ALGORITMO T = MOCHILA (M, B, W, N)

#RELLENAR CASOS BASE

T = MATRIZ DE N FILAS 1…N; M+1 COLUMNAS 0…M

PARA CADA FILA i = 1…N:

T[i][0] = 0

PARA CADA COLUMNA j=0…M:

Si j >= B[1]:

T[1][j] = Bi //Capacidad de mochila suficiente para llevar el obj1

EN OTRO CASO

T[1][j] = 0 //No hay capacidad para llevar el obj1

#RELLENAR CASO GENERAL

PARA CADA FILA i = 2...N:

PARA CADA COLUMNA j = 1...M: //Vemos si cabe i en la mochila

Si j - wi >= 0:

T[i][j] = max { Bi + T[i-1][j-wi], T[i-1][j]}

En otro caso:

T[i][j] = T[i-1][j]

FIN-PARA

FIN-PARA

V = T[N][M] //Máximo beneficio de considerar llevarnos todos los objetos del 1 hasta n para la capacidad m

DEVOLVER V

**CAMINOS MÍNIMOS**

**Sea G=(V,A) un grafo dirigido (también no dirigidos), con arcos ponderados con pesos no negativos. El problema consiste en hallar el camino mínimo (secuencia de arcos que unen un nodo inicial y un nodo destino, cuya suma de pesos es mínima) entre cualquier par de nodos del grafo.**

Análisis del problema

n nodos: 1...n

Lnxn -> Matriz de adyacencia

L(i,j) -> Coste de la arista (i,j)

Dk(i,j) //Distancia minima de i -> j considerando a k como nodo intermedio

Do(i,j) = L(i,j)

Problema N-etapico

En cada etapa k se considera pasar por el nodo intermedio k para ir de i a j.

En cada etapa k se deberán tomar unas decisiones:

Pasar por k en el camino de i a j : Dk(i,j) = Dk-1(i,k) + Dk-1(k,j)

No pasar por k: Dk(i,j) = Dk-1(i,j)

**Dk(i,j) = min {Dk-1(i,k) + Dk-1(k,j), Dk-1(i,j)} --> Resolución por recurrencias**

**Caso Base:** Do(i,j) = L(i,j) **//No pasar por ningún nodo intermedio, coste del arco que los une, si no hay arco** Do(i,j) = INFINITO

**OBJETIVO:** Dn(i,j) //Parará cuando conozcamos ese objetivo

**Estructura de Datos:**

**D(i,j): TABLA DONDE i ES EL NODO ORIGEN Y j ES EL NODO DESTINO, Y D(i,j) ES LA DISTANCIA MÍNIMA CONOCIDA DESDE i HASTA j**

**PRINCIPIO DE OPTIMALIDAD DE BELMANN:**

**D0[i][j] es óptimo: El mejor camino de i a j sin pasar por ningún nodo es D[i][j]. D1[i][j] también es óptimo: Selecciona el mejor camino**

**entre ir directos de i a j o pasando por el nodo 1.**

**... Dk[i][j] es óptimo: En caso contrario, habría otros nodos en el**

**camino de i a j pasando por {1...k-1} tal que su coste sea menor que el considerado. Esto es imposible, dado que la ecuación recurrente siempre selecciona el menor coste.**

**ALGORITMO DE FLOYD: //P(i,j) = nodo intermedio para ir de i hasta j**

Procedimiento Floyd(MatridAdy[1..n][1..n])

//Caso base

Para i=1 hasta n, hacer:

Para j=1 hasta n, hacer:

D[i][j]= MatrizAdy[i][j] // Long. del camino mínimo conocido inicial

P[i][j]= 0 // Para ir desde i a j inicialmente no se pasa por ningún nodo

Fin-Para

Fin-Para

//Caso general

Para k=1 hasta n, hacer: //Resolver cada etapa k

Para i=1 hasta n, hacer:

Para i=1 hasta n, hacer:

Si D[i][j]> D[i][k]+D[k][j], entonces: //Casos ecuación en recurrencia min...

D[i][j]= D[i][k]+D[k][j]

P[i][j]= k //Nodo intermedio

Fin-Si

Fin-Para

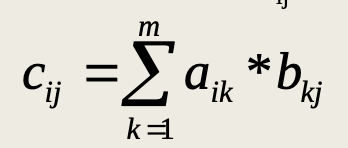
Fin-Para

Fin-Para

Devolver D,P

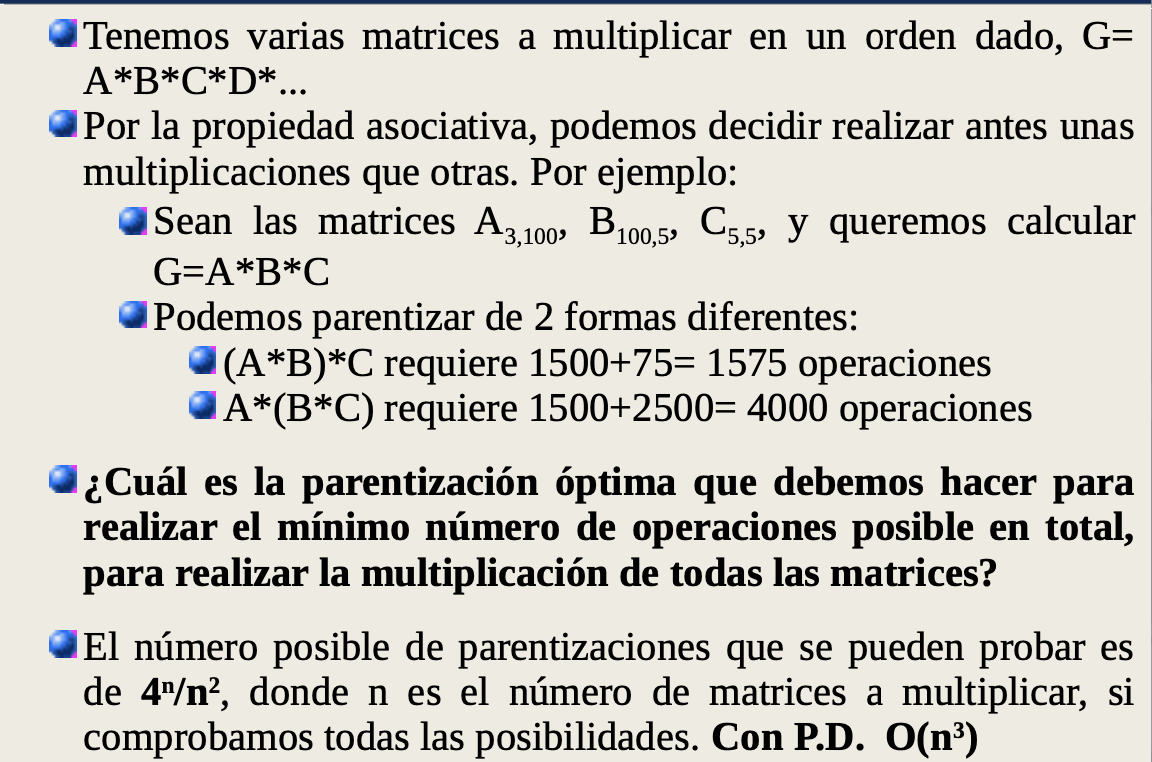
**MULTIPLICACIÓN DE MATRICES.**

**Sean 2 matrices An,m y Bm,p. Su producto es Cn,p= A\*B. Cada componente de la matriz C, cij, se calcula:**

****

**Operaciones: Hay que calcular n\*p coeficientes cij, y para calcular cada coeficiente es necesario hacer m multiplicaciones. Total: n\*p\*m.**

**ENUNCIADO DEL PROBLEMA:**

****