CASE STUDY 1 - Matrix Chain Multiplication

See also Cormen chpt 16. (chpt 15 in the 2nd edition).

Let A and B be two matrices.

- The product AB is defined only if the number of columns of A equals the number of rows of B.

- If A has dimensions $p \times q$ and B has dimensions $q \times r$ then AB has dimensions $p \times r$, and it takes pqr scalar multiplications to compute AB.

- Given three matrices A, B, C (with compatible dimensions)
$$((AB)C) = (A(BC))$$

- Even though $(AB)C$ and $A(BC)$, they can take quite different times to compute.

- e.g. if A is 100×10, B is 10×50, C is 50×5 then computing $(AB)C$ takes $100 \cdot 10 \cdot 50 + 100 \cdot 50 \cdot 5 = 75000$ operations while computing $A(BC)$ takes $10 \cdot 50 \cdot 5 + 100 \cdot 10 \cdot 5 = 7500$ operations.

Matrix-chain muliplication problem

We are given matrices $A_1 A_2 \cdots A_n$.
Matrix A_i has dimensions $p_{i-1} \times p_i$.

Problem: What is the minimum number of scalar multiplications needed to evaluate
$A_1 A_2 A_3 \cdots A_n$?

Subproblem:
For each i, j such that $1 \leq i \leq j \leq n$:
What is the minimum number of scalar multiplications needed to evaluate
$A_i A_{i+1} A_{i+2} \cdots A_j$?

Note: the matrix $A_i A_{i+1} \cdots A_j$ has p_{i-1} rows and p_j columns.

Looking for the recursion

Let $m[i, j]$ denote the number of scalar multiplications needed to evaluate $A_i A_{i+1} \cdots A_j$.

If $i = j$ then $m[i, j] = 0$ since we just have to get the matrix A_i.

If $k \geq i$ and $k < j$ then the minimum number of scalar multiplications needed to evaluate

$$(A_i A_{i+1} \ldots A_k)(A_{k+1} \cdots A_j)$$

is

(the min. number needed to compute $A_i A_{i+1} \ldots A_j$)

+(the min. number needed to compute $A_i A_{i+1} \ldots A_j$)

+(the operations needed to multiply the two matrices)

That is,

$$m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j$$

In order to find the minimum, we choose the k that minimizes this expression.

Recursion and algorithm

- If $i = j$ then $m[i, j] = 0$.
- If $i < j$ then

$$m[i, j] = \min_{i \le k < j} \{m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j\}$$

We can evaluate this using memoization.

1. $m[i, j] \leftarrow -1$ for all $1 \le i \le j \le n$.
2. Output $MCR(1, n)$

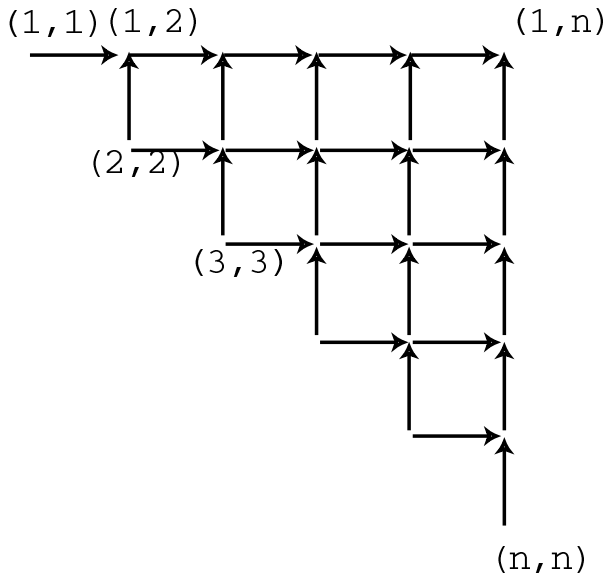
$MCR(i, j)$.
3. **if** $m[i, j] \ge 0$ **then**
4. **return** $m[i, j]$
5. **else**
6. **if** $i = j$ **then**
7. $m[i, j] = 0$.
8. **else**
9. $m[i, j] \leftarrow \min_{i \le k < j} \{MCR[i, k] + MCR[k + 1, j] + p_{i-1} p_k p_j\}$
10. **return** $m[i, j]$.

This takes $O(n^3)$ time.

Recursion-free version

The subproblem for (i, j) depends on subproblem (i, k) and also on subproblem $(k + 1, j)$ for all $i \leq k \leq j$.

Dependency graph looks like:



So a loop like:

for $i \leftarrow 1$ to n
  for $j \leftarrow 1$ to n
    Compute $m[i, j]$

won't work. Instead we need something like:

for $l \leftarrow 1$ to n
  for all i, j such that $j = i + l - 1$
    Compute $m[i, j]$

Recursion free dynamic programming solution

$MatrixChainOrder(p_1, p_2, \ldots, p_n)$.
1. **for** $l \leftarrow 1$ to n **do**
2. **for** $i \leftarrow 1$ to $n - l + 1$ **do**
3. $j \leftarrow i + l - 1$ [so $[i, j]$ contains l elements]
4. **if** $i = j$ **then**
5. $m[i, j] \leftarrow 0$
6. **else**
7. $m[i, j] \leftarrow \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j\}$

Multiplying the matrices

We have computed the minimum number of multiplications required. How do we go about multiplying the matrices.

We assume that there is a library function $MatrixMultiply(A, B)$ that multiplies A and B.

The following returns the product of $A_i A_{i+1} \ldots A_j$.

$Multiply(i, j)$
1. **if** $i = j$ **then**
2. **return** A_i.
3. **else**
4. find k such that $m[i, j] = m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j$.
5. $A_L \leftarrow Multiply(i, k)$
6. $A_R \leftarrow Multiply(k + 1, j)$
7. **return** $MatrixMultiply(A_L, A_R)$.