4,5)+10*16*9, \\
&\qquad M(2,4)+M(5,5)+10*9*22\} \\
&= \min\{2160, 5408, 3150\} \\
&= 2160; k=2
\end{aligned}
$$

Updating

	1	2	3	4	5
1	0	350^{12}	$910^{(12)3}$	$1385^{(12)(34)}$	
2	-	0	800^{23}	$1170^{2(34)}$	$2160^{2((34)5)}$
3	-	-	0	720^{34}	$1710^{(34)5}$
4	-	-	-	0	3168^{45}
5	-	-	-	-	0

Our K matrix:

	1	2	3	4	5
1	-	1	2	2	
2	-	-	2	2	2
3	-	-	-	3	4
4	-	-	-	-	4
5	-	-	-	-	-

Finally,

$$
\begin{aligned}
M(1,5) &= \min\{M(1,1)+M(2,5)+7*10*22, M(1,2)+M(3,5)+7*5*22, \\
&\qquad M(1,3)+M(4,5)+7*16*22, M(1,4)+M(5,5)+7*9*22\} \\
&= \min\{3700, 2830, 6542, 2771\}
\end{aligned}
$$

Giving the final matrix:

	1	2	3	4	5
1	0	350^{12}	$910^{(12)3}$	$1385^{(12)(34)}$	$2771^{((12)(34))5}$
2	-	0	800^{23}	$1170^{2(34)}$	$2160^{2((34)5)}$
3	-	-	0	720^{34}	$1710^{(34)5}$
4	-	-	-	0	3168^{45}
5	-	-	-	-	0

Our K matrix:

	1	2	3	4	5
1	-	1	2	2	4
2	-	-	2	2	2
3	-	-	-	3	4
4	-	-	-	-	4
5	-	-	-	-	-

Giving the optimal multiplications as: ((12)(34))5 for 2771 operations.

2 Problem 2

Design an $O(n^2)$-time dynamic programming algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers. You need to show the inductive formula and a pseudo code.

Solution For this problem, we will sort the sequence, and then find the longest common subsequence between the original and the sorted sequence. The inductive formula then comes from the longest common subsequence problem.

$$c(i,j) = \{ \begin{array}{ll} 0 & if\ i = 0 or j = 0 \\ c(i-1, j-1) + 1 & if x_i = y_i \\ \max(c(i, j-1), c(i-1, j)) & if x_i \neq y_i \} \end{array}$$

```
LCS-Length(X, Y)
{
   m = 1ength[X]
   n = 1ength[Y]
   for i = 1 to m
      do c[i, 0] = 0
   for j = 1 to n
      do c[0, j] = 0
   for i = 1 to m
      do for j = 1 to n
         do if xi = yj
            then {  c[i, j] = c[i-1, j-1] + 1
                 b[i, j] = DIAG_ARROW
               }
            else if c[i-1, j] = c[i, j-1]
               then {  c[i, j] = c[i-1, j]
                    b[i, j] = UP_ARROW
                  }
               else {  c[i, j] = c[i, j-1]
                    b[i, j] = LEFT_ARROW
                  }
 Return c and b
}

Print-LCS(b, X, i, j)
{
 if i = 0 or j = 0
   then return
 if b[i, j] = DIAG_ARROW
   then {  Print-LCS(b, X, i-1, j-1)
        print xi
      }
   else if b[i, j] = UP_ARROW
      then Print-LCS(b, X, i-1, j)
      else Print-LCS(b, X, i, j-1)
}

maxIncSub( seq[1...n])
{
```

```
   sortSeq = SortLowtoHigh( seq[1...n]) )
   c[n,n],b[n,n]
   c,b =LCS-Length(seq[1...n], sortSeq[1...n])
   Print-LCS(b, seq[1...n], n,n)
}
```

Analysis:
Our basic operation is the comparison in the subprogram. Our input size is the size of our sequence n. It takes $O(n^2)$ comparisions for the subprogram. Note that sorting the sequence can take $n\log(n)$ and thus not affect the complexity. Printing the sequenced takes $O(n)$ time and thus not affect the complexity. So the total complexity is $O(n^2)$.

3 Problem 3

Design a dynamic programming algorithm to find the order to weld the n pieces such that the total cost is minimized. You need only to show the formula and initial conditions from which an algorithm can be developed.

Solution For this problem, we will use that if given a sequence of pipes with weights $w_1w_2...w_i...w_j...w_n$ the inductive formula for welding the pipes $w_i....w_j$ is:

$$c(i,j) = \min_{i \leq k < j} \{c(i,k) + c(k+1,j) + w(i...k) + w(k+1...j)\}$$
$$c(i,i) = 0$$

Qualitatively, what this is saying is that, the cost of making a pipe with pipes i through j is the cost of making the pipes $i..k$ and $k+1...j$ and the cost of picking them up.

Next we will show this works for the following sequence of pipes:
W[1] = 8, W[2] = 1, W[3] = 7, W[4] = 2, W[5] = 9.
We'll build the matrix similar to our first problem of matrix multiplication optimization.

	1	2	3	4	5
1	0				
2	-	0			
3	-	-	0		
4	-	-	-	0	
5	-	-	-	-	0

$$
\begin{aligned}
c(1,2) &= \min\{c(1,1) + c(2,2) + w(1) + w(2)\} \\
&= 0 + 0 + 8 + 1 = 9; k = 1 \\
c(2,3) &= \min\{c(2,2) + c(2,3) + w(2) + w(3)\} \\
&= 0 + 0 + 1 + 7; k = 2 \\
c(3,4) &= \min\{c(3,3) + c(4,4) + w(3) + w(4)\} \\
&= 0 + 0 + 7 + 2 = 9; k = 3 \\
c(4,5) &= \min\{c(4,4) + c(5,5) + w(4) + w(5)\} \\
&= 0 + 0 + 2 + 9 = 11; k = 4
\end{aligned}
$$

	1	2	3	4	5
1	0	9^{12}			
2	-	0	8^{23}		
3	-	-	0	9^{34}	
4	-	-	-	0	11^{45}
5	-	-	-	-	0

Our K matrix:

	1	2	3	4	5
1	-	1			
2	-	-	2		
3	-	-	-	3	
4	-	-	-	-	4
5	-	-	-	-	-

$$
\begin{aligned}
c(1,3) &= \min\{c(1,1)+c(2,3)+w(1)+w(2..3), c(1,2)+c(3,3)+w(1...2)+w(3)\} \\
&= \min\{0+8+8+(1+7), 9+0+(8+1)+7\} \\
&= 24; k=1 \\
c(2,4) &= \min\{c(2,2)+c(3,4)+w(2)+w(3...4), c(2,3)+c(4,4)+w(2...3)+w(4)\} \\
&= \min\{0+9+1+(7+2), 8+0+(1+7)+2\} \\
&= 18; k=2 \\
c(3,5) &= \min\{c(3,3)+c(4,5)+w(3)+w(4...5), c(3,4)+c(5,5)+w(3...4)+w(5)\} \\
&= \min\{0+11+7+(2+9), 9+0+(7+2)+9\} \\
&= 27; k=4
\end{aligned}
$$

	1	2	3	4	5
1	0	9^{12}	$24^{1(23)}$		
2	-	0	8^{23}	$18^{(23)4}$	
3	-	-	0	9^{34}	$27^{(34)5}$
4	-	-	-	0	11^{45}
5	-	-	-	-	0

Our K matrix:

	1	2	3	4	5
1	-	1	1		
2	-	-	2	2	
3	-	-	-	3	4
4	-	-	-	-	4
5	-	-	-	-	-

$$c(1,4) = \min\{c(1,1) + c(2,4) + w(1) + w(2...4), c(1,2) + c(3,4) + w(1...2) + w(3...4),$$
$$c(1,3) + c(4,4) + w(1...3) + w(4)\}$$
$$= \min\{36, 36, 42\}$$
$$= 36; k = 1$$
$$c(2,5) = \min\{c(2,2) + c(3,5) + w(2) + w(3...5), c(2,3) + c(4,5) + w(2....3) + w(4...5)$$
$$c(2,4) + c(5,5) + w(2...4) + w(5)\}$$
$$= \min\{46, 48, 37\}$$
$$= 37; k = 4$$

	1	2	3	4	5
1	0	9^{12}	$24^{1(23)}$	$36^{1((23)4)}$	
2	-	0	8^{23}	$18^{(23)4}$	$37^{((23)4))5}$
3	-	-	0	9^{34}	$27^{(34)5}$
4	-	-	-	0	11^{45}
5	-	-	-	-	0

Our K matrix:

	1	2	3	4	5
1	-	1	1	1	
2	-	-	2	2	4
3	-	-	-	3	4
4	-	-	-	-	4
5	-	-	-	-	-

$$c(1,5) = \min\{c(1,1) + c(2,5) + w(1) + w(2...5), c(1,2) + c(3,5) + w(1...2) + w(3...5),$$
$$c(1,3) + c(4,5) + w(1...3) + w(4...5), c(1,4) + c(5,5) + w(1...4) + w(5)\}$$
$$= \min\{64, 63, 62, 63\}$$
$$= 54; k = 3$$

	1	2	3	4	5
1	0	9^{12}	$24^{1(23)}$	$36^{1((23)4)}$	$62^{(1(23))(45))}$
2	-	0	8^{23}	$18^{(23)4}$	$37^{((23)4))5}$
3	-	-	0	9^{34}	$27^{(34)5}$
4	-	-	-	0	11^{45}
5	-	-	-	-	0

Our K matrix:

	1	2	3	4	5
1	-	1	1	1	3
2	-	-	2	2	4
3	-	-	-	3	4
4	-	-	-	-	4
5	-	-	-	-	-

So our solution is to weld 2 to 3, 1 to 23, 4 to 5, and 123 to 45 or (1(23))(45) giving a total cost of 62.

4 Problem 4

Coin Matrix Problem

Solution For this problem we will use the following inductive formula:

$$
\begin{aligned}
c[1,1] &= V[1,1] \\
c[1,j] &= c[1,j-1] + V[1,j]; j > 1 \\
c[i,1] &= c[i-1,1] + V[i,1]; i > 1 \\
c[i,j] &= \max\{c[i-1,j] + V[i,j], c[i,j-1] + V[i,j]\}; i,j > 1
\end{aligned}
$$

We will also construct a path matrix where if we choose the left choice, it will add an $\leftarrow$ to the matrix element and if we choose the upper choice, we add a $\uparrow$ to the matrix.

```
printPath( k[1...n,1...m] )
{
  row = n
  col = m
  String[n+m-2] answer;
  answer.add( n + " - " + m + " end")
  while(n >=2 || m >=2)
  {
   if( k[n,m] == UP_ARROW )
   {
   n = n -1
   answer.add( n + " - " + m + " go down")
   }
   else // k[n,m] == LEFT_ARROW
   {
   m = m -1
   answer.add( n + " - " + m + " go right")
   }
  }
  for(int i = n+m-2, i >0, i--)
  {
   print(answer[i])
  }
}

FindBestPath( V[1...n,1...m] )
{
   c[1...n,1...m]
   k[1...n,1...m]

   c[1,1] = V[1,1]
   for(int j = 2, j <= m, j++)
   {
    c[1,j] = c[1,j-1] + V[1,j]
    k[1,j] = LEFT_ARROW
   }

   for(int i = 2, j <= n, i++)
   {
```

```
    c[i,1] = c[i-1,1] + V[i,1]
    k[i,1] = RIGHT_ARROW
   }

   for(int i = 2, i <=n, i++)
   {
      for(int j =2, j<=m, j++)
      {
       if(c[i-1, j) >= c [i,j-1] )
       {
        c[i,j] = c[i-1,j] + V[i,j]
        k[i,j] = LEFT_ARROW
       }
       else
       {
        c[i,j] = c[i,j-1] + V[i,j]
        k[i,j] = UP_ARROW
       }
      }
   }
   printPath( k[1...n,1...m] )
   print( "Total amount gained is: " + c[n,m] )

}
```

Analysis:
Our input size is the size of the the chessboard and our basic operation is the comparision operator. For an $n \times m$ chessboard we construct an $(n+1) \times (m+1)$ matrix. It's size is: $nm+n+m+1 \in O(nm)$. So our construction is $O(nm)$. Printing out a path only takes $O(n+m)$ comparisions and so does not affect the total complexity. Thus our total complexity is $O(nm)$.