BITS, PILANI – K. K. BIRLA GOA CAMPUS

Design & Analysis of Algorithms

(CS F364)

Lecture No. 6

Dynamic Programming Algorithm

The **dynamic-programming** algorithm can be broken into a sequence of **four steps.**

1. Characterize the structure of an optimal solution.

   Optimal Substructure Property

2. **Recursively define** the value of an optimal solution.
3. Compute the value of an optimal solution in a bottom-up fashion.

   Overlapping subproblems

4. Construct an optimal solution from computed information.

(not always necessary)

Matrix Chain Multiplication

- Thus, our **goal** today is:
- Given a **chain of matrices** to multiply,
  determine the **fewest number of multiplications**
  necessary to compute the product.
- Let $\mathbf{d_i x d_{i+1}}$ denote the dimensions of matrix A_i.
- Let $\mathbf{A = A_0\ A_1 \ldots A_{n-1}}$
- Let $\mathbf{N_{i,j}}$ denote the minimal number of multiplications necessary to find the product: $A_i\ A_{i+1} \ldots A_j$.
- To determine the minimal number of multiplications necessary $\mathbf{N_{0,n-1}}$ to find A,
- That is, determine how to parenthisize the multiplications

Matrix Chain Multiplication

Questions?

- **How many possible parenthesization?**
- **At least lower bound?**

The number of parenthesizations is atleast $\boldsymbol{\Omega(2^n)}$

Exercise: Prove

The **exact number** is given by the **recurrence relation**

$$T(n) = \sum_{k=1}^{n-1} T(k)T(n-k)$$

Because, the original product can be split into **two parts**

In **(n-1)** places.

Each split is to be parenthesized **optimally**

Matrix Chain Multiplication

Step1:

Optimal Substructure Property

If a particular parenthesization of the whole product is optimal,

then any sub-parenthesization in that product is optimal as well.

What does it mean?

- ***If*** **(A (B ((CD) (EF)))) is optimal**
- ***Then*** **(B ((CD) (EF))) is optimal as well**

How to Prove?

Matrix Chain Multiplication

- **Cut - Paste Argument**
  - **Because if it wasn't,**

  and say (((BC) (DE)) F) was better,

  then it would also follow that

  (A (((BC) (DE)) F)) was better than

  (A (B ((CD) (EF)))),

  - **contradicting** its **optimality!**

Matrix Chain Multiplication

Step 2:

Recursive Formulation

Let **M[i,j]** represent the minimum number of multiplications required for matrix product Ai ×⋯× Aj, For 1≤i≤j<n

High-Level Parenthesization for $A_{i..j}$

Notation: $A_{i..j} = A_i \times \ldots . \times A_j$

For any optimal multiplication sequence,

at the last step we are multiplying two matrices

$A_{i..k}$ and $A_{k+1..j}$ for **some k**, i.e.,

$$\mathbf{A_{i..j} = (A_i \times \ldots . \times A_k)(A_{k+1} \times \ldots . \times A_j) = A_{i..k} A_{k+1..j}}$$

Matrix Chain Multiplication

Thus,

$M[i, j] = M[i, k] + M[k+1, j] + d_{i-1}d_kd_j$

Thus the problem of determining the optimal sequence of multiplications is broken down to the **following question?**

How do we decide where to split the chain?

OR (what is k)?

Answer:

Search **all possible values of k** & take the **minimum** of it.

Matrix Chain Multiplication

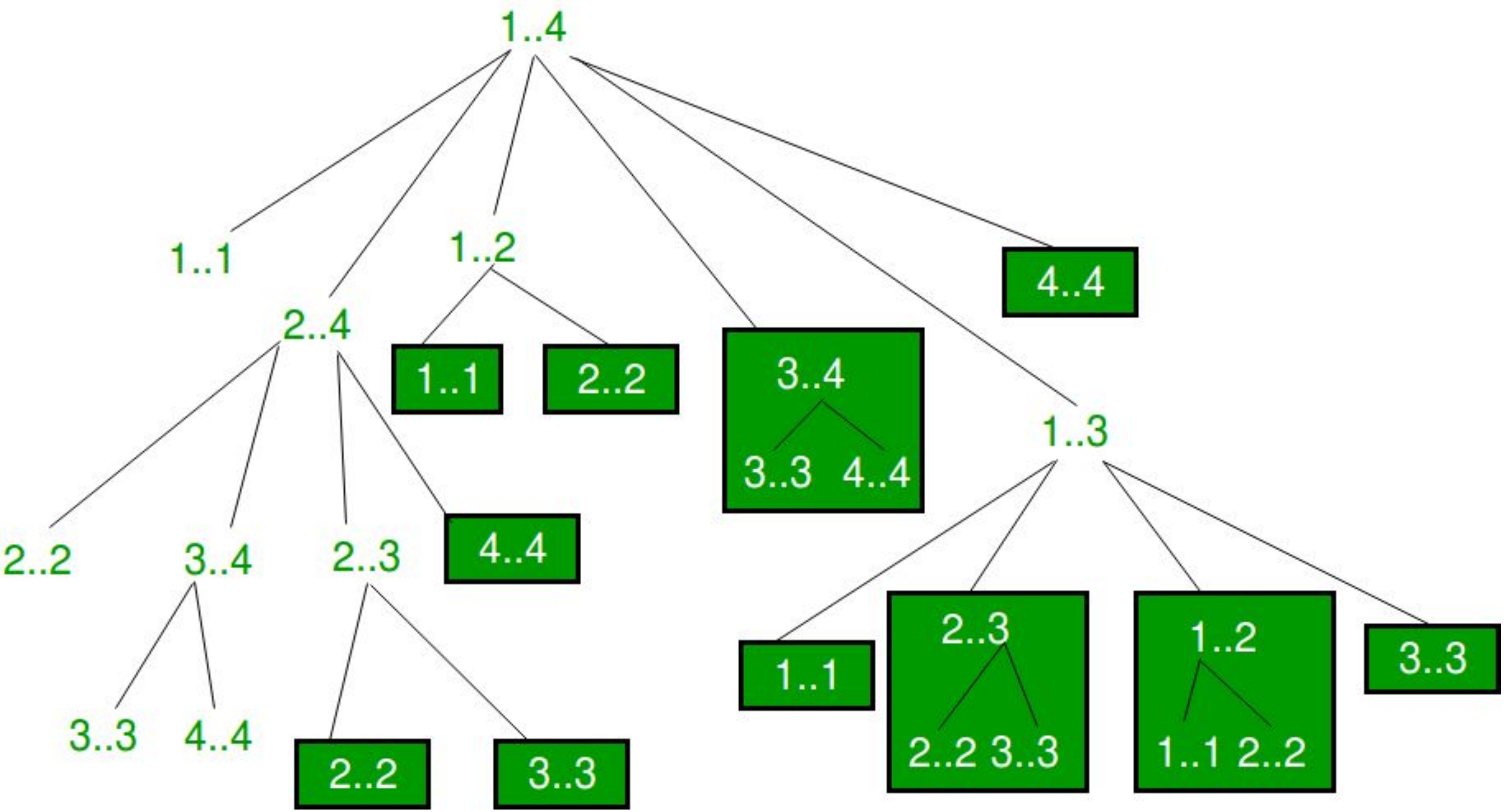
- ***Therefore,***

$$M[i,j] = \begin{cases} 0, if\ i = j \\ \min\limits_{i \le k < j} \{M[i,k] + M[k+1,j] + d_{i-1}\, d_k d_j\}, if\ i\ < j \end{cases}$$

Step3:

Compute the value of an optimal solution in a **bottom-up fashion**

Overlapping Subproblem



Matrix Chain Multiplication

Which sub-problems are necessary to solve first?

By Definition M[i,i] = 0

Clearly it's necessary to solve the smaller problems before the larger ones.

- In particular, we need to know **M[i, i+1]**, the number of multiplications to multiply any **adjacent pair** of matrices before we move onto larger tasks.

  Chains of length 1

- The next task we want to solve is finding all the values of the form **M[i, i+2],** then **M[i, i+3], etc.**

  Chains of length 2 & then **chains of length 3 & so on**

Matrix Chain Multiplication

That is, we calculate in the order

$$m[1,2], m[2,3], m[3,4], \ldots, m[n-3,n-2], m[n-2,n-1], m[n-1,n]$$

$$m[1,3], m[2,4], m[3,5], \ldots, m[n-3,n-1], m[n-2,n]$$

$$m[1,4], m[2,5], m[3,6], \ldots, m[n-3,n]$$

$$\vdots$$

$$m[1,n-1], m[2,n]$$

$$m[1,n]$$

Matrix Chain Multiplication

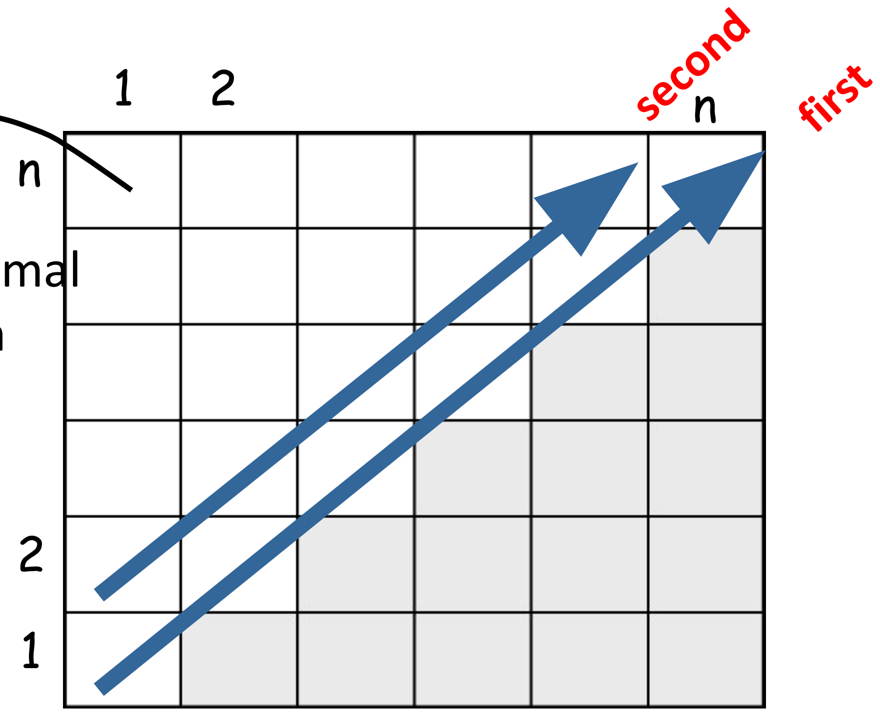
- This tells us the order in which to build the table:

  By diagonals

Diagonal indices:

- **On diagonal 0, $j=i$**
- **On diagonal 1, $j=i+1$**
- **On diagonal q, $j=i+q$**
- **On diagonal $n-1$, $j=i+n-1$**

Computing the table



M[1, n] gives the optimal solution to the problem

Values M[i, j] depend only on values that have been previously computed

Matrix Chain Multiplication

Example

- **Array dimensions:**
- A_1: 2 x 3 , A_2: 3 x 5 , A_3: 5 x 2
- A_4: 2 x 4 , A_5: 4 x 3

$$M[2,5] = min \begin{cases} M[2,2] + M[3,5] + d_1\, d_2 d_5 \\ M[2,3] + M[4,5] + d_1\, d_3 d_5 \\ M[2,4] + M[5,5] + d_1\, d_4 d_5 \end{cases}$$

Matrix Chain Multiplication

Table for M[i, j]

i, j	1	2	3	4	5
1	0	30	42	58	78
2		0	30	54	72
3			0	40	54
4				0	24
5					0

$$M[2,5] = min \begin{cases} M[2,2] + M[3,5] + d_1\, d_2 d_5 \\ M[2,3] + M[4,5] + d_1\, d_3 d_5 \\ M[2,4] + M[5,5] + d_1\, d_4 d_5 \end{cases} = \min \begin{cases} 0 + 54 + 45 = 99 \\ 30 + 24 + 18 = 72 \\ 54 + 0 + 36 = 90 \end{cases} = \min \begin{cases} (A_2)(A_3 A_4 A_5) \\ (A_2 A_3)(A_4 A_5) \\ (A_2 A_3 A_4)(A_5) \end{cases}$$

Matrix Chain Multiplication

Optimal locations for parentheses:

Idea: Maintain an array ***s[1….n,1….n]*** where ***s[i, j]*** denotes ***k*** for the optimal splitting in computing $\mathbf{A_{i..j} = A_{i..k}\ A_{k+1..j}}$

Array ***s[1….n,1….n]*** can be used recursively to compute the multiplication sequence

$s[1,n]$	$(A_1 \cdots A_{s[1,n]})(A_{s[1,n]+1} \cdots A_n)$
$s[1,s[1,n]]$	$(A_1 \cdots A_{s[1,s[1,n]]})(A_{s[1,s[1,n]]+1} \cdots A_{s[1,n]})$
$s[s[1,n]+1,n]$	$(A_{s[1,n]+1} \cdots A_{s[s[1,n]+1,n]})\times$ $(A_{s[s[1,n]+1,n]+1} \cdots A_n)$
$\vdots$	$\vdots$

Do this recursively until the multiplication sequence is determined.

Matrix Chain Multiplication

Table for s[i, j]

i, j	1	2	3	4	5
1		1	1	3	3
2			2	3	3
3				3	3
4					4
5					

The multiplication sequence is recovered as follows.

s[1, 5] = 3 $(A_1 A_2 A_3)(A_4 A_5)$

s[1, 5] = 1 $(A_1 (A_2 A_3))$

Hence the final multiplication sequence is

$(A_1 (A_2 A_3))(A_4 A_5)$

Matrix Chain Multiplication

Pseudo code

Matrix-Chain-Order(p)

```
n ← length[p] − 1
for i ← 1 to n
        do m[i, i] ← 0
for l ← 2 to n                         ▷ l is the chain length.
        do for i ← 1 to n − l + 1
                do j ← i + l − 1
                   m[i, j] ← ∞
                   for k ← i to j − 1
                        do q ← m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j
                           if q < m[i, j]
                              then m[i, j] ← q
                                   s[i, j] ← k
return m and s
```

Construct optimal sub-problems "bottom-up." and remember them.

M[i, i]'s are easy, so start with them

M[i, i]'s are easy, so start with them

Running time: $O(n^3)$

Observations

Question 1

How many subproblems are used in an optimal solution for the original problem?

- Coin Change Problem

  Two　　Which?

- Matrix Chain Multiplication

  Two　　Which?

Question 2

How many choices we have in determining which subproblems to use in an optimal solution?

- Coin Change Problem

  Two

  Why?

- Matrix Chain Multiplication

  $j - i$ choices for k (splitting the product)

DP – Time Complexity

Intuitively, the running time of a dynamic programming algorithm depends on two factors:

1. Number of subproblems overall
2. How many choices we have for each subproblem

Matrix multiplication:

- $O(n^2)$ subproblems
- At most $n-1$ choices

Therefore, Time complexity is $O(n^3)$ overall