

## CS432: Matrix Multiplication

- Reading: Section 8.3
- Note: see code in the book
  - It implements this strategy

## Multiplying a Sequence of Matrices

- Multiplying two  $n \times n$  matrices
  - Costs  $n^3$  scalar multiplications
- Multiplying an  $m \times k$  matrix by a  $k \times n$  matrix?
  - Costs  $m \times k \times n$  scalar multiplications
  - $C = AxB$ , means  
 $C[i][j] = \text{Sum}\{p=1 \text{ to } k\} A[i][p] \times B[p][j]$
- What if more than two matrices?
  - ABC could be calculated (AB)C or A(BC)
  - Same answer, but the number of scalar multiplications will probably vary

## Multiplying a Sequence of Matrices

- Example:  
A is  $300 \times 200$ . B is  $200 \times 100$ . C is  $100 \times 300$ .
- Cost for (AB)C
  - First, do (AB): Cost:  $300 \times 200 \times 100 = 6,000,000$
  - Next, that times C: Cost:  $300 \times 100 \times 300 = 9,000,000$
  - **Total cost:** 15,000,000
- Cost for A(BC)
  - First, do (BC): Cost:  $200 \times 100 \times 300 = 6,000,000$
  - Next, A times that: Cost:  $300 \times 200 \times 300 = 18,000,000$
  - **Total cost:** 24,000,000
- So: Order affects cost (but not the result)

## Problem Statement

- Given a sequence of  $n$  matrices:  
 $M_1, M_2, \dots, M_n$   
determine the best way to group these by pairs to minimize the number of scalar multiplications
  - Note: in what follows,  $r_i$  is the row-dimension of matrix  $i$ , and  $c_j$  is the column dimension of matrix  $j$
  - So  $M_i$  has dimensions  $[r_i \times c_i]$
- Brute force approach? Find all possible groupings. But there are  $4^{n-2}/(n-1)^2$  groupings
  - Trust us on this (or see text and its exercise 11)

## Recursive Definition

- Let  $s(i,j)$  be the minimum number of scalar multiplications needed to multiply the sequence from  $M_i$  through  $M_j$ 
  - Then  $s(1,n)$  is the answer for the whole problem
- What if  $k$  were some point between  $i$  and  $j$ ?
  - $(M_i \times \dots \times M_k) \times (M_{k+1} \times \dots \times M_j)$
  - To multiply these two partial results costs:  $r_i \times c_k \times c_j$
  - So the cost for all of it would be:  
 $s(i,k) + s(k+1,j) + r_i \times c_k \times c_j$
- Which value of  $k$  to use? Try all values from  $i$  upto  $j$ 
  - $s(i,j) = \min\{i \leq k < j\} (s(i,k) + s(k+1,j) + r_i \times c_k \times c_j)$

## Why Dynamic Programming Here?

- Subproblems overlap
  - Subproblems for  $s(2,6)$  include  $s(2,4)$  and  $s(3,5)$
- Repeating subproblems
  - E.g. Larger problem  $s(2,6)$  requires solutions to smaller problem  $s(3,5)$
  - But so will larger problem  $s(3,7)$
- Are there easy base cases? Sure!
  - $s(i,i+1)$  is the cost for two adjacent matrices
  - Only one way to do multiply two and it costs:  
 $r_i \times c_i \times c_j$
  - Also,  $s(i,i)$  is a base case: 0 scalar multiplications

## Table for Matrix Multiplication

- A table for  $s(i,j)$ 
  - Desired answer is  $s(1,n)$ , which is top-right corner
  - Base cases,  $s(i,i) = 0$ . This is the "main" diagonal.
  - Adjacent pairs,  $s(i,i+1)$ . The diagonal just above the "main" diagonal.
- Strategy:
  - Work from main diagonal towards top-right corner
  - Fill in by diagonals, until we hit top-right corner
  - For each cell, calculate for the appropriate number of values of  $k$ 
    - Uses previously calculated results
    - See next slides

## Example

- $n=4$ , with dimensions:  $[30 \times 1]$ ,  $[1 \times 40]$ ,  $[40 \times 10]$ ,  $[10 \times 25]$
- Table initially:

	j=1	2	3	4
i=1	0			
2		0		
3			0	
4				0

## Costs for Adjacent Matrices?

- Dimensions:  $[30 \times 1]$ ,  $[1 \times 40]$ ,  $[40 \times 10]$ ,  $[10 \times 25]$
- Cost to multiply adjacent matrices?
  - $s(1,2) = 30 \times 1 \times 40 = ?$
  - $s(2,3) = 1 \times 40 \times 10 = ?$
  - $s(3,4) = 40 \times 10 \times 25 = ?$

	j=1	2	3	4
i=1	0	?		
2		0	?	
3			0	?
4				0

## Costs for Adjacent Matrices? Answers

- Dimensions:  $[30 \times 1]$ ,  $[1 \times 40]$ ,  $[40 \times 10]$ ,  $[10 \times 25]$
- Cost to multiply adjacent matrices?
  - $s(1,2) = 30 \times 1 \times 40 = 1200$
  - $s(2,3) = 1 \times 40 \times 10 = 400$
  - $s(3,4) = 40 \times 10 \times 25 = 10,000$

	j=1	2	3	4
i=1	0	1200		
2		0	400	
3			0	10,000
4				0

## Costs for Sequences of Three?

- Dimensions:  $[30 \times 1]$ ,  $[1 \times 40]$ ,  $[40 \times 10]$ ,  $[10 \times 25]$
- Next diagonal is all seq. of three:  $s(1,3)$  and  $s(2,4)$
- Reminder:
 
$$s(i,j) = \min_{\{1 \leq k < j\}} (s(i,k) + s(k+1,j) + r_i \times c_k \times c_j)$$
- For  $s(1,3)$ ,  $\min_{\{1 \leq k < 3\}} (s(1,k) + s(k+1,3) + r_1 \times c_k \times c_3)$ 
  - $k=1$ :  $s(1,1) + s(2,3) + 30 \times 1 \times 10 = 0 + 400 + 300 = 700$ 
    - This is cost of  $M_1(M_2M_3)$
  - $k=2$ :  $s(1,2) + s(3,3) + 30 \times 40 \times 10 = 1200 + 0 + 12,000 = 13,200$ 
    - This is cost of  $(M_1M_2)M_3$
  - The min?  $k=1$ , cost is 700
- For  $s(2,4)$ ,  $\min_{\{2 \leq k < 4\}} (s(2,k) + s(k+1,4) + r_2 \times c_k \times c_4)$ 
  - Compares  $k=2$ ,  $M_2(M_3M_4)$ , with  $k=3$ ,  $(M_2M_3)M_4$
  - Do on your own. Answer is:  $k=3$ , cost is 650

## Consider $s(1,3)$ . Two choices! $k=1$

- Dimensions:  $[30 \times 1]$ ,  $[1 \times 40]$ ,  $[40 \times 10]$ ,  $[10 \times 25]$
- Reminder:
 
$$s(i,j) = \min_{\{i \leq k < j\}} (s(i,k) + s(k+1,j) + r_i \times c_k \times c_j)$$
 so for  $s(1,3)$ 

$$\min_{\{1 \leq k < 3\}} (s(1,k) + s(k+1,3) + r_1 \times c_k \times c_3)$$
- So for  $k=1$ , look at  $s(1,1)$  and  $s(2,3)$

	j=1	2	3	4
i=1	0	1200	700	
2		0	400	
3			0	10,000
4				0

### Consider $s(1,3)$ . Two choices! $k=2$

- Dimensions:  $[30 \times 1]$ ,  $[1 \times 40]$ ,  $[40 \times 10]$ ,  $[10 \times 25]$
- Reminder:
 
$$s(i,j) = \min_{\{1 \leq k < j\}} \{s(i,k) + s(k+1,j) + r_i \times c_k \times c_j\}$$
 so for  $s(1,3)$ 

$$\min_{\{1 \leq k < 3\}} \{s(1,k) + s(k+1,3) + r_1 \times c_k \times c_3\}$$
- So for  $k=2$ , look at  $s(1,2)$  and  $s(3,3)$

	j=1	2	3	4
i=1	0	1200	700	
2		0	400	
3			0	10,000
4				0

Diagram showing arrows from  $k=1$  to  $s(1,2)$  and  $s(3,3)$ , and from  $k=2$  to  $s(1,3)$ .

### Costs for Sequences of Three? Answers

- From last slide:  $s(1,3) = 700$  and  $s(2,4) = 650$

	j=1	2	3	4
i=1	0	1200	700 ( $k=1$ )	
2		0	400	650 ( $k=3$ )
3			0	10,000
4				0

### Sequences of Four?

- Dimensions:  $[30 \times 1]$ ,  $[1 \times 40]$ ,  $[40 \times 10]$ ,  $[10 \times 25]$
- Next diagonal is all seq. of four:  $s(1,4)$ , our goal!
- Reminder:
 
$$s(i,j) = \min_{\{1 \leq k < j\}} \{s(i,k) + s(k+1,j) + r_i \times c_k \times c_j\}$$
 so for this step:
 
$$s(1,4) = \min_{\{1 \leq k < 4\}} \{s(1,k) + s(k+1,4) + r_1 \times c_k \times c_4\}$$
- $k=1$  is best cost for  $M_1(M_2M_3M_4)$ 
  - $s(1,1) + s(2,4) + 30 \times 1 \times 25 = 0 + 650 + 750 = 1400$
- $k=2$  is best cost for  $(M_1M_2)(M_3M_4)$ 
  - $s(1,2) + s(3,4) + 30 \times 40 \times 25 = 1200 + 10,000 + 30,000 = 41,200$
- $k=3$  is best cost for  $(M_1M_2M_3)M_4$ 
  - $s(1,3) + s(4,4) + 30 \times 10 \times 25 = 700 + 0 + 7500 = 8200$
- Minimum:  $k=1$ , 1400. (Note how much better!)

### Costs for Sequences of Four? Answer

- From last slide:  $s(1,4) = 1400$  for  $k=1$

	j=1	2	3	4
i=1	0	1200	700 ( $k=1$ )	1400 ( $k=1$ )
2		0	400	650 ( $k=3$ )
3			0	10,000
4				0

### Finding the Actual Grouping

- This table finds  $s(1,n)$  but how do we remember the groupings
  - Solution: for each cell, remember the value of  $k$  that was the minimum
  - Keep them in a separate table
- Use these to "back-track" through the  $s(i,j)$  table
- Our example:
  - For  $s(1,4)$ , it was  $k=1$ . Thus  $M_1(M_2M_3M_4)$  was best.
  - Look at position  $(2,4)$ . There is was  $k=3$ . Thus  $(M_2M_3)M_4$  was the best way to do  $(M_2M_3M_4)$
  - So grouping was:  $M_1((M_2M_3)M_4)$  with cost=1400

### Complexity for Dyn. Programming Solution

- Time complexity
  - Certainly not exponential!
  - The code (see book) has three nested loops
  - But consider:
    - We fill in  $n(n-1)/2$  cells above the main diagonal
    - For each, we  $O(n)$  calculations as  $k$  varies between  $i$  and  $j$
  - Overall:  $\Theta(n^3)$
  - BTW, there exists a  $\Theta(n^2)$  solution. (Frances Yao, "Speed-up in Dynamic Programming", 1982.)
- Space complexity
  - $\Theta(n^2)$