W, 09/25/19

Fall'19 CSCE 629

Analysis of Algorithms

Fang Song

Texas A&M U

Lecture 11

- Elements of DP
- Matrix-chain multiplication

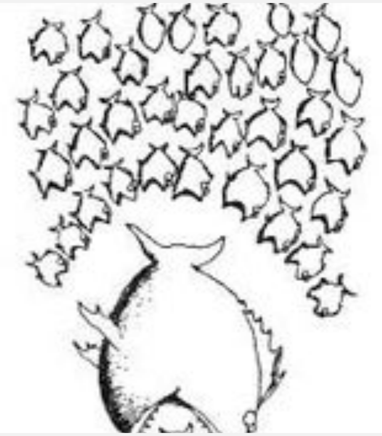
Credit: based on slides by K. Wayne

Dynamic Programming: recap

- Break up a problem into a series of overlapping subproblems
- There is an ordering on the subproblems, and a relation showing how to solve a subproblem given answers to "smaller" ones.

An implicit DAG: nodes=subproblems, edges = dependencies

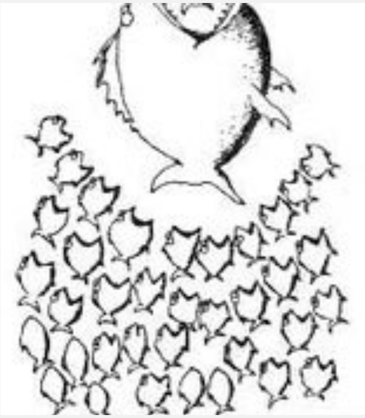
Top-down



Credit: Mary Wootters

- **DP is about smart recursion (i.e. without repetition) by momoization**

- **Usually easy to express by building up a table iteratively**



Bottom-up

A DP recipe

1. Formulate the problem recursively (key step!)
   a) Specification. Describe what problems to solve (not how)
   b) Recursion. Give a recursive formula for the whole problem in terms answers to smaller instances of the same problem
   c) Step back and double check!

2. Build solutions to your recurrence (kinda routine)
   a) Identify subproblems
   b) Choose a memoization data structure
   c) Identify dependencies and find a good order (DAG in topological order)
   d) Write down your algorithm
   e) Analyze time (and space)
   f) Further improvements if possible

We usually go with bottom-up approach in this class

Matrix chain multiplication

In which **order** to multiply a sequence of rectangular matrices?

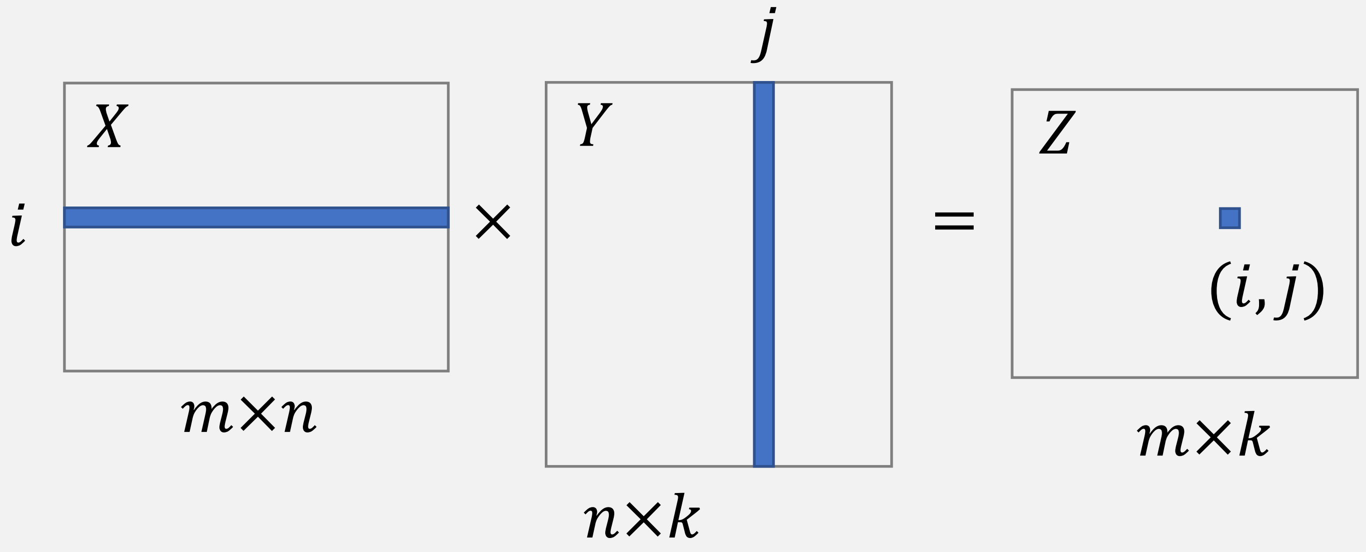
$A \times (B \times C)$ vs. $(A \times B) \times C$

- Both correct: associativity
- Does the order matter?
- What we care: # of multiplications of numbers

Review: matrix multiplication

- Matrices $X_{m\times n}$ and $Y_{n\times k}$

Compatible: # column(X) = # row(Y)



$$z_{ij} = \sum_{\ell=1}^{n} x_{i\ell}\, y_{\ell j}$$

n multiplications of real numbers

- Computing $Z = X_{m\times n} Y_{n\times k}$: mnk scaler multiplications

Let's forget about Strassen's divide-&-conquer algorithm for now

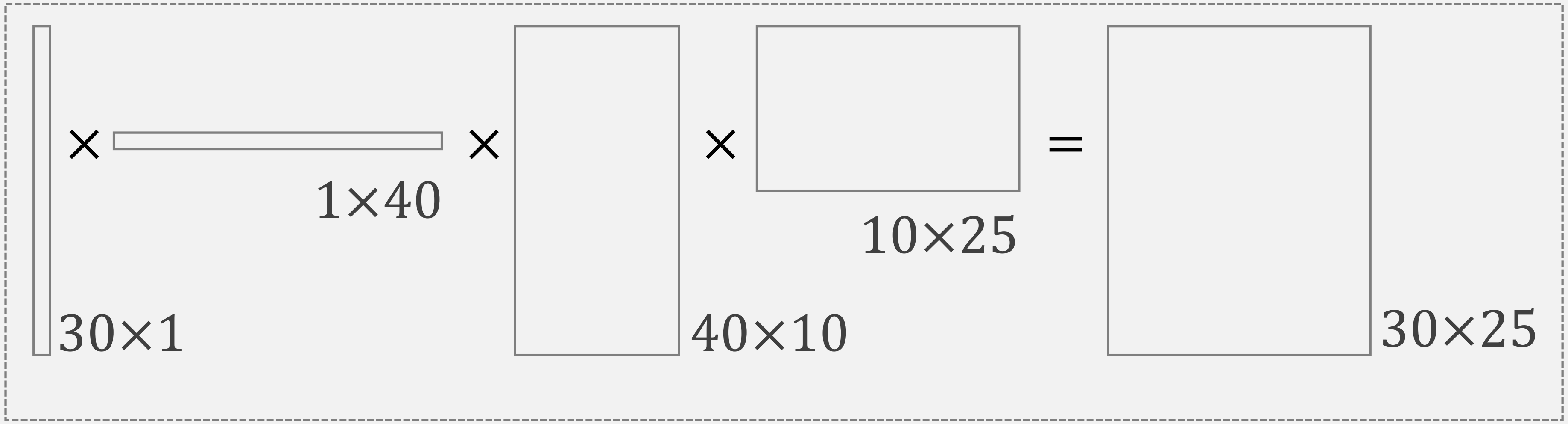
Why order matters …

$A: 30\times1$

$B: 1\times40$

$C: 40\times10$

$C: 10\times25$



$((AB)(CD))$ vs. $(A((BC)D))$

41200 scalar mult.

1400 scalar mult.

Matrix chain order problem

- Input. Matrices $A_1, \ldots, A_n$
  - A_i size $d_{i-1} \times d_i$
- Output. Optimal order for computing $\Pi_i A_i$
  - Minimum # of scalar multiplications

- Brute-force

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

  - Exercise. Prove $P(n) = \Omega(2^n)$

DP1: develop a recursion

- Input. Matrices $A_1, \ldots, A_n$
  - A_i size $d_{i-1} \times d_i$
- Output. Optimal order for computing $\Pi_i A_i$
  - Minimum # of scalar multiplications

$1a.$ specification

Def. $M(i,j)$ = min # of mult. needed to compute $P_{ij} := A_i A_{i+1} \ldots A_j$

- Goal. Find $M(1,n)$
- Basis: $M(i,i) = 0$
- Recursion: how to define $M(i,j)$ recursively?

$1b.\, b$ recursion

DP1: develop a recursion

- Assuming optimal order divides at k $A_i \dots A_j$

$$(P_{ik})_{d_{i\_1} \times d_k} \quad (A_i \dots A_k)(A_{k+1} \dots A_j) \quad (P_{k+1j})_{d_k \times d_j}$$

$$M(i,k) \qquad M(k+1,j)$$

  - Cost to compute P_{ik}: $M(i,k)$
  - Cost to compute P_{k+1j}: $M(k+1,j)$
  - Cost to compute: $P_{ik} \times P_{kj}$: $d_{i-1} \times d_k \times d_j$

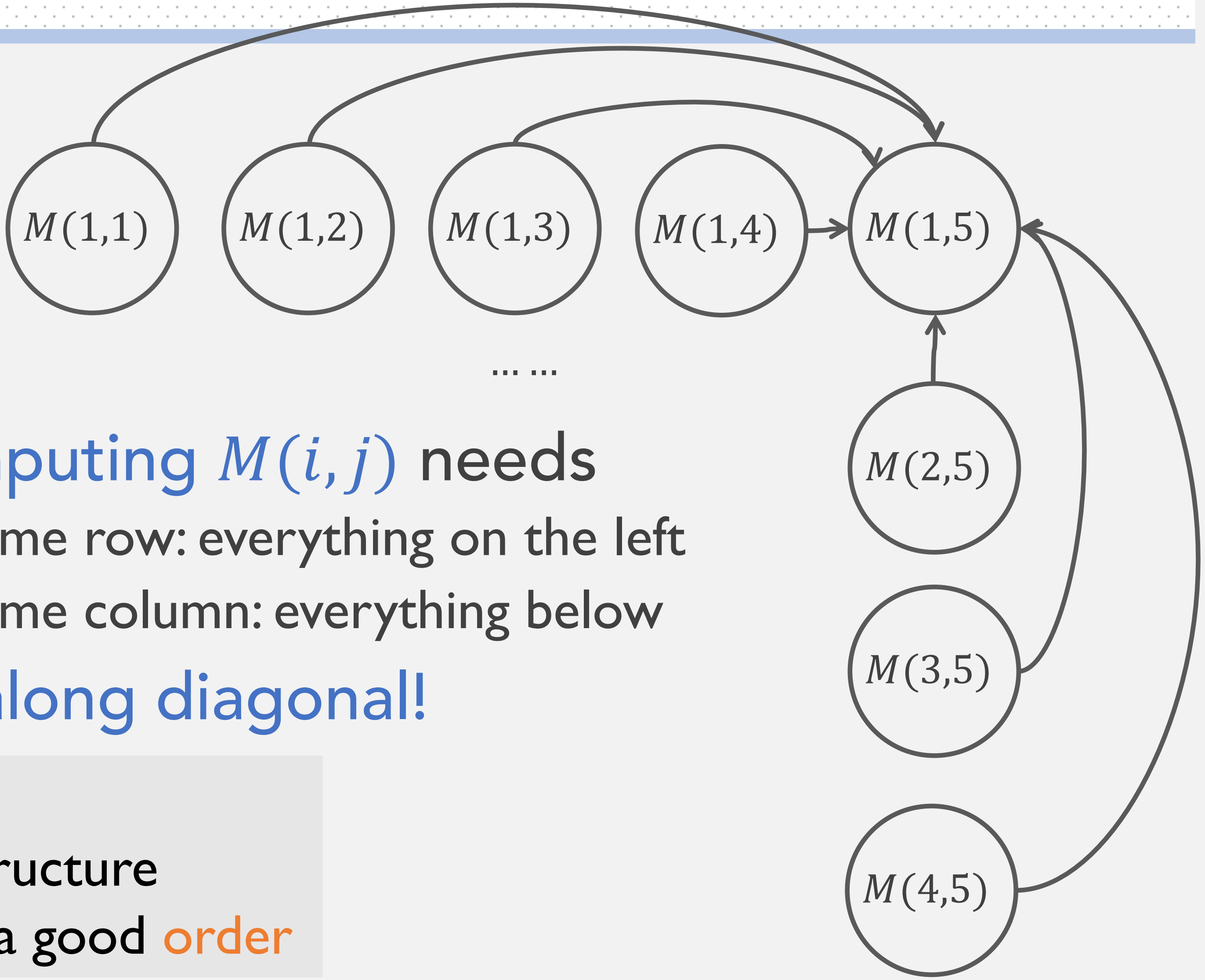
computing $Z = X_{m \times n} Y_{n \times k}$: mnk scaler multiplications

$$M(i,j) = \begin{cases} 0 & \text{if } i = j \\ \min_{i \le k < j} \{M(i,k) + M(k+1,j) + d_{i-1} d_k d_j\} & \text{otherwise} \end{cases}$$

- How many subproblems in total? $O(n^2)$

DP2: build solutions bottom up

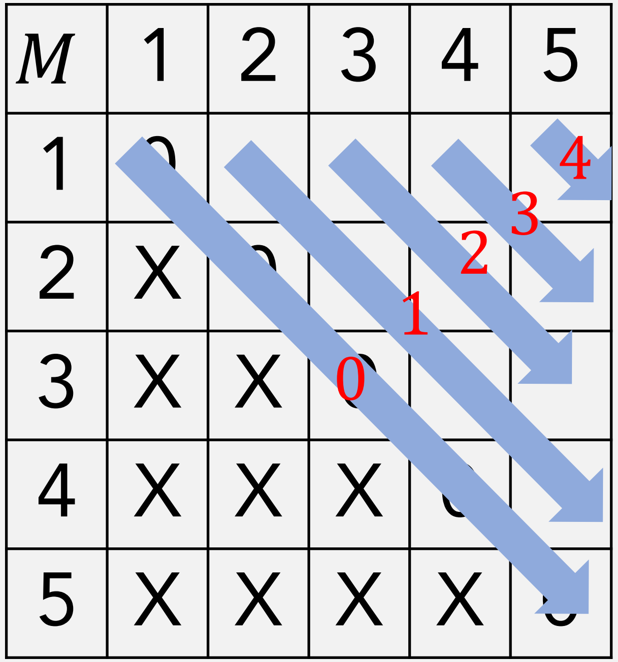
M	1	2	3	4	5
1	0				
2	X	0			
3	X	X	0		
4	X	X	X	0	
5	X	X	X	X	0



- Computing $M(i,j)$ needs
  - Same row: everything on the left
  - Same column: everything below
- Go along diagonal!

2a. Identify subproblems
2b. Choose a memoization data structure
2c. Identify dependencies and find a good order

DP2: build solutions bottom up



MatrixChain (n)

// $\boldsymbol{M(i,j)}$ memoize subproblem values

For $i = 2, \dots, n$

    $M[i,i] \leftarrow 0$

For $l = 1, \dots, n-1$ // diagonals

    For $i = 1, \dots, n-l$ //row

        $j = i + l$ // the column of row i on l-th diag.

        $M[i,j] \leftarrow \infty$

        For $k = i, \dots j-1$

            $M[i,j] = min\{M[i,j], M[i,k] + M[k+1,j] + d_{i-1}d_k d_j\}$

2d. Write down your algorithm
2e. Analyze time (and space)

- Running time: $O(n^3)$

Example

	1	2	3	4
1	0	1200	700	1400
2	X	0	400	650
3	X	X	0	10,000
4	X	X	X	0

$A_1: 30\times1$

$A_2: 1\times40$

$A_3: 40\times10$

$A_4: 10\times25$

$M(1,2) = 30\times1\times40$

$M(2,3) = 1\times40\times10$

$M(3,4) = 40\times10\times25$

$$M(1,3) = \min\{M(1,2) + 30\times40\times10,\\ M(2,3) + 30\times1\times10\}$$

$$M(2,4) = \min\{M(2,3) + 1\times10\times25,\\ M(3,4) + 1\times40\times25\}$$

$M(1,4) = \min\{\ldots\}$

DP3: constructing an optimal solution

```
MatrixChain (n)
// M(i, j) memoize subproblem values
// S[i, j] memoize optimal split index
……
For l =
    For i = 1, … , n − l
        j = i + l // the column of row i on l-th diag.
        For k = i, … j − 1
            M[i, j] = min{M[i, j], M[i, k] + M[k + 1, j], d_{i−1} d_k d_j}
        Record the optimal k: S[i, j] ← k
```

- **Exercise.** Find the optimal order of multiplication from S