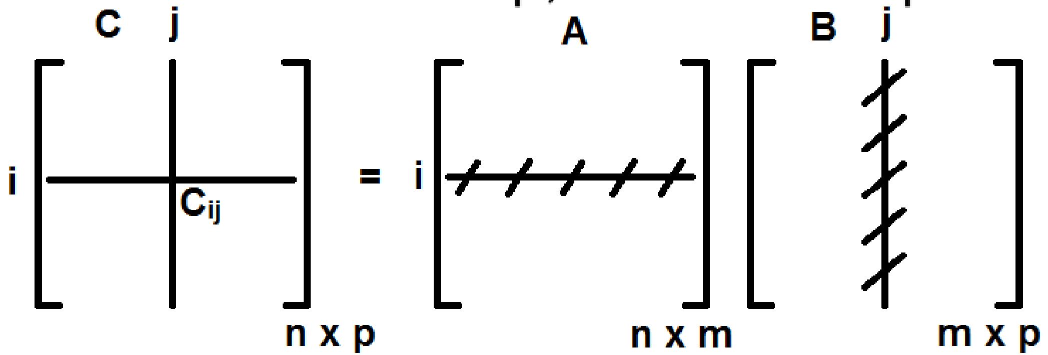
Module 7
Dynamic Programming

S. Lakshmivarahan
School of Computer Science
University of Oklahoma
USA-73019
Varahan@ou.edu

Matrix Chain Product

- Let A and B be two matrices compatible for multiplication and let C be the product: C=AB
- If A is n x m and B is m x p, then C is n x p.



$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$ - Requires 2m operations

Matrix Chain Product

- Thus, the total cost for finding C is 2nmp since there are np elements in C
- Dropping the factor two, we will say that the cost is nmp
- Consider an expression

$$E_4 = A_1 A_2 A_3 A_4$$

where A_i is a matrix of size d_{i-1} x d_i for $1 \leq i \leq 4$

Matrix Chain Product

- If we did not parenthesize, the compiler, given the implicit rules for evaluating expressions, will evaluate it as

$$E_4 = A_1(A_2(A_3A_4))$$

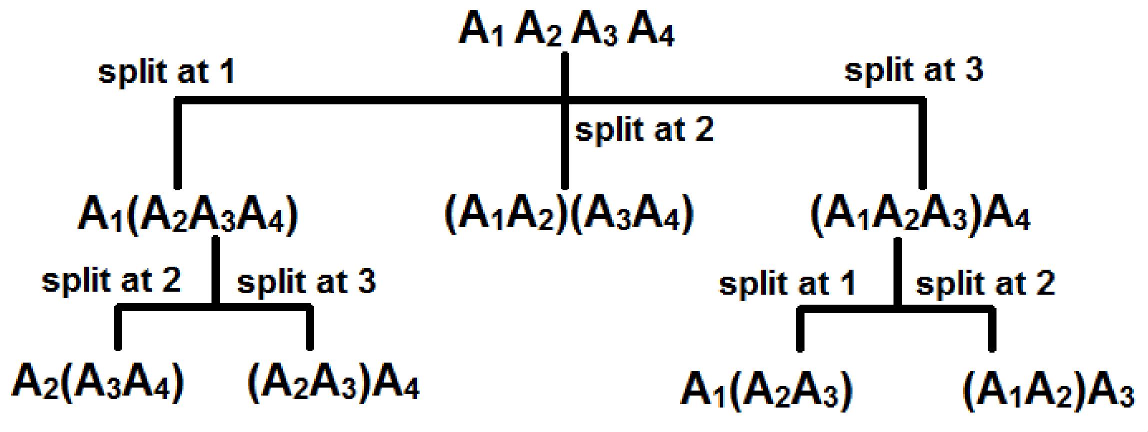
- However, it turns out that the total cost evaluation of E is very sensitive to how this expression is parenthesized

Matrix Chain Product

- To illustrate, let

A_1	A_2	A_3	A_4
30 x 1	1 x 40	40 x 10	10 x 25

- Then, there are four different ways to parenthesize E_4



- .

Matrix Chain Product

- Combining, we get

$A_1 (A_2 (A_3 A_4))$ $(A_1A_2)(A_3A_4)$ $A_1(A_2A_3)A_4$

$A_1((A_2A_3)A_4)$ $((A_1A_2)A_3)A_4$

- Hence, there are only 4 distinct ways to parenthesize E_4

Matrix Chain Product

- Now we compute the cost corresponding to each way of computing E

1. $A_1 (A_2 (A_3 A_4))$: 40x1x25 + 1x40x25 + 30x1x25 = 11,750
2. $A_1((A_2A_3)A_4)$: 1x40x10 + 1x10x25 + 30x1x25 = 1,400
3. $(A_1A_2)(A_3A_4)$: 30x1x40 + 40x10x25 + 30x40x25 = 41,200
4. $((A_1A_2)A_3)A_4$: 30x1x40 + 30x40x10 + 30x10x25 = 20,700

- Hence, E = $A_1((A_2A_3)A_4)$ is the optimal way

Matrix Chain Product

- Generalizing, let

$$E_n = A_1 A_2 A_3 \ldots A_{n-1} A_n$$

where A_i is a d_{i-1} x d_i matrix

- How many ways to parenthesize E_n?

Matrix Chain Product

- Let us divide E_n into two subexpressions as

$$E_n = (A_1 A_2 \dots A_k)|(A_{k+1} A_{k+2} \dots A_n)$$

$$= E_k * E_{n-k}$$

i.e. E_n is the product of two subexpressions E_k and E_{n-k}.

Matrix Chain Product

- Let P(n) be the number of ways to parenthesize E_n
- Then, by divide and conquer principle,

Matrix Chain Product

- Since we can divide E_n in (n-1) ways by changing k=1,2,…,n-1, it follows that

$$P(n) = \sum_{k=1}^{n-1} P(k)P(n-k) \text{ for } n\geq 2 \rightarrow ①$$

where P(1) = 1

Matrix Chain Product

- Let $P(k) \geq 2^n$. Then $P(n\text{-}k) \geq 2^{n-k}$ substituting

$$P(n) \geq \sum_{k=1}^{n-1} 2^n = (n-1)2^n$$

thus, $P(n)$ grows faster than 2^n.

- The sequence of integers generated by ① is called by Catalan numbers.

Matrix Chain Product

- It can be verified that

$$P(n) = \Omega\left(\frac{4^n}{n^{3/2}}\right)$$

- Thus, the number of (feasible) ways to parenthesize is way too large so that the simple enumeration scheme used in E_4 will not work.

Matrix Chain Product

- Dynamic programming is a method that uses the above divide and conquer principle to exhaust all possible solutions with the condition that no subproblem is solved more than once

Matrix Chain Product

- In the case of E_4, the subproblems can be solved in the following order:

1. (A_1A_2) (A_2A_3) (A_3A_4)

- There is only one way to compute each of these products involving two matrices

Matrix Chain Product

2. A) Consider $A_1 A_2 A_3$:

- This can be computed as $(A_1 A_2) A_3$ or $A_1 (A_2 A_3)$. Since the optimal cost of A_1A_2, A_2A_3 are known in step 1, we use those results to find the best between these two ways

Matrix Chain Product

B) Similarly, consider $A_2 A_3 A_4$:

- This can be computed as $A_2 (A_3 A_4)$ and $(A_2 A_3) A_4$ since optimal cost of A_3A_4 and to A_3 are known in Step 1, we can now find the best way to compute $A_2 A_3 A_4$.

Matrix Chain Product

3. Consider $A_1 A_2 A_3 A_4$

- We can do it in three ways

a) $A_1 (A_2 A_3 A_4)$

b) $(A_1 A_2) (A_3 A_4)$

c) $(A_1 A_2 A_3) A_4$

Matrix Chain Product

- Using the optimal cost for computing (A_2 A_3 A_4) found in Step 2, we can find the cost of computing E_4 using (a)
- Using the optimal cost of computing (A_1 A_2) and (A_3 A_4) from Step 1, we can compute the cost of computing E_4 using (b)
- Using the optimal cost of computing ($A_2A_3A_4$) in Step 2, we can compute the cost of computing E_4 using (c)

Matrix Chain Product

- By comparing these three costs, we can find the optimal cost of computing E_4.

Matrix Chain Product

Cost=

	A_1	A_2	A_3	A_4
A_1	O	1200	700	1400
A_2	-	O	400	650
A_3	-	-	O	10,000
A_4	-	-	-	O

A_1: 30 x 1
A_2: 1 x 40
A_3: 40 x 10
A_4: 10 x 25

Matrix Chain Product

optimal split = point

	A_1	A_2	A_3	A_4
A_1	O	1	1	1
A_2	-	O	2	3
A_3	-	-	O	3
A_4	-	-	-	O

Matrix Chain Product

- The minimum cost for computing E_4 is 1,400
- The optimal parenthesis that gives this minimum cost is

$$\begin{aligned} E_4 &= A_1(A_2A_3A_4) \\ &= A_1((A_2A_3)) \end{aligned}$$

since the split for $A_1A_2A_3A_4$ is at 1 and that for $A_2A_3A_4$ is at 3.

Matrix Chain Product

- A general recurrence for the optimal cost M(i,j) to compute the matrix product $A_i\, A_{i+1}\; \ldots\; A_j$ for $i < j$ is obtained as follows, A_i is a matrix of the size $d_{i-1}\; x\; d_i$

$$E(i\colon j) = A_i A_{i+1} \ldots A_k \;\mid\; A_{k+1} \ldots A_j$$

Results in a matrix of size d_{i-1} x d_k

↓

Costs = m(i, k)

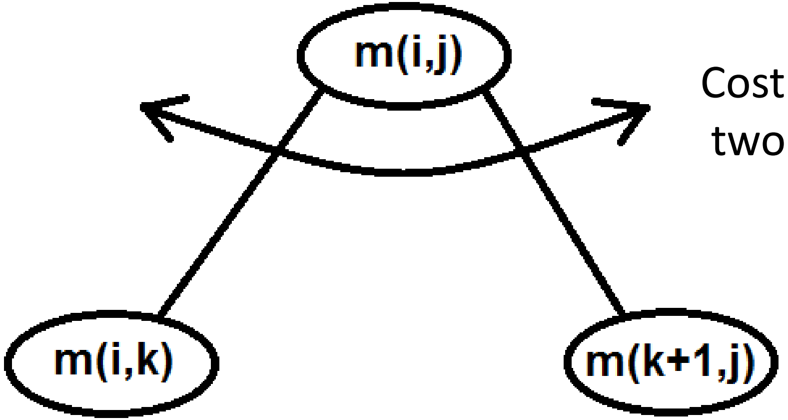
Results in a matrix of size d_k x d_j

↓

Costs = m(k+1, j)

Matrix Chain Product

- Therefore,



Cost of multiplying two parts = $d_{i-1}d_kd_j$

- Thus,

$$m(i,j) = \min_{i\leq k\leq j-1}\{m(i,j) + m(k+1,j) + d_{i-1}d_kd_j\}$$

- Hence, for E_n the optimal cost is m(1,n)

Matrix Chain Product

Problems:

1) Compute the optimal cost and parenthesis for multiplying matrices of order 10 x 3, 3 x 15, 15 x 25 and 25 x 57.
2) What happens when all the matrices are square matrices of the same size.