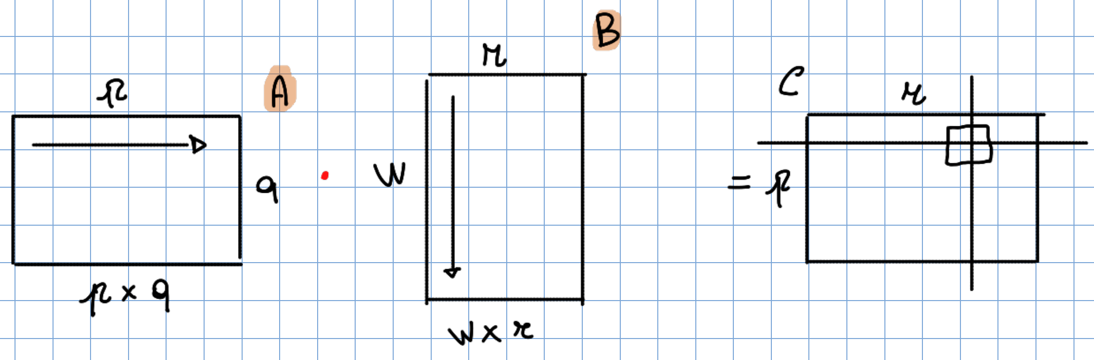
# PROG DINAMICA

Problema di ottimizzazione di moltiplicazioni tra matrici




$A \times B$

si può fare solo se $q = w$

```
MATRIX-MULTIPLY(A, B, p, q, r)
  C ← new-Matrix(p, r)
  for i ← 1 to p do:
     for J ← 1 to r do:
        C[i, J] ← 0
        for K ← 1 to q do
            C[i, J] ← A[i, q] + B[i, K] · B[K, J]
  return C
```

è lento

faccio $p \cdot q \cdot r$ di moltiplicazioni scalari

$$\underbrace{\underbrace{(\underset{P_0 \times P_1}{A} \times \underset{P_1 \times P_2}{B})}_{P_0 \times P_2} \times \underset{P_2 \times P_3}{C}}_{P_0 \times P_3}$$

← possiamo moltiplicare in qualsiasi ordine

Come ottimizziamo le moltiplicazione?

Supponiamo di avere: A, B, C

$(\underset{10\times100}{A} \times \underset{100\times5}{B}) \times \underset{5\times50}{C}$

$10 \times 100 \times 5 = 5000$

$10 \times 5 \times 50 = 2500$

$7500$

$\underset{10\times100}{A} \times (\underset{100\times5}{B} \times \underset{5\times50}{C})$

$100 \times 5 \times 50 = 25000$

$10 \times 100 \times 50 = 50000$

$75000$

→ numero di operazioni ←

Scegliere il giusto ordine di moltiplicazione rende la moltiplicazione più veloce

Come troviamo la parentesizzazione migliore per minimizzare il numero di moltiplicazioni scalari?

$A_1, A_2, A_3, A_4 \dots A_n$

$P_0 P_1 \quad P_1 P_2 \quad P_2 P_3 \quad P_3 P_4 \dots P_{n-1} P_n$

quindi abbiamo un vettore $P = P_0, P_1, P_2, P_3 \dots P_n$

$$\underset{P_{i-1}\,P_i}{A_i}$$

1) Partiamo con un approccio ricorsivo

$A_1, A_2, A_3, A_4 \dots A_n$

$(A_1 \dots A_K)(A_{K+1} \dots A_n)$

se $K=1$ o $K=n$ non faccio niente

Devo trovare il valore di k che conviene di più?

sol ottima

$\boxed{S^*_{1,n}} = S^*_{1,K} + S^*_{K+1,n} + P_0 P_K P_n$ → numero di moltiplicazioni per $A_{1,K} \times A_{K+1,n}$

numero di moltiplicazioni scalari

$A_{1,n} = A_{1,K} \times A_{K+1,n}$

$(A_1 \dots A_n) = (A_1 \dots A_K)(A_{K+1} \dots A_n)$

Dimostrazione:

$S^*_{1,n} = S_{1,K} + S^*_{K+1,n} + P_0 + P_K + P_n$ → sostituendo

non ottima quindi $S^*_{1,K} \leq S_{1,K}$

$$\underbrace{S^*_{1,n}}_{} > S^*_{1,k} + S^*_{k+1,n} + P_0 P_k P_n = \underbrace{S_{1,n}}_{}$$

Contraddizione

2) Definire la funzione ricorsiva per trovare la sol ottima

$$S_{1,n} = \begin{cases} 0 & \text{se } n = 1 \\ \min\limits_{1 \le k \le n} (S_{1,k} + S_{k+1,n} + P_0 P_k P_n) & \text{se } n > 1 \end{cases}$$

Generalizzando

$$A_i \dots A_j$$

$$(A_i \dots A_k)(A_{k+1} \dots A_j)$$

$$S_{i,j} = \begin{cases} 0 & \text{se } i = j \\ \min\limits_{i \le k < j} (S_{i,k} + S_{k+1,j} + P_{i-1} P_k P_j) & \text{se } i < j \end{cases}$$

MATRIX-CHAIN-ORDER $(p, i, j)$:

if $i = j$ then return 0

$sm = +\infty$ → elemento più grande

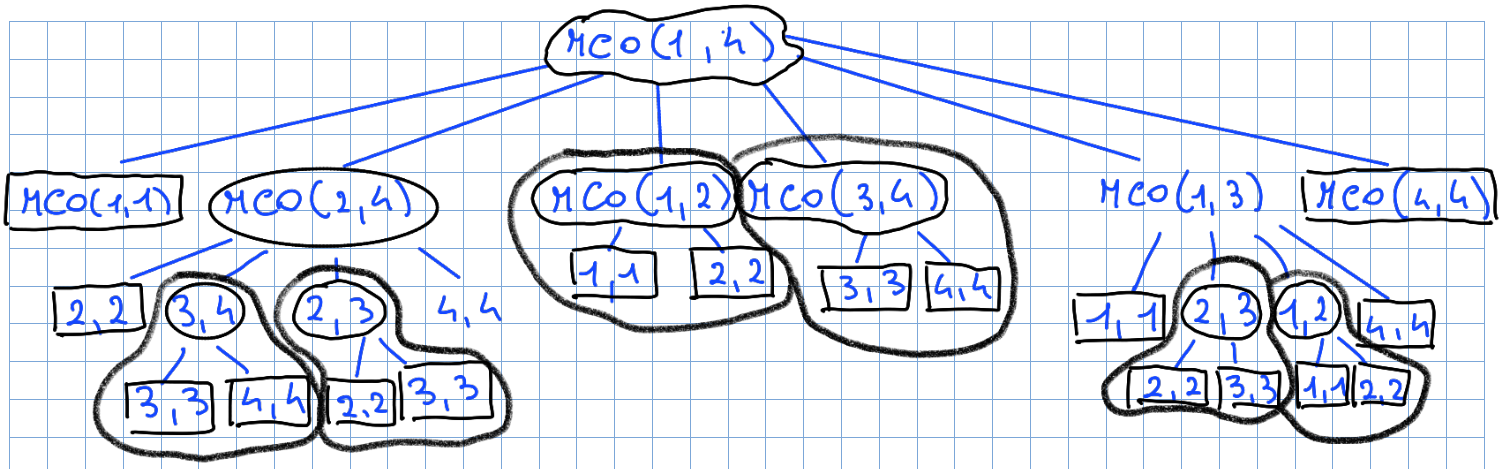
for $k \leftarrow i$ to $j-1$ DO

  $S_{i,k} \leftarrow MCO(P, i, k)$

  $S_{k+1,j} \leftarrow MCO(P, k+1, j)$

  if $Sm > S_{i,k} + S_{k+1,j} + P_{i-1} P_k P_j$ then

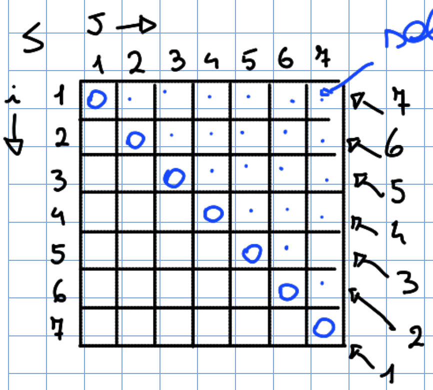
entro sicuramente la prima volta  $Sm \leftarrow S_{i,k} + S_{k+1,j} + P_{i-1} P_k P_j$




Ci sono dei sottoproblemi che si ripetono, o uso la memorizzazione o un approccio bottom-up

3) algoritmo ottimizzato

non posso usare un array




soluzione ottima

$S_{(i,j)}$

$A_i \dots A_j$

$j-i+1$ → dimensione del problema

→ $l$

Pattern di risoluzione

1 → $A_{1,2}$ $A_{2,2}$ $A_{3,3}$ $A_{4,4}$ $A_{5,5}$ $A_{6,6}$ $A_{7,7}$

2 → $A_{1,2}$ $A_{2,3}$ $A_{3,4}$ $A_{4,5}$ $A_{5,6}$ $A_{6,7}$

3 → $A_{1,3}$ $A_{2,4}$ $A_{3,5}$ $A_{4,6}$ $A_{5,7}$

4 → $A_{2,4}$ $A_{2,5}$ $A_{3,6}$ $A_{4,7}$

5 → $A_{1,5}$ $A_{2,6}$ $A_{3,7}$

6 → $A_{1,6}$ $A_{2,7}$

7 → $A_{1,7}$

```
MATRIX-CHAIN-ORDER(p, n)
        S <- new matrix(n, n)
n       for i <- 1 to n do S[i,i] <- 0              // Dim 1
n       for l <- 2 to n do                           // Dim l > 1
n-l         for i <- 1 to n-l+1 do
                j <- i + l - 1
                S[i,j] = +∞
n               For k <- 1 to j-1 do
                    if S[i,j] > S[i,k] + S[k+1,j] + P[k] x P[j]
                        S[i,j] <- S[i,k] + S[k+1,j] + P[k] x P[j]
        return S[1, n]
```

$O(n^3)$

4)

$A_1 \; A_2 \; A_3 \; A_4$

P: 2, 10, 3, 20, 4

Dim 1: $A_{12} \to A_{11}\,A_{22}$ $\quad A_{23} \to A_{22}\,A_{33}$ $\quad A_{34} \to A_{33}\,A_{44}$

Dim 2:

$A_{13}$, k=2: $A_{11}\,A_{23}$: $0+600+400$ $\quad A_{12}\,A_{33}$: $\underline{60+0+120}$

$A_{24}$, k=2: $A_{22}\,A_{34}$: $\underline{0+240+120}$ $\quad A_{23}\,A_{44}$: $600 \ldots$

scegliamo il migliore

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | $60^1$ | $180^2$ | $324^2$ |
| 2 | | 0 | $600^2$ | $360^2$ |
| 3 | | | 0 | $240^3$ |
| 4 | | | | 0 |

Dim 3:

$A_{14}$, k=2:

$A_{11}\,A_{24}$: $0+360+80$ $\quad A_{12}\,A_{34}$: $\underline{60+240+24}$ $\quad A_{13}\,A_{44}$: $180+0+160$

migliore

↓

la moltiplicazione ottimale è:

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

```
MATRIX-CHAIN-ORDER(p, m)
  S <- new matrix(m, m)
  D <- new Matrix(m, m)
  for i <- 1 to m do S[i,i] <- 0
  for l <- 2 to m do
      for i <- 1 to m-l+1 do
          J <- i + l - 1
          S[i, J] = + ∞
          For K <- 1 to J-1 do
              if S[i,J] > S[i,K] + S[K+1,J] + P[K] x P[J]
                 S[i,J] <- S[i,K] + S[K+1,J] + P[K] x P[J]
                 D[i,J] <- K
  return S[1, m]

Print-chain(D, i, J)
  if i = J then
      print(A_i)
  else
      K <- D[i,J]
      print "("
      print-chain(D, i, K)
      print-chain(D, K+1, J)
      print ")"
```

$O(m^2)$

sulle matrice di prima

$((A_1 A_2)(A_3 A_4))$