Matrix Chain Multiplication

Ibrahim Albluwi

Matrix Multiplication Review. Given two matrices A and B of sizes $d_0 \times d_1$ and $d_2 \times d_3$ respectively:

- d_1 and d_2 must be equal for the multiplication to be valid.
- The result of $A \bullet B$ is a matrix of size $d_0 \times d_3$.

Example.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} \bullet \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

$$4 \times 2 \qquad\qquad 2 \times 4 \qquad\qquad 4 \times 4$$

$$\begin{aligned} c_{11} &= (a_{11} \times b_{11}) + (a_{12} \times b_{21}) \\ c_{12} &= (a_{11} \times b_{12}) + (a_{12} \times b_{22}) \\ c_{11} &= (a_{11} \times b_{13}) + (a_{11} \times b_{23}) \\ c_{11} &= (a_{11} \times b_{14}) + (a_{11} \times b_{24}) \end{aligned}$$

> **Cost of Matrix Multiplication.** Given two matrices A and B of sizes $d_0 \times d_1$ and $d_2 \times d_3$ respectively, the number of operations performed is $d_0 \times (d_1 = d_2) \times d_3$.

Example. Multiplying the above two matrices that are of sizes 4×2 and 2×4 requires filling 4×4 (i.e. $d_0 \times d_3$) cells in the result matrix, each requiring 2 (i.e. d_1 or d_2) multiplications, which makes the total $4 \times 4 \times 2 = 32$ operations.

Multiplying Multiple Matrices. Consider the following three matrices that we would like to multiply.

$$\underset{[8 \times 2]}{A} \bullet \underset{[2 \times 8]}{B} \bullet \underset{[8 \times 2]}{C}$$

These matrices can be multiplied in two different ways: $(A \bullet B) \bullet C$ or $A \bullet (B \bullet C)$

Method 1. $(A \bullet B) \bullet C$

$(A \bullet B)$ requires $8 \times 2 \times 8 = 128$ operations and produces a matrix K of size $[8 \times 8]$.

$(\ K\ ) \bullet C$ requires $8 \times 8 \times 2 = 128$ operations and produces a matrix of size $[8 \times 2]$

Total $= 128 + 128 = 256$ operations (counting only multiplications between numbers)

Method 2. $A \bullet (B \bullet C)$

$(B \bullet C)$ requires $2 \times 8 \times 2 = 32$ operations and produces a matrix K of size $[2 \times 2]$.

$A \bullet (\ K\ )$ requires $8 \times 2 \times 2 = 32$ operations and produces a matrix of size $[8 \times 2]$

Total $= 32 + 32 = 64$ operations (counting only multiplications between numbers)

It is clear that the order of multiplication affects the number of performed operations.

Another Example. Consider the following three matrices that we would like to multiply.

$$\underset{[1 \times n]}{A} \bullet \underset{[n \times 1]}{B} \bullet \underset{[1 \times n]}{C}$$

Do the Math. Show that $(A \bullet B) \bullet C$ requires $2n$ operations while $A \bullet (B \bullet C)$ requires $2n^2$ operations.

Problem Statement.

> **Matrix Chain Multiplication.** Given a chain of matrices to be multiplied:
>
> $$\underset{[d_0 \times d_1]}{A_1} \bullet \underset{[d_1 \times d_2]}{A_2} \bullet \underset{[d_2 \times d_3]}{A_3} \bullet \cdots \bullet \underset{[d_{n-1} \times d_n]}{A_n}$$
>
> Find the parenthesization that requires the minimum number of operations.

Brute Force Solution. Compute the number of operations for all possible parenthesizations and pick the minimum. The number of possible parenthesizations is exponential (it is called the *Catalan Number*, which is $\sim \dfrac{4^n}{\pi n \sqrt{n}}$).

Definition.
Given the matrix chain $A_1 \bullet A_2 \bullet A_3 \bullet \ldots \bullet A_n$, let $\text{opt}(i, j)$ be the minimum number of operations that can be performed when multiplying the matrices $i \longrightarrow j$ (inclusive). Hence:

- $\text{opt}(1, n)$ is the main problem we would like to solve.
- $\text{opt}(1, 1)$, $\text{opt}(2, 2)$, $\text{opt}(3, 3)$, etc. all have a solution of **0**.
  (These represent subproblems involving only *one* matrix in the chain)

All Possible Parenthesizations. The possible parenthesizations for the matrix chain $A_1 \bullet A_2 \bullet A_3 \bullet A_4$:

$A_1 \bullet (A_2 \bullet (A_3 \bullet A_4))$

$A_1 \bullet ((A_2 \bullet A_3) \bullet A_4)$

$(A_1 \bullet A_2) \bullet (A_3 \bullet A_4)$

$((A_1 \bullet A_2) \bullet A_3) \bullet A_4$

$(A_1 \bullet (A_2 \bullet A_3)) \bullet A_4$

Observation. The optimal solution is the minimum between the cost of three possible decisions:

1. Multiply A_1 with the result of $(A_1 \bullet A_2 \bullet A_3)$. This covers the first two parenthesizations.
2. Multiply the result of $(A_1 \bullet A_2)$ with the result of $(A_3 \bullet A_4)$.
3. Multiply the result of $(A_1 \bullet A_2 \bullet A_3)$ with A_4. This covers the last two parenthesizations.

These three decisions represent the three possible final multiplications. Each decision involves finding the solution for two subproblems, each involving part of the matrix chain.

In other words, we could write:

$$\begin{aligned}\text{opt}(1,4) = \min(\ & \text{opt}(1,1)+\text{opt}(2,4) + \text{cost of multiplying } A_1 \text{ with } (A_1 \bullet A_2 \bullet A_3),\\ & \text{opt}(1,2)+\text{opt}(3,4) + \text{cost of multiplying } (A_1 \bullet A_2) \text{ with } (A_3 \bullet A_4),\\ & \text{opt}(1,3)+\text{opt}(4,4) + \text{cost of multiplying } (A_1 \bullet A_2 \bullet A_3) \text{ with } A_4 \quad).\end{aligned}$$

In General. Given a chain of n matrices to multiply, there are $n-1$ possible split points (i.e. $n-1$ possible decisions to take on what the final multiplication should be).

$$\begin{array}{l} A_1 \bullet (A_2 \bullet A_3 \bullet A_4 \bullet \ldots \bullet A_{n-1} \bullet A_n)\\ (A_1 \bullet A_2) \bullet (A_3 \bullet A_4 \bullet \ldots \bullet A_{n-1} \bullet A_n)\\ (A_1 \bullet A_2 \bullet A_3) \bullet (A_4 \bullet \ldots \bullet A_{n-1} \bullet A_n)\\ \ldots\\ (A_1 \bullet A_2 \bullet A_3 \bullet A_4 \bullet \ldots \bullet A_{n-1}) \bullet A_n \end{array}$$

For each split point, there are two subproblems to solve and a final multiplication to be performed between the resulting matrix on the left of the split point and the resulting matrix on the right of the split point.

Observation.

Consider the following parenthesization:

$$\underset{[d_{i-1}\times d_i]}{(\ A_i} \bullet \underset{[d_i\times d_{i+1}]}{A_{i+1}} \bullet \underset{[d_{i+1}\times d_{i+2}]}{A_{i+2}} \bullet \cdots \bullet \underset{[d_{k-1}\times d_k]}{A_k\ )} \bullet \underset{[d_k\times d_{k+1}]}{(A_{k+1}} \bullet \underset{[d_{k+1}\times d_{k+2}]}{A_{k+2}} \bullet \cdots \bullet \underset{[d_{k-1}\times d_j]}{A_j\ )}$$

- Regardless of how matrices i to k are multiplied, the resulting matrix must be of size $[d_{i-1} \times d_k]$
- Regardless of how matrices k+1 to j are multiplied, the resulting matrix must be of size $[d_k \times d_j]$

> Multiplying the result of $(A_i \longrightarrow A_k)$ with the result of $(A_{k+1} \longrightarrow A_j)$ requires $d_{i-1} \times d_k \times d_j$ operations.

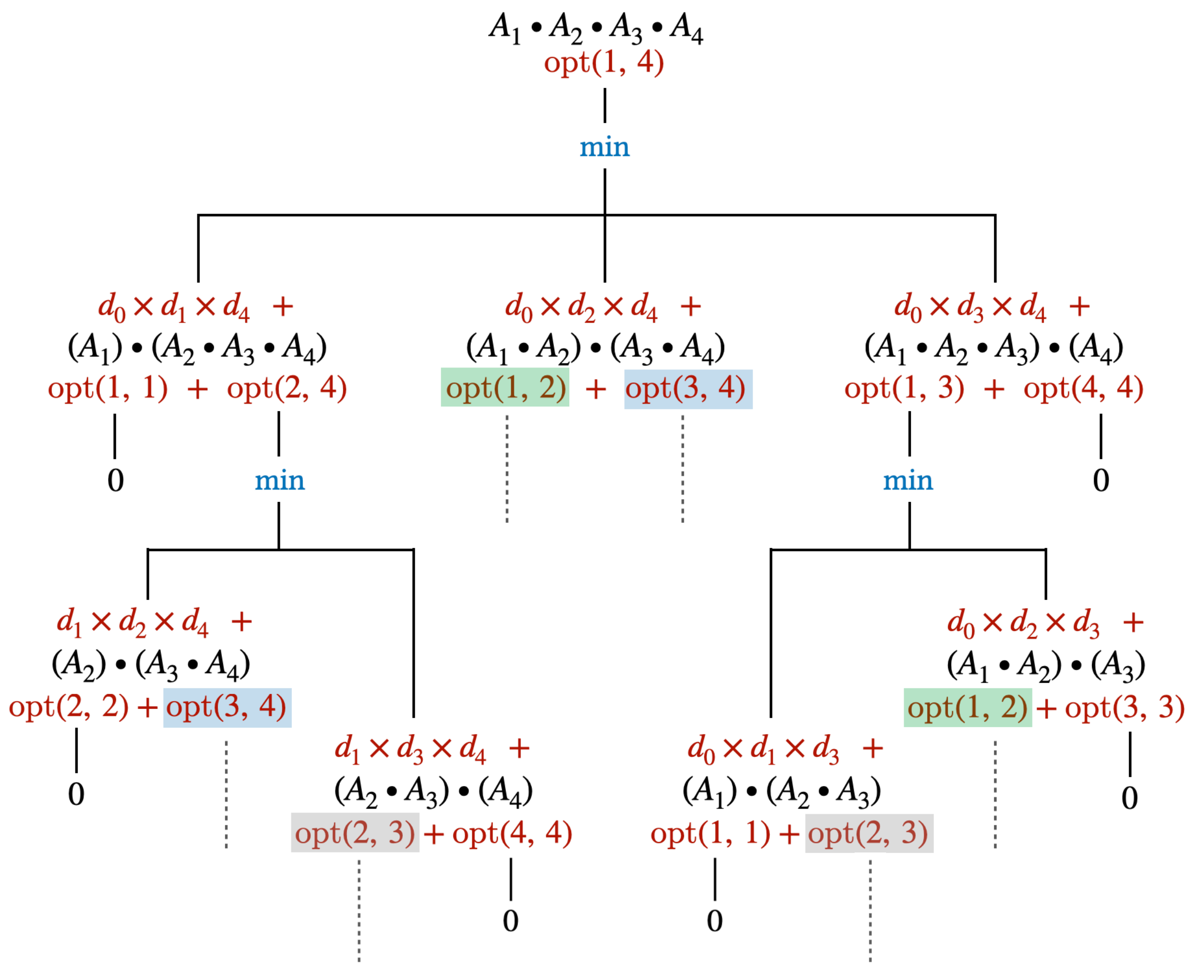
Optimal Substructure. Assuming $0 \le i \le n$ and $i \le j \le n$.

$$\text{opt}(i,j) = \begin{cases} 0 & \text{if } i=j \text{ or } i=0 \\ \min\limits_{i \le k < j} \{\text{opt}(i,k) + \text{opt}(k+1,j) + d_{i-1}\times d_k \times d_j\} & \text{otherwise} \end{cases}$$

Using this optimal substructure on $A_1 \bullet A_2 \bullet A_3 \bullet A_4$:

$$\begin{aligned}\text{opt}(1,4) = \min(\ & \text{opt}(1,1)+\text{opt}(2,4) + d_0\times d_1\times d_4,\\ & \text{opt}(1,2)+\text{opt}(3,4) + d_0\times d_2\times d_4,\\ & \text{opt}(1,3)+\text{opt}(4,4) + d_0\times d_3\times d_4 \quad).\end{aligned}$$

A Partial Trace. The highlighted subproblems are overlapping.

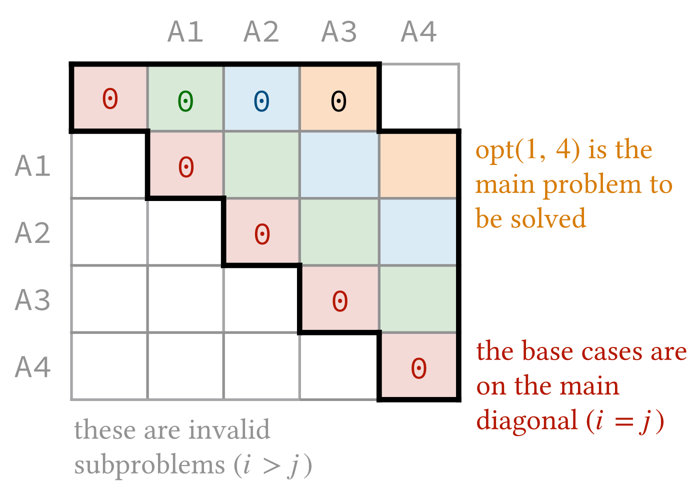


Bottom-up Solution.

We need to create a 2D array for storing the results of subproblems to avoid computing the more than once.

Note that:

- The *diagonal* starting at column 0 represents subproblems of size 0 matrices,
- The *diagonal* starting at column 1 represents subproblems of size 1 matrix,
- The *diagonal* starting at column 2 represents subproblems of size 2 matrices,
- etc.



Therefore, we will fill the diagonals one by one (smallest subproblems followed by larger subproblems)

We assume that the input to the problem is the dimensions $d_0, d_1, \ldots, d_n$, which are stored in `d[0]`, `d[1]`, …, `d[n]`, where `d[]` is a 1D array of size `n+1`.

```
MCM(d[], n):

Create array RESULT[n+1][n+1]
Create array SPLIT[n+1][n+1]  ──→ stores at SPLIT[i][j] the optimal split point for opt(i, j)

FOR every diagonal diag = 0  to  n-1:
    FOR every row i = 1  to  n - diag:
        j = i + diag
        SOLVE(i, j, d, SPLIT, RESULT)  ──→ solve opt(i, j)

RETURN RESULT[1][n]
```

```
SOLVE(i, j, d[], SPLIT[][], RESULT[][]):

IF i >= j:
    RESULT[i][j] = 0
    RETURN

RESULT[i][j] = ∞
FOR  k = i  to  j-1:
    cost = RESULT[i][k] + OPT[k+1][j] + (d[i-1] * d[k] * d[j])
    IF cost < RESULT[i][j]:
       RESULT[i][j] = cost
       SPLIT[i][j] = k
```

Example Trace.
Assume that `d[] = {10, 1, 2, 3, 4}`:

$$A_1 \bullet A_2 \bullet A_3 \bullet A_4$$
$[10 \times 1]$ $[1 \times 2]$ $[2 \times 3]$ $[3 \times 4]$

		A1	A2	A3	A4
	0	0	0	0	
A1		0	20	36	58
A2			0	6	18
A3				0	24
A4					0

OPT[][]

		A1	A2	A3	A4
A1			1	1	1
A2				2	3
A3					3
A4					

SPLIT[][]

```
RESULT[1][2] = d[0] x d[1] x d[2] + opt[1][1] + opt[2][2] = 20 + 0 + 0 = 20
 SPLIT[1][2] = 1


RESULT[2][3] = d[1] x d[2] x d[3] + opt[2][2] + opt[3][3] = 6 + 0 + 0 = 6
 SPLIT[1][2] = 2


RESULT[3][4] = d[2] x d[3] x d[4] + opt[3][3] + opt[4][4] = 24 + 0 + 0 = 24
 SPLIT[1][2] = 2


RESULT[1][3] = min( d[0] x d[1] x d[3] + opt[1][1] + opt[2][3] = 30 +  0 + 6 = 36,
                    d[0] x d[2] x d[3] + opt[1][2] + opt[3][3] = 60 + 20 + 0 = 80)
             = 36
 SPLIT[1][3] = 1


RESULT[2][4] = min( d[1] x d[2] x d[4] + opt[2][2] + opt[3][4] =  8 + 0 + 24 = 36,
                    d[1] x d[3] x d[4] + opt[2][3] + opt[4][4] = 12 + 6 + 0  = 18)
             = 18
 SPLIT[2][4] = 3


RESULT[1][4] = min(d[0] x d[1] x d[4] + opt[1][1] + opt[2][4] = 40 + 0 + 18  = 58,
                   d[0] x d[2] x d[4] + opt[1][2] + opt[3][4] = 80 + 20 + 24 = 124,
                   d[0] x d[3] x d[4] + opt[1][3] + opt[4][4] = 120 + 36 + 0 = 156)
             = 58
 SPLIT[1][4] = 1
```

Running Time Analysis

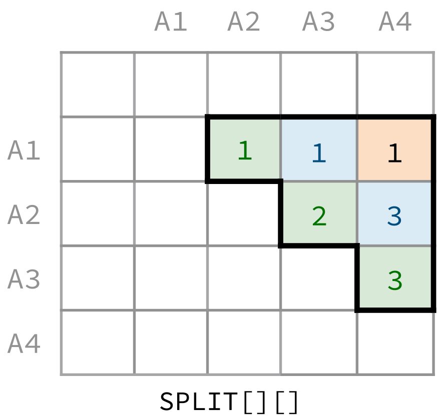
Counting how many times `cost` is computed in function `SOLVE`:

There are n diagonals:

Diagonal $0:$	n cells	$\times$	0 computations
Diagonal $1:$	$n-1$ cells	$\times$	1 computation
Diagonal $2:$	$n-2$ cells	$\times$	2 computations
Diagonal $3:$	$n-3$ cells	$\times$	3 computations
Diagonal $n-1:$	$n-(n-1)$ cells	$\times$	$n-1$ computations

$$\text{Total} = \sum_{i=0}^{n-1} (n-i) \times i \;=\; \sum_{i=0}^{n-1} ni - i^2 \;=\; n\sum_{i=0}^{n-1} i \;-\; \sum_{i=1}^{n-1} i^2 \;=\; \tfrac{n}{2} \times n(n-1) \;-\; \tfrac{n}{6}(n+1)(2n+1) \;=\; \Theta(n^3)$$

Printing the Optimal Parenthesization.




```
PRINT(SPLIT[][], i, j):
IF (i == j):
  DISPLAY "A" + i
  RETURN

DISPLAY "("

PRINT(SPLIT, i, SPLIT[i][j])
PRINT(SPLIT, SPLIT[i][j]+1, j)

DISPLAY ")"
```