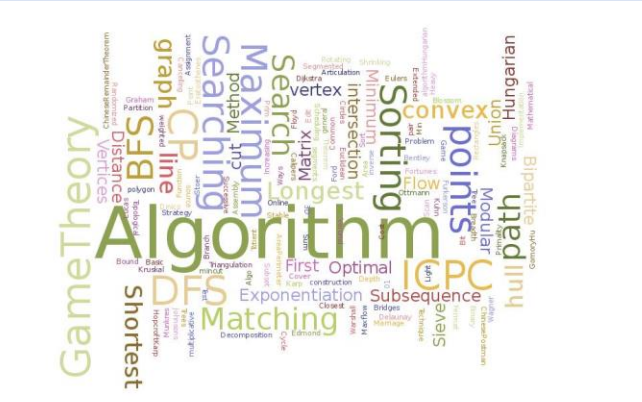
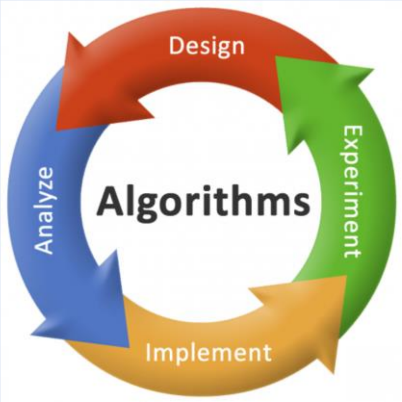
Dynamic Programming
Matrix Chain Multiplication

COP 3503
Fall 2021
Department of Computer Science
University of Central Florida
Dr. Steinberg





The Problem

- Our input is a sequence of n matrices $\langle A_1, A_2, \ldots, A_n \rangle$
- The desire output is a multiplication setup $A_1 * A_2 * A_3 * \cdots * A_n$ $\left((A_1 * A_2) * (A_3 * A_4) * A_5\right)$
- A product of matrices is fully parenthesized if it is either: one matrix, product of two fully parenthesized matrix products, or surrounded by parentheses
- 5 ways we can parenthesize the above product:
    1. $\left(\left((\mathrm{A}_1\mathrm{A}_2)\mathrm{A}_3\right)\mathrm{A}_4\right)$
    2. $\left((\mathrm{A}_1\mathrm{A}_2)(\mathrm{A}_3\mathrm{A}_4)\right)$
    3. $\left(\left(\mathrm{A}_1(\mathrm{A}_2\mathrm{A}_3)\right)\mathrm{A}_4\right)$
    4. $\left(\mathrm{A}_1\left((\mathrm{A}_2\mathrm{A}_3)\mathrm{A}_4\right)\right)$
    5. $\left(\mathrm{A}_1\left(\mathrm{A}_2(\mathrm{A}_3\mathrm{A}_4)\right)\right)$

Multiplying Matrices

```
Matrix-Multiplication(A,B)
if A.cols not equal B.rows
        return error
for i = 1 to A.rows  //p
        for j = 1 to B.cols //r
                c[i,j] = 0
                for k = 1 to A.cols  //q
                        c[i,j] = c[i,j] + A[i,k] * B[k,j]
return C
```

Number of Scalar Multiplications is p * r * q

So why is this even relative?

- Every multiplication can be costly.
- Type equation here.
- $\langle A_1, A_2, \dots, A_n \rangle$
  - 10 x 100, 100 x 5, 5 x 50
  - $\left((A_1 A_2) A_3\right) \rightarrow$ 5000 scalar multiplications
  - $\left(A_1 (A_2 A_3)\right) \rightarrow$ 75000 scalar multiplications
- The objective of this is to MINIMIZE the number of multiplications.

The Matrix-Chain Multiplication

- We are given a chain of n matrices $\langle A_1, A_2, \dots, A_n \rangle$ where $i = 1,2, \dots, n$ matrix A_i has dimension $p_{i-1} x p_i$
- The output is a fully parenthesize product of $A_1, A_2, \dots, A_n$ in a way that MINIMIZES the number of scalar multiplcations.

Dynamic Programming

- Step 1: Optimal parenthesization of $A_i \dots A_j$ which splits the product between A_k and A_{k+1} must contain within it optimal parenthesization of $A_i \dots A_k$ and $A_{k+1} \dots A_j$
- To optimally parenthesize $A_i \dots A_j$
  - Examine ALL candidates k, k = i , i + 1,…, j – 1 for splitting
  - Take optimal parenthesization $A_i \dots A_k$ and $A_{k+1} \dots A_j$

Dynamic Programming cont.

- Step 2 is to recursively define an optimal solution
- $m[i, j]$ = minimum of scalar multiplications needed to compute $A_i A_{i+1} \dots A_j$
- $m[i, j] = m[i, k] + m[k + 1, j] + P_{i-1} P_k P_j$
- Table S will store the optimal splitting

Dynamic Programming cont.

- Step 3 Computing the Optimal Answer

```
MATRIX-CHAIN-ORDER(P)
n = P.length – 1
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
                m[i,j] = ∞
                for k = i to j – 1
                        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
return m,s
```

Dynamic Programming cont.

- Step 4 Display the Solution

```
PRINT-OPTIMAL-PARENTHSIS(s,i,j)
if i == j
        print Ai
else
        print "("
        PRINT-OPTIMAL-PARENTHSIS(s,i,s[i,j])
        PRINT-OPTIMAL-PARENTHSIS(s,s[i,j] + 1,j)
        print ")"
```

Example