Matrix-chain Products

- Given: $p_i \times p_i$ matrices $A_i$, $1 \leq i \leq n$
- Problem: Compute $A_1 \cdot A_2 \cdot \ldots \cdot A_n$

**Example:**

$A \cdot (B \cdot C)$, where:

- $A$ is $1 \times 10$
- $B$ is $10 \times 2$
- $C$ is $2 \times 10$

In general:

$p \times q \text{ times } q \times r$ costs $p \cdot q \cdot r$

**Greedy Algorithm?**

In above example, it was best to start with the cheapest adjacent pair.

Always true?

No.

A Greedy Algorithm?

Simple Algorithm

- Just try all possible parenthesizations
- How many are there?
  - $P(1) = 1$
  - $P(n) = \sum_{k=1}^{n-1} P(k)P(n-k), n > 1$
  - $P(n) = \frac{1}{n} \left( \frac{2n-2}{n-1} \right) = \Omega \left( \frac{n^{3/2}}{n} \right)$

Repeated Subproblems

- All 5 Parenthesizations of $A_1 \cdot A_2 \cdot A_3 \cdot A_4$:

Optimal Substructure:

- **Theorem:** if the last multiply is $(A_1 \ldots A_i)(A_{i+1} \ldots A_n)$, then $A_1 \ldots A_i$ is optimally parenthesized, as is $A_{i+1} \ldots A_n$.
- **Proof:** Could improve if not.

  - Useful? Two problems:
    - Don’t know $i$.
    - $(A_1 \ldots A)$ is a prefix of input, but not $(A_{i+1} \ldots A_n)$
Optimal Substructure: Strengthened Induction Hyp.

- **Theorem:** if the last mult in opt calculation of $A_1...A_j$ is $(A_1...A_k) \cdot (A_{k+1}...A_j)$, then $A_1...A_j$ is optimally parenthesized, as is $A_{k+1}...A_j$.

  **Proof:** Could improve if not.

- Let $M[i,j] = \text{min ops to multiply } A_i...A_j$

  $$M[i,j] = \begin{cases} 0 & i = j \\ \min \{ (M[i,k] + M[k+1,j]) + p_i \cdot (p_k \cdot p_j) \} & i < j \end{cases}$$

**Example:**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $p_0 = 2$, $A_1: 2 \times 3$
- $p_1 = 3$, $A_2: 3 \times 1$
- $p_2 = 1$, $A_3: 1 \times 5$
- $p_3 = 5$, $A_4: 5 \times 1$
- $p_4 = 1$

An O(n^3) Algorithm

// Goal: $M[i,j] = \text{min ops to multiply } A_i...A_j$

for $j := 1$ to $n$ do

  $M[i,j] := 0$;

  for $i := (j - 1)$ downto 1 do

  $M[i,j] := \min \{ (p_i \cdot p_k \cdot p_j) + M[i,k] + M[k+1,j]) \}$

Notes

- Diagonal $M[i+2]$, e.g., gives best cost for multiplying adjacent triples $A_iA_{i+1}A_{i+2}$
- Exercise: rewrite alg to compute successive diagonals instead of successive columns
- Question: can it be rewritten to compute successive rows?
- $n^3 \rightarrow n \log n$ time is possible (but not easy)
- General structure of algorithm is useful for other problems on trees
  - E.g., go look up “CKY” alg for context-free parsing