CMSC 451 Oct 5

Chain Matrix Multiplication

Problem: Given a sequence of matrices

$$A_1, A_2, \cdots, A_n.$$

find the fastest way to compute the product

$$C = A_1 A_2 \cdots A_n$$

Matrix Multiplication

A: $p \times q$ matrix, B: $q \times r$ matrix

$C = A \cdot B$: $p \times r$ matrix, s.t.

$$C[i,j] = \sum_{k=1}^{q} A[i,k] \cdot B[k,j]$$

Algorithm:

```
for i ← 1 to p
    for j ← 1 to r
        C[i,j] ← 0
    for k ← 1 to q
        C[i,j] ← C[i,j] + A[i,k] · B[k,j]
```

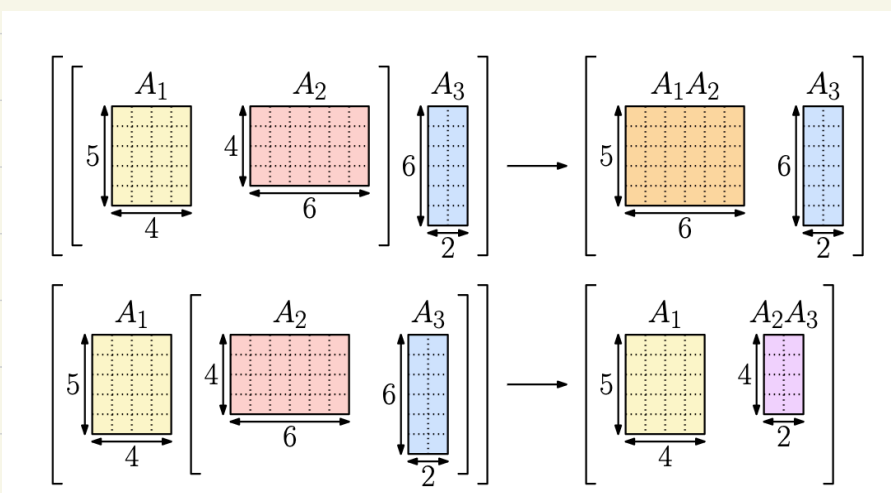
Time complexity $O(pqr)$

Chain Multiplication Case:

$A_1 : 5\times4$, $A_2 : 4\times6$, $A_3 : 6\times2$

$(A_1 A_2) A_3 \longrightarrow$ cost $= 5\times4\times6 + 5\times6\times2 = 180$

$A_1 (A_2 A_3) \longrightarrow$ cost $= 4\times6\times2 + 5\times4\times2 = 88$

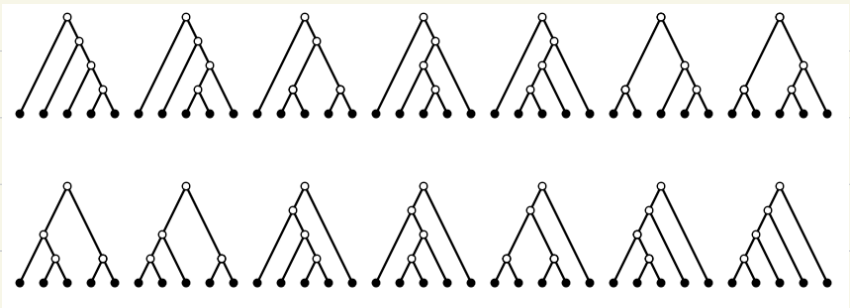


Problem: Find the best order of multiplications w/ smallest number of operations.

A_i has dimensions $p_{i-1} \times p_i$

An order can be represented by a binary tree.

$n=5$:



Let $P(n)$ be # of binary trees with n leaves.

$$P(n) = \begin{cases} 1 & \text{if } n=1 \\ \sum_{k=1}^{n-1} P(k)\,P(n-k) & n \geq 2 \end{cases}$$

$P(n) = C(n-1)$ Catalan Number

$$C(n) = \frac{1}{n+1}\binom{2n}{n} = \Theta\left(\frac{4^n}{n^{3/2}}\right)$$

Brute-force is exponential time!

DP Formulation.

Subproblem. $M(i,j)$ = the minimum cost of multiplying $A_i, A_{i+1}, \ldots, A_j$ into $A(i,j)$

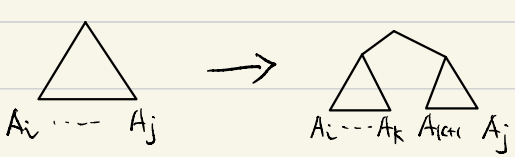
Goal: $M(1,n)$

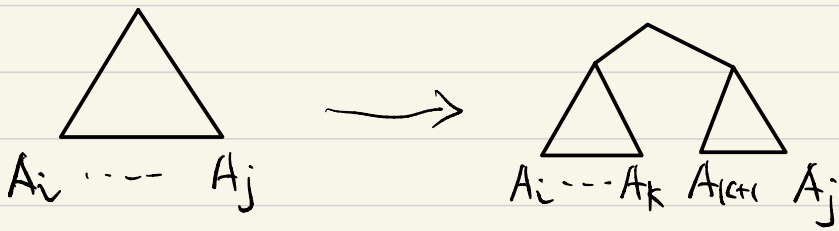
Base cases: $M(i,i) = 0$

Decomposing into subproblems.

Suppose now want to do $M(i,j)$.

Last step must be multiplying $A(i,k)$ with $A(k+1,j)$ for $i \leq k < j$.





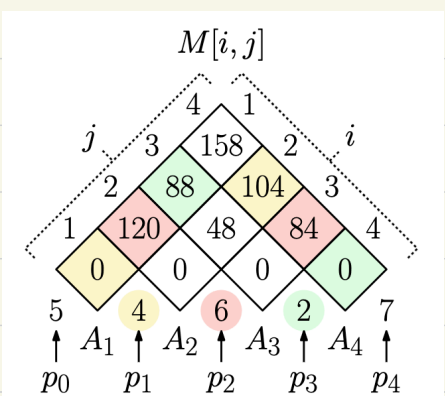
Cost: $M(i,k) + M(k,j) + p_{i-1} \cdot p_k \cdot p_j$

Optimal strategy: choose the k w/ smallest cost

$$M(i,j) = \begin{cases} 0 & i = j \\ \min_{i \le k \le j-1} \left\{ \begin{matrix} M(i,k) + M(k,j) \\ + p_{i-1} \cdot p_k \cdot p_j \end{matrix} \right\} & i < j \end{cases}$$

Memoized Implementation.

```
memo-cmm(i,j)
    if (M[i,j] = -1)
        ans ← +∞
        for k ← i to j-1
            cost ← memo-cmm(i,k) + memo-cmm(k+1,j)
                    + p[i-1] * p[k] * p[j]
            if cost < ans
                ans ← cost
        M[i,j] ← ans
    return M[i,j]
```



Bottom-up Implementation.

Not trivial to find order !

```
for i ← 1 to n
    for j ← i to n
```

Solution: compute by interval length.

```
bottom-up-cmm()
    for i ← 1 to n      M[i,i] ← 0
    for l ← 2 to n
        for i ← 1 to n-l+1
            j ← i+l-1
            ans ← ∞
            for k ← i to j-1
                M[i,j] ← min(M[i,j],
                             M[i,k] + M[k+1,j] + P_{i-1} P_k P_j)
    return M[1,n]
```

Running time $O(n^3)$