PROGRAMMAZIONE DINAMICA

1) Definire una sottostruttura ottima
2) Definire una funzione ricorsiva per il calcolo del costo di una soluzione ottima
3) Calcolo del costo di una soluzione ottima
4) Costruzione di una soluzione ottima

N.B Trovare tutte le soluzioni ottime ha un costo esponenziale

PARENTESIZZAZIONE DI UNA SEQUENZA DI MATRICI

Avendo una catena di matrici sfruttiamo la proprietà associative per velocizzare le procedure di moltiplicazioni

Per moltiplicare utilizziamo l'algoritmo MATRIX-MULTIPLY

```
MATRIX-MULTIPLY (A, B, C, p, q, r)
 FOR i ← 1 TO p DO
  FOR j ← 1 TO r DO
  C[i][j] ← 0          ottiene la matrice risultato
 FOR K ← 1 TO q DO
  C[i][j] ← C[i][j] + A[i][K] · B[K][j]
```

usando l'algoritmo ci rendiamo conto che il numero di moltiplicazioni è $p \cdot q \cdot r$

Sfruttando le proprietà associative possiamo arrivare allo stesso risultato con meno operazioni

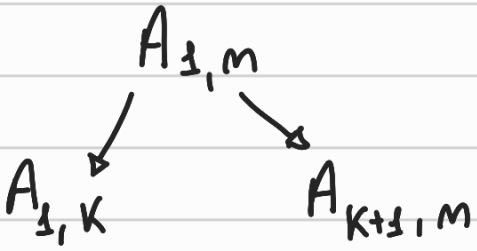
PROBLEMA di MINIMIZZAZIONE

Una sequenza parentesizzata:

$(S_1 \times S_2)$

S_1 → sequenza già parentesizzata

Quindi avendo K matrici tale che $1 \leq K < n$



Vale la sottostrutture ottima

Definiamo una soluzione:

$$m[i,j] = \begin{cases} i = j = 0 \text{ moltiplicazioni} \\ i < j \Rightarrow \min_{i \leq k < j} \left(m[i,k] + m[k+1,n] + P[i-1]P[k]P[j] \right) \end{cases}$$

Si consideri P come l'array con le dimensioni delle matrici

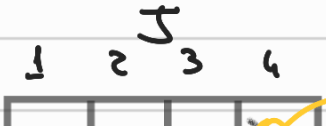
$m[i,k]$ → Costo sequenza 1

$m[k+1,n]$ → Costo sequenza 2

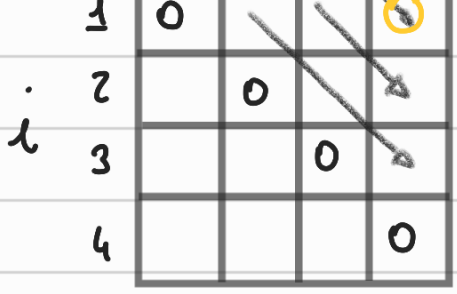
$P[i-1]P[k]P[j]$ → costo moltiplicazione

Usiamo la programmazione dinamica e memorizziamo in una matrice

La matrice avrà dimensioni $n \times n$



→ celle che avrà il nostro risultato



Costruiamo la funzione che riempie la matrice

```
MATRIX-CHAIN-FOLDER (P, n)
    m ← NEW MATRIX (n, n)
  FOR i ← 1 TO n DO
    m[i][i] ← 0
  FOR l ← 2 TO n DO
  FOR i ← 1 TO n-l+1 DO
     j ← i + l - 1
     m[i][j] ← +∞
  FOR k ← i TO j-1 DO
     q ← m[i][k] + m[k+1][j] + P[i-1] P[k] P[j]
  IF (q < m[i][j])
     m[i][j] ← q
RETURN m[i][n]
```

mettiamo gli elementi della diagonale

per scorrere le diagonale

Avremo una complessità $O(n^3)$, ma meglio di esponenziale

Possiamo usare una matrice ausiliaria che ci indica in che ordine parentesizzare