Algorithms Theory

14 – Dynamic Programming (2)

Matrix-chain multiplication

P.D. Dr. Alexander Souza

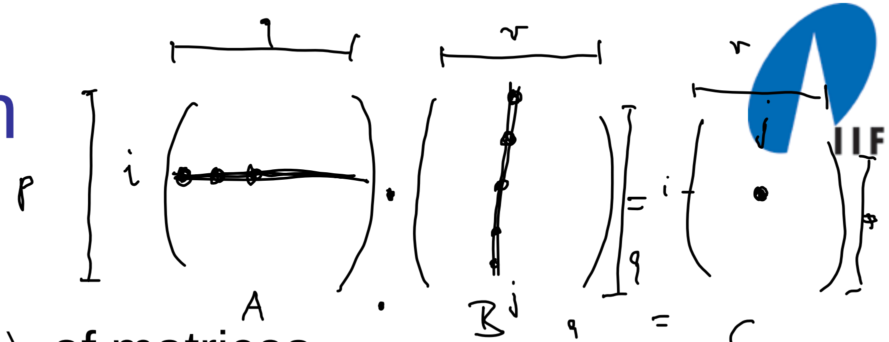
Winter term 11/12

Optimal substructure

Dynamic programming is typically applied to
optimization problems.

An optimal solution to the original problem contains
optimal solutions to smaller subproblems.

Matrix-chain multiplication



$c_{ij} = \sum_{k=1}^{q} a_{ik} \cdot b_{kj}$

$p \cdot r \cdot q$

Given: sequence (chain) $\langle A_1, A_2, \ldots, A_n \rangle$ of matrices

Goal: compute the product $A_1 \cdot A_2 \cdot \ldots . \cdot A_n$

Problem: Parenthesize the product in a way that minimizes the number of scalar multiplications.

Definition: 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.

- (A)
- $((\ldots)\cdot(\ldots))$

Examples of fully parenthesized matrix products of the chain $\langle A_1, A_2, \ldots, A_n \rangle$

All possible fully parenthesized matrix products of the chain $\langle A_1, A_2, A_3, A_4 \rangle$ are:

$$(A_1 \cdot (A_2 \cdot (A_3 \cdot A_4)))$$

$$(A_1 \cdot ((A_2 \cdot A_3) \cdot A_4))$$

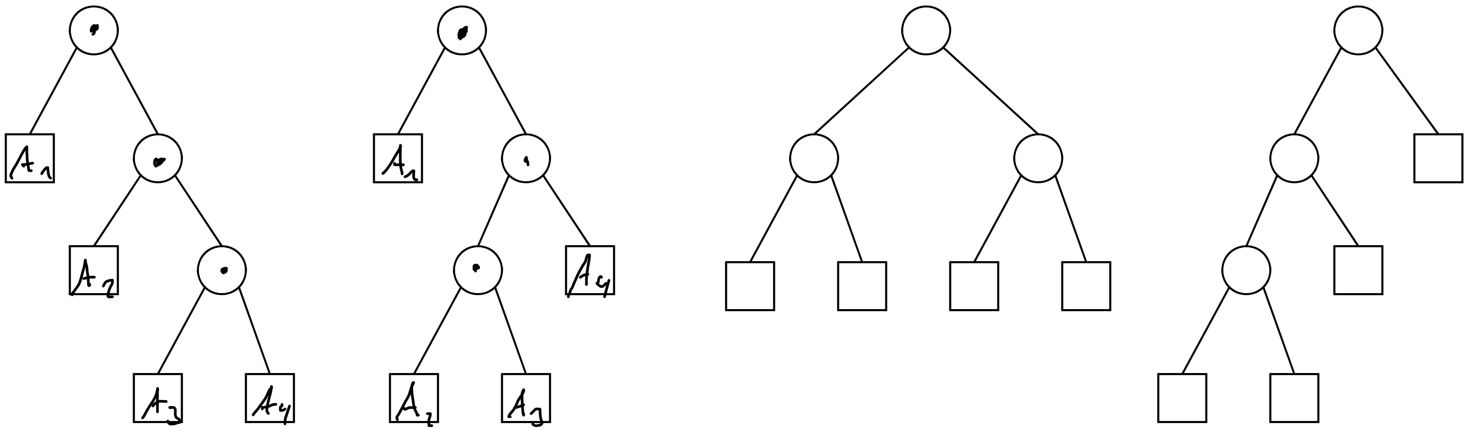
$$((A_1 A_2)(A_3 A_4))$$

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

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

Number of different parenthesizations

Different parenthesizations correspond to different trees:



$(A_1 \cdot ((A_2 \cdot A_3) \cdot A_4))$

$(A_1 (A_2 (A_3 \cdot A_4)))$

Number of different parenthesizations

Let *P(n)* be the number of alternative parenthesizations of the product $A_1 \ldots A_k A_{k+1} \ldots A_n$.

(A)

$$P(1) = 1$$

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

$((A_1 \cdot \ldots \cdot A_k) \cdot (A_{k+1} \cdot \ldots \cdot A_n))$

$P(k)$ $\quad$ $P(n-k)$

$$P(n+1) = \frac{1}{n+1}\binom{2n}{n} \approx \frac{4^n}{n\sqrt{\pi n}} + O\left(\frac{4^n}{\sqrt{n^5}}\right)$$

exponentially many possibilities

$$P(n+1) = C_n \quad n - \text{th Catalan number}$$

Remark: Determining the optimal parenthesization by exhaustive search is not reasonable.

Multiplying two matrices

$$A = (a_{ij})_{p \times q},\ B = (b_{ij})_{q \times r},\ A \cdot B = C = (c_{ij})_{p \times r},$$

$$c_{ij} = \sum_{k=1}^{q} a_{ik} b_{kj}.$$

Algorithm *Matrix-Mult*

Input: $(p \times q)$ matrix A, $(q \times r)$ matrix B

Output: $(p \times \mathrm{r})$ matrix $C = A \cdot B$

```
for i := 1 to p do
    for j :=1 to r do
        C[i, j] := 0
        for k := 1 to q do
            C[i, j] := C[i, j] + A[i, k] ·B[k,j]
```

Number of multiplications and additions: $p \cdot q \cdot r$

Remark: Using this algorithm, multiplying two $(n \times n)$ matrices requires n^3 multiplications. This can also be done using $O(n^{2.376})$ multiplications.

Matrix-chain multiplication: Example

Computation of the product $A_1 A_2 A_3$, where

A_1 : (10×100) matrix
A_2 : (100×5) matrix
A_3 : (5×50) matrix

a) Parenthesization $((A_1 \cdot A_2) \cdot A_3)$ requires

10×5 $A' = (A_1 \cdot A_2)$: $10 \cdot 5 \cdot 100 = 5000$

10×50 $A' A_3$: $10 \cdot 50 \cdot 5 = 2500$

Sum: 7500

Matrix-chain multiplication: Example

A_1 : (10×100) matrix
A_2 : (100×5) matrix
A_3 : (5×50) matrix

a) Parenthesization $(A_1 \cdot (A_2 \cdot A_3))$ requires

100×50: $A'' = (A_2 A_3)$: $100 \cdot 50 \cdot 5 = 25000$

10×50: $A_1 A''$: $10 \cdot 50 \cdot 100 = 50000$

Sum: 75000

Factor of 10 more multiplications.

Structure of an optimal parenthesization

$(A_{i...j}) = ((A_{i...k})\ (A_{k+1....j}))\quad i \le k < j$

$(A_1 \cdots A_k)(A_{k+1} \cdots A_n)$

$A_i \in [p_{i-1} \times p_i]$

Any optimal solution to the matrix-chain multiplication problem contains optimal solutions to subproblems.

$A_1 \cdot \ldots \cdot A_n$

$p_0\ p_1 \cdots p_n$

Determining an optimal solution recursively:

Let m[i,j] be the minimum number of operations needed to compute the product $A_{i...j}$:

$A_i \cdot \ldots \cdot A_j$

$p_{i-1}\ \boxed{A_{i...k}}^{\,p_k} \cdot \boxed{A_{k+1...j}}^{\,p_j}\ p_k \rightsquigarrow p_{i-1} \cdot p_k \cdot p_j$

m[i,j] = 0, if $i = j$

$A_i \cdot \ldots \cdot A_k \quad A_{k+1} \cdot \ldots \cdot A_j$

→ m[i,j] = $\min\limits_{i \le k < j} \{ m[i,k] + m[k+1, j] + p_{i-1} p_k p_j \}$, otherwise

→ s[i,j] = optimal splitting value k, i.e. the optimal parenthesization of $(A_{i...j})$ splits the product between A_k and A_{k+1}

Recursive matrix-chain multiplication

no dynamic programming yet

Algorithm *rec-mat-chain*(p, i, j)

Input: sequence $p = \langle p_0, p_1, \dots, p_n \rangle$,
where $(p_{i-1} \times p_i)$ is the dimensionen of matrix A_i

Invariant: rec-mat-chain(p, i, j) returns $m[i, j]$

1 **if** $i = j$ **then return** 0

2 m[i, j] := ∞

3 **for** $k := i$ **to** $j - 1$ **do**

4 $\quad m[i, j] := \min(m[i,j],\ p_{i-1} \cdot p_k \cdot p_j +$ rec-mat-chain(p, i, k) + rec-mat-chain($p, k+1, j$))

5 **return** $m[i, j]$

Initial call: rec-mat-chain(p,1, n)

Recursive matrix-chain multiplication: Running time

Let $T(n)$ be the time taken by rec-mat-chain(p,1,n).

$$T(1) \geq 1$$

$$T(n) \geq 1 + \sum_{k=1}^{n-1} \left(T(k) + T(n-k) + 1\right)$$

$$\geq n + 2\sum_{i=1}^{n-1} T(i)$$

$$\Rightarrow T(n) \geq 3^{n-1} \quad \text{(induction)}$$

Exponential running time!