Matrix Chain Multiplication

- Problem
 - Find an optimal parenthesization of matrix chain multiplication
- Matrix chain multiplication
 - $\mathbf{M} = \mathbf{M}_1 \mathbf{M}_2 \dots \mathbf{M}_n$
- Parenthesization
 - A product of matrices is fully parenthesized if it is either a single matrix or the product of two fully parenthesized matrix products, surrounded by parentheses

Example

• The product of $M_1M_2M_3M_4$ can be parenthesized in five distinct ways

$$(M_{1}(M_{2}(\underline{M_{3}M_{4}}))),$$

$$(\underline{M_{1}((\underline{M_{2}M_{3}})M_{4})}),$$

$$((\underline{M_{1}M_{2}})(M_{3}M_{4})),$$

$$((\underline{M_{1}(\underline{M_{2}M_{3}}))M_{4}}),$$

$$(((\underline{M_{1}M_{2}})M_{3})M_{4}),$$

Cost of matrix multiplication

- The cost of matrix multiplication is dominated by scalar multiplications.
- The cost of multiplying a $p \times q$ matrix with a $q \times r$ matrix is pqr

```
for (i=0; i<p; i++)
for (j=0; j<r; j++) {
    C[i][j] = 0;
    for (k=0; k<q; k++)
        C[i][j] += A[i][k]*B[k][j];
}
```

Why parenthesization is important

- Consider a chain <A1, A2, A3> and the dimensions of the matrices are 10×100, 100×5, and 5×50.
 - Cost for ((A1A2)A3) is 10*100*5+10*5*50 = 7,500
 - Cost for (A1(A2A3)) is 100*5*50+10*100*50=75,000
 - One is 10 times faster than the other!

The Matrix Chain Multiplication Problem

• Given a chain of $\langle M_1, M_2, ..., M_n \rangle$ of matrices, where for i = 1, 2, ..., n, matrix M_i has dimension $p_{i-1} \times p_i$, fully parenthesize the product in a way that $M_1 M_2 ... M_n$ minimizes the number of scalar multiplications

Counting the number of parenthesizations

• We can split a sequence of n matrices between k_{th} and $(k+1)_{st}$ matrices. We obtain the recurrence

$$t(n) = \begin{cases} 1, & n = 1\\ \sum_{k=1}^{n} t(k)t(n-k), & n > 1 \end{cases}$$

• The solution is the sequence of *Catalan numbers*

$$t(n) = C(n-1), \text{ where } C(n-1) = \frac{1}{n+1} {2n \choose n} = \Omega(\frac{4^n}{n^{3/2}})$$

A recursive equation (optimal substructure)

- Let m[i,j] be the minimum number of scalar multiplications needed to compute M_{i..j}
 - The cost for is $M_{1..n}$ is m[1,n]
- Assume that the optimal parenthesization splits the product $M_i M_{i+1} ... M_i$ between M_k and M_{k+1}
 - Based on the principle of the optimality $m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_kp_j$
 - We obtain the following recurrence

$$m[i, j] = \begin{cases} 0, & i = j \\ \min_{i \le k < j} (m[i, k] + m[k+1, j] + p_{i-1} p_k p_j), & i < j \end{cases}$$

Dynamic programming

- We start from each single matrix
- We then calculate the minimum number of multiplications for length l sequence for l=2,...n

- The final solution is that for length n

An implementation using dynamic programming

```
Cost:

n + \sum_{l=2}^{n} \sum_{i=1}^{n-l+1} (l-1)
= n + \sum_{l=2}^{n} (n-l+1)(l-1)
= n + \sum_{j=1}^{n-1} (n-j)j
= n + n \sum_{j=1}^{n-1} j - \sum_{l=1}^{n-1} j^{2}
= n + \frac{n^{2}(n-1)}{2} - \frac{n(n-1)(2n-1)}{6}
= (n^{3} + 5n)/6
\in \Theta(n^{3})
```

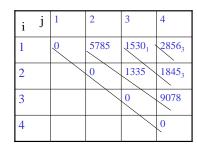
Construct the optimal parenthesization

- In our algorithm, the matrix s tracks the split point
 - Can you use the matrix to construct the optimal parenthesization?

```
PrintOptimalParens(s, i, j) \\ \{ \\ if (i==j) \\ print ("M"_i) \\ else \\ \{ \\ print("("); \\ PrintOptimalParens(s, i, s[i][j]); \\ PrintOptimalParens(s, s[i][j]+1, j); \\ Print(")") \\ \} \\ \}
```

Example

M1	13×5
M2	5×89
M3	89×3
M4	3×34



 $m[1][3] = min(m[1][1] + m[2][3] + 13*5*3, m[1][2] + m[3][3] + 13*89*3) = min(1530,9256) = 1530 \\ m[2][4] = min(m[2][2] + m[3][4] + 5*89*34, m[2][3] + m[4][4] + 5*3*34) = min(24208,1845) = 1845 \\ m[2][4] = min(m[2][2] + m[3][4] + m[3]$

 $\begin{array}{ll} \min[1][4] = \min(m[1][1] + m[2][4] + 13*5*34, & k=1 \\ m[1][2] + m[2][4] + 13*89*34, & k=2 \\ m[1][3] + m[4][4] + 13*3*34) & k=3 \\ = \min(4055, 54201, 2856) = 2856 & k=3 \end{array}$