CSF 364
Design & Analysis of Algorithms

ALGORITHM DESIGN TECHNIQUES

Matrix-Chain Multiplication: Problem Definition

2/21/2015 Sundar B. CSIS, BITS, Pilani

Example – Matrix-Chain Multiplication

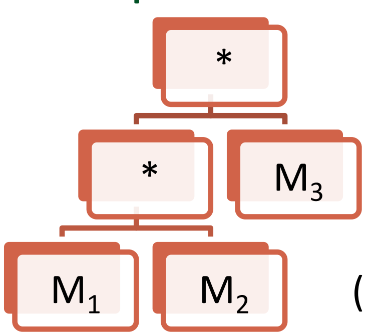
- Consider the following expression:
  - $M_1 * M_2 * M_3$
    - where M_j is a matrix of dimensions $p_{j-1} * p_j$ for $j = 1$ to 3
- Matrix Multiplication is associative
  - i.e. $(M_1 * M_2) * M_3 = M_1 * (M_2 * M_3)$
    - Exercise: Prove this.
- Then the above expression can be evaluated
  - either as $(M_1 * M_2) * M_3$
    - using $(p_0 * p_1 * p_2) + (p_0 * p_2 * p_3)$ scalar multiplications
  - or as $M_1 * (M_2 * M_3)$
    - using $(p_1 * p_2 * p_3) + (p_0 * p_1 * p_3)$ scalar multiplications

EXAMPLE – MATRIX-CHAIN MULTIPLICATION

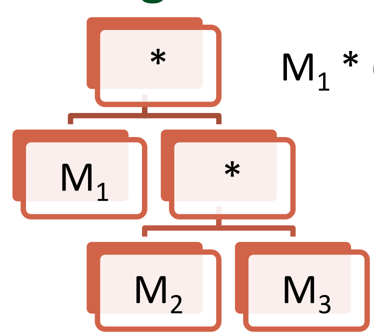
- Consider the following generalized expression:
  - $M_1 * M_2 * \ldots * M_n$
    - where M_j is a matrix of dimensions $p_{j-1} * p_j$ for $j = 1$ to n
- Problem:
  - Given the above expression, how do we minimize the number of scalar multiplications?
    - This depends on the way the expression is parenthesized (which determines the order of evaluation)
- Definition:
  - Given a chain $(M_1 * M_2 * \ldots * M_n)$ of n matrices, where for $j = 1$ to n, M_j is a matrix of dimensions $p_{j-1} * p_j$
  - find the optimal parenthesization
    - i.e. the parenthesization resulting in minimal number of scalar multiplications

Example – McM - Brute Force Solution

- Algorithm BF_MCM:
  - Find all possible parenthesizations
  - For each possible parenthesization, count the scalar multiplications required.
  - Find the minimum among these counts.
- Time Complexity:
  - O(Par(n)) where Par(n) is the number of possible parethesizations.
  - Each parenthesization is a parse tree: e.g.



$(M_1 * M_2) * M_3$



$M_1 * (M_2 * M_3)$

EXAMPLE – MCM - BRUTE FORCE SOLUTION

- What is Par(n)?
  - A chain of n matrices can be split between the k^{th} and $(k+1)^{st}$ matrices for any k =1,2, …n-1;
  - Then the sub-chains can be parenthesized independently
- Thus Par(n) =
  - 1 if n=1
  - $\sum_{k=1 \text{ to } n-1}$ Par(k) * Par(n-k) if n>=2
- Par(n) grows at the same rate as B(n)
  - where B(n) is the number of different binary trees with n nodes and
  - $B(n) = \Omega(4^n / n^{3/2})$ **// see Problem 12-4 in Cormen et. al.**
- Conclusion: Time taken by BF_MCM is exponential in n.