# CS 362, Lecture 7

Jared Saia
University of New Mexico

## Today's Outline

- Matrix Multiplication



## Matrix Chain Multiplication

Problem:

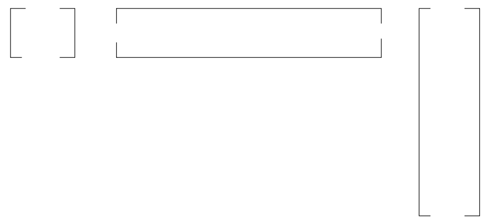
- We are given a sequence of $n$ matrices, $A_1, A_2, \ldots, A_n$, where for $i = 1, 2, \ldots, n$, matrix $A_i$ has dimension $p_{i-1}$ by $p_i$
- We want to compute the product, $A_1 A_2, \ldots, A_n$ as quickly as possible.
- In particular, we want to fully *paranthesize* the expression above so there are no ambiguities about the how the matrices are multiplied
- A product of matrices is *fully parenthisized* if it is either a single matrix, or the product of two fully parenthesized matrix products, sorrounded by parantheses



## Paranthesizing Matrices

- There are many ways to paranthesize the matrices
- Each way gives the same output (because of associativity of matrix multiplications)
- However the way we paranthesize will effect the *time* to compute the output
- Our Goal: Find a paranthesization which requires the minimal number of scalar multiplications

## Example



- In this example, it's much better to multiply the last two matrices first (this gives us a short, narrow matrix on the right)
- Worse to multiply the first two matrices first (this gives us a short wide matrix on the left)
- In general, our goal is to find ways to always create narrow and short resulting matrices.



## A Problem

Problem: There can be many ways to paranthesize. E.g.

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



## A Problem

- Let $P(n)$ be the number of ways to paranthesize $n$ matrices. Then $P(1) = 1$
- For $n \geq 2$, we know that a fully paranthesized product is the product of two fully paranthesized products, and the split can occur anywhere from $k = 1$ to $k = n - 1$.
- Hence for $n \geq 2$:

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

- In the hw, you will show that the solution to this recurrence is $\Omega(2^n)$



## The Pattern

Q: Can we develop a DP Solution to this problem?

- **Formulate the problem recursively.**. Write down a formula for the whole problem as a simple combination of answers to smaller subproblems
- **Build solutions to your recurrence from the bottom up.** Write an algorithm that starts with the base cases of your recurrence and works its way up to the final solution by considering the intermediate subproblems in the correct order.

## Key Observation

- Let $A_{i..j}$ (for $i \leq j$) be the matrix that results from evaluating the product $A_i A_{i+1}, \ldots A_j$
- Imagine we are computing $A_{i..j}$
- The last multiplication we do must look like this:

$$A_{i..j} = (A_{i..k}) * (A_{k+1..j})$$

  for some $k$ between $i$ and $j - 1$
- Then total cost to compute $A_{i..j}$ is:

  cost to compute $A_{i..k}$ +
  cost to compute $A_{k+1..j}$ +
  cost to multiply $A_{i..k}$ and $A_{k+1..j}$



## Recursive Formulation

- For any integers $x, y$, let $m(x, y)$ be the minimum cost of computing $A_{x..y}$
- Then for any $k$ between $i$ and $j - 1$,

$$\begin{aligned} m(i,j) \quad \leq \quad & \text{optimal cost to compute } A_{i..k} + \\ & \text{optimal cost to compute } A_{k+1..j} + \\ & \text{cost to multiply } A_{i..k} \text{ and } A_{k+1..j} \end{aligned}$$

- In other words:

$$\begin{aligned} m(i,j) \quad \leq \quad & m(i,k) + \\ & m(k+1,j) + \\ & \text{cost to multiply } A_{i..k} \text{ and } A_{k+1..j} \end{aligned}$$



## Cost to Multiply

- $A_{i..k}$ is a $p_{i-1}$ by $p_k$ matrix
- $A_{k+1..j}$ is a $p_k$ by $p_j$ matrix
- Thus multiplying $A_{i..k}$ and $A_{k+1..j}$ takes $p_{i-1}p_kp_j$ operations
- Hence we have:

$$\begin{aligned} m(i,j) \quad \leq \quad & m(i,k) + \\ & m(k+1,j) + \\ & p_{i-1}p_kp_j \end{aligned}$$



## Recursive Formulation

- We've shown that $m(i,j) \leq m(i,k) + m(k+1,j) + p_{i-1}p_kp_j$ for any $k = i, i+1, \ldots, j-1$
- Further note that the optimal parenthesization must use some value of $k = i, i+1, \ldots, j-1$. So we need only pick the best
- Thus we have:

$$\boxed{\begin{aligned} & m(i,j) = 0 \text{ if } i = j \\ & m(i,j) = \min_{i \leq k < j}\{m(i,k) + m(k+1,j) + p_{i-1}p_kp_j\} \end{aligned}}$$

## The Recursive Algorithm

- We now have enough information to write a recursive function to solve the problem
- The recursive solution will have runtime given by the following recurrence:
- $T(1) = 1$,
- $T(n) = 1 + \sum_{k=1}^{n-1}(T(k) + T(n-k) + 1)$
- Unfortunately, the solution to this recurrence is $\Omega(2^n)$ (as shown on p. 346 of the text)



## DP Algorithm

- Note that we must solve one subproblem for each choice of $i$ and $j$ satisfying $1 \leq i \leq j \leq n$
- This is only $\binom{n}{2} + n = \Theta(n^2)$ subproblems
- The recursive algorithm encounters each subproblem many times in the branches of the recursion tree.
- However, we can just compute these subproblems from the bottom up, storing the results in a table (this is the DP solution)



## Pseudocode

```
Matrix-Chain-Order(int p[]){
  n = p.length - 1;
  for (i=1;i<=n;i++){
    m(i,i) = 0;
  }
  for (l=2;l<=n;l++){ \\l is chain length
    for (i=1;i<=n-l+1;i++){
      j = i+l-1;
      m[i,j] = MAXINT;
      for(k=i;k<=j-1;k++){
        q = m[i,k] + m[k+1,j] + p[i-1]*p[k]*p[j];
        if(q<m[i,j]){
          m[i,j] = q;
          s[i,j] = k;
        }
      }}}}
```



## Psuedocode

- This code computes both the optimal cost and a parenthesization that achieves that cost
- It uses an $m$ array to store the optimal costs of computing $m(i,j)$. It also uses a $s$ array, where $s(i,j)$ stores the $k$ value which gives $m(i,j)$
- The parenthesization can be recovered from the $s$ array using the pseudocode in the book on p. 338.

## Analysis

- This code has three nested loops, each of which takes on at most $n-1$ values, and the inner loop takes $O(1)$ time.
- Thus the runtime is $O(n^3)$
- The algorithm also requires $\Theta(n^2)$ space



## Example

- Consider the sequence of three matrices, $A_1, A_2, A_3$ whose dimensions are given by the sequence $3, 1, 2, 1$ (i.e. $p_0 = 3$, $p_1 = 1$, $p_2 = 2$, $p_3 = 1$)
- Let's construct the tables giving the optimal parenthesization
- The $(i, j)$ entry of the first table will give the optimal cost for computing $A_{i..j}$, the $(i, j)$ entry of the second table will give a $k$ value which achieves this optimal cost



## Computations

- $m(1,1) = m(2,2) = m(3,3) = 0$
- $m(1,2) = p_0 p_1 p_2 = 6$
- $m(2,3) = p_1 p_2 p_3 = 2$



## Computations

$$\begin{aligned} m(1,3) &= \min \left\{ \begin{array}{l} m(1,1) + m(2,3) + p_0 p_1 p_3), \\ m(1,2) + m(3,3) + p_0 p_2 p_3) \end{array} \right\} \\ &= \min \left\{ \begin{array}{l} 0 + 2 + 3, \\ 6 + 0 + 6 \end{array} \right\} \\ &= 5 \end{aligned}$$

## Example, m array

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0 | 6 | 5 |
| 2 | - | 0 | 2 |
| 3 | - | - | 0 |



## Example, s array

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | - | 1 | 1 |
| 2 | - | - | 2 |
| 3 | - | - | - |



## Example

- Thus an optimal parenthesization is $(A_1(A_2A_3))$
- The cost of this is 5



## Example II

- Consider the sequence of three matrices, $A_1, A_2, A_3, A_4$ whose dimensions are given by the sequence $3, 1, 2, 1, 2$ (i.e. $p_0 = 3$, $p_1 = 1$, $p_2 = 2$, $p_3 = 1$, $p_4 = 2$)
- Let's construct the tables giving the optimal parenthesization
- The $(i, j)$ entry of the first table will give the optimal cost for computing $A_{i..j}$, the $(i, j)$ entry of the second table will give a $k$ value which achieves this optimal cost

## Example II, m array

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 6 | 5 | 10 |
| 2 | - | 0 | 2 | 4 |
| 3 | - | - | 0 | 4 |
| 4 | - | - | - | 0 |



## Example II, s array

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 1 | 1 | 1 |
| 2 | - | - | 2 | 3 |
| 3 | - | - | - | 3 |
| 4 | - | - | - | - |



## Example Computation

$$
\begin{aligned}
m(1,4) &= \min \left\{ \begin{array}{l} m(1,1)+m(2,4)+p_0p_1p_4), \\ m(1,2)+m(3,4)+p_0p_2p_4), \\ m(1,3)+m(4,4)+p_0p_3p_4) \end{array} \right\} \\
&= \min \left\{ \begin{array}{l} 0+4+6, \\ 6+4+12, \\ 5+0+6 \end{array} \right\} \\
&= 10
\end{aligned}
$$

This minimum is achieved when $k = 1$



## Example II

- Thus an optimal parenthesization is $(A_1((A_2A_3)A_4))$
- The cost of this is 10

## In-Class Exercise

- Consider the sequence of three matrices, $A_1, A_2, A_3$ whose dimensions are given by the sequence $1, 2, 1, 2$ (i.e. $p_0 = 1$, $p_1 = 2$, $p_2 = 1$, $p_3 = 2$)
- Q1: What are the m array and s array for these inputs?
- Q2: What is the optimal parenthesization?