Multiplicación en cadena de matrices, parte 1

Multiplicación en cadena de matrices

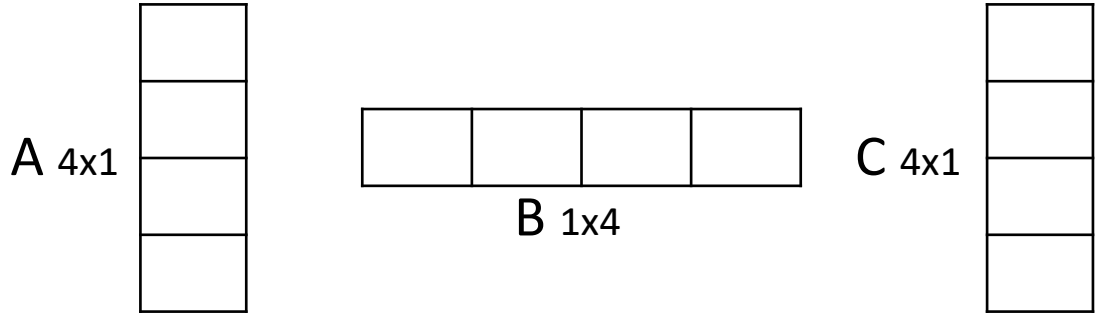
Entrada: Dado un conjunto de matrices $A_1, A_2, \ldots, A_N$ las cuales son compatibles para realizar el producto (el número de columnas de A_i es igual al número de filas de A_{i+1} para $i=1:N-1$), la entrada será el arreglo de enteros positivos no nulos $P = \{p_0, p_1, p_N\}$, tal que la matriz A_i tiene una dimensión p_{i-1} x p_i

Dicho producto no es conmutativo, es decir, no se puede cambiar el orden de los elementos, lo único que se puede hacer es escoger donde situar los paréntesis.

Salida: La 'parentización' del producto en cadena $A_1*A_2*\ldots*A_N$ que minimiza la cantidad total de operaciones (multiplicaciones escalares)

Multiplicación en cadena de matrices

Ejemplo 1: Si A es 10x100, B es 100x5, C es 5 x50



Alternativa 1: ((A*B)*C) = (4x1x4) + (4x4x1) = 32 productos

Alternativa 2: (A*(B*C)) = (1x4x1) + (4x1x1) = 8 productos

Ejemplo 2: Si A es 10x100, B es 100x5, C es 5 x50

Alternativa 1: ((A*B)*C) = (10x100x5) + (10x5x50) = 7.500

Alternativa 2: (A*(B*C)) = (100x5x50) + (10x100x50) = 75.000

Solución mediante programación dinámica

En general, con *n* matrices, cuántas alternativas diferentes de 'parentización' existen? $(\boldsymbol{N-1})!$

Para comenzar a vislumbrar una solución por programación dinámica pensemos: ¿cómo definimos que elementos constituyen un subproblema?

La última multiplicación se llevaría a cabo sobre dos subproductos: $(A_1, A_2, \ldots, A_k)$ y $(A_{k+1}, A_{k+2}, \ldots, A_N)$ con $k = N-1$ posibles particiones

Si repetimos el proceso, digamos en el subproducto izquierdo, la multiplicación se llevaría a cabo sobre dos subproductos: $(A_1, A_2, \ldots, A_{k'})$ y $(A_{k'+1}, A_{k'+2}, \ldots, A_k)$ con $k' = k-1$ posibles particiones

Siendo así, podemos decir de manera general que en algún momento tenemos que resolver el subproducto de los elementos *i* a *j* con $i \leq j$

Solución mediante programación dinámica

Notación: sea $M_{i,j}$ con $1 \leq i \leq j \leq N$ la cantidad óptima de operaciones (multiplicaciones) para los matrices *i* a *j*

Si *i=j*, el problema es trivial pues no se necesitan multiplicaciones para calcular ese subproducto, es decir $M_{i,i} = 0$ para $1 \leq i \leq N$

Considerando esta notación, y sabiendo que calcular el subproducto $(A_i, A_{i+1}, \ldots, A_k) * (A_{k+1}, A_{k+2}, \ldots, A_j)$ implica $p_{i-1} * p_k * p_j$ operaciones, podemos definir la siguiente relación de recurrencia:

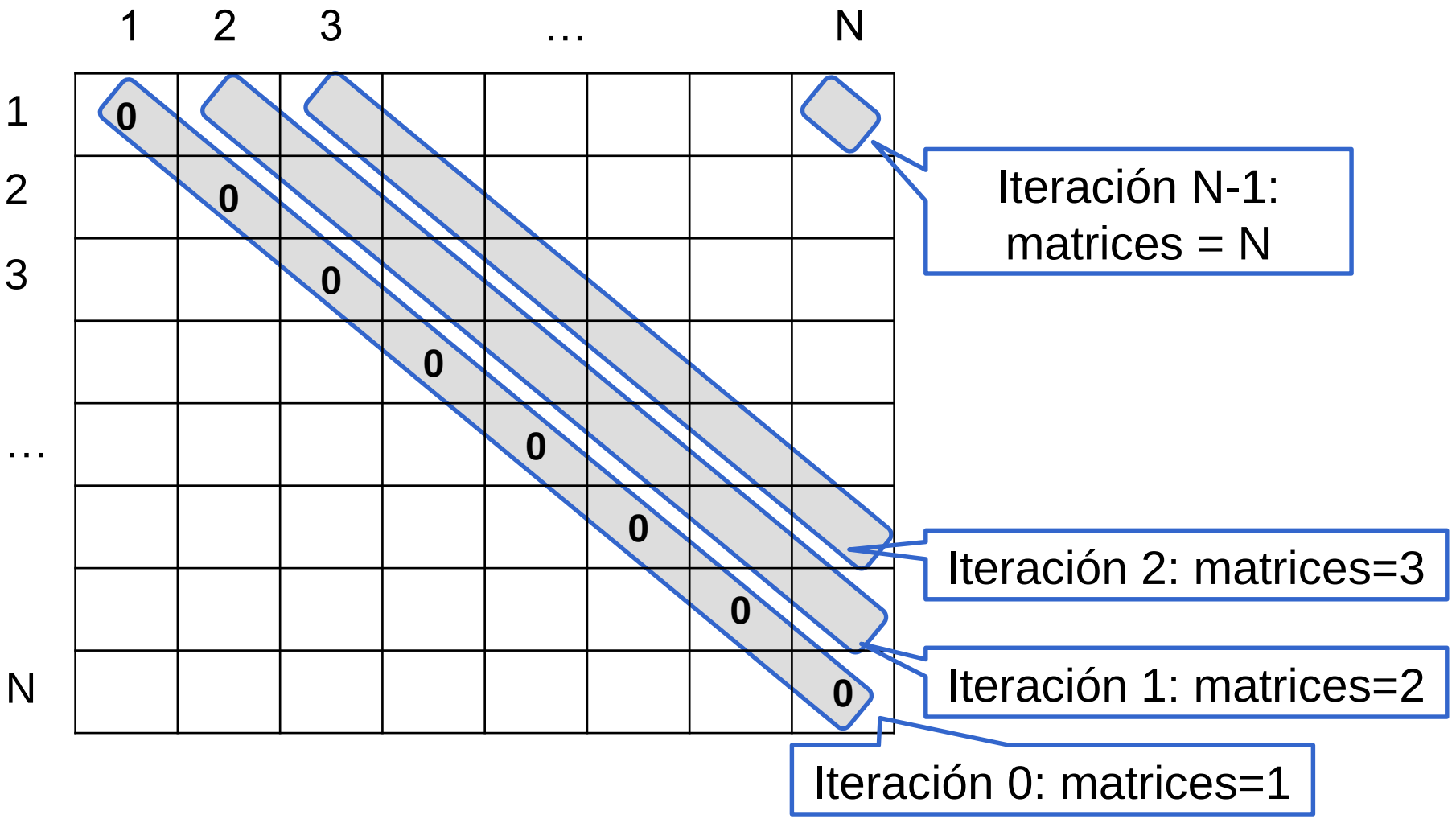
$$M[i,j] = \begin{cases} 0 & si\ i = j \\ MIN_{i \leq k < j}\left(M[i,k] + M[k+1,j] + p_{i-1}p_k p_j\right) & si\ i \neq j \end{cases}$$

En otras palabras, dados los elementos *i* a *j*, se evalúan las *j-i* posibilidades y se escoge la mejor alternativa.

Solución mediante programación dinámica

```
for i = 1 to N:
   M_{i,i} = 0
for matrices = 2 to N:
   for i = 1 to N-matrices+1:
      j = i+matrices-1
      menor = INF
      for k = i to j-1:
         menor = min(menor, M_{i,k} + M_{k+1,j} + p_{i-1}*p_k*p_j)
      M_{i,j} = menor
print(M_{1,N})
```

Solución mediante programación dinámica



¿Cuál es la complejidad de este algoritmo? $O(N^3)$