

PREDA - UNED

Programación y Estructuras de Datos Avanzadas

Capítulo 5

Programación dinámica

Planteamiento

- Almacenar resultados parciales ya calculados para reutilizarlos repetidamente durante la resolución del problema, reduciendo el coste de cómputo
- Comparte algunos casos de aplicación con el esquema "divide y vencerás"
- Conviene si:
 - existen llamadas recursivas repetidas
 - se cumple el **Principio de Optimalidad de Bellman**
 - dada una secuencia óptima de decisiones, toda sub secuencia de ella es, a su vez, óptima
- Se suele usar una tabla para guardar los resultados parciales (coste temporal vs. coste espacial)

PD vs. DyV

- PD
 - 1. Divide el problema en subproblemas
 - 2. Los subproblemas son dependientes entre sí
 - 3. Se almacena la solución de un subproblema
 - 4. Algoritmo Bottom-Up
- DyV
 - 1. Divide el problema en subproblemas
 - 2. Los subproblemas son independientes entre sí
 - 3. No se almacena la solución de un subproblema
 - 4. Algoritmo Top-Down

Proceso

- Establecimiento de las ecuaciones que representan el problema
- Identificación de los resultados parciales
- Construcción de la tabla de resultados parciales:
 - Inicialización de la tabla con los casos base que establece la ecuación del problema
 - Establecimiento del orden de llenado de la tabla, de forma que se calculen en primer lugar los resultados parciales que requieren pasos posteriores
 - Sustitución de las llamadas recursivas del algoritmo por consultas a la tabla

Ejemplo

- $Fibonacci(n) = \begin{cases} 1 & \text{si } n = 0, 1 \\ Fibonacci(n - 1) + Fibonacci(n - 2) & \text{si } n > 1 \end{cases}$

```
fun Fib(n: entero): entero
    si n ≤ 1 entonces
        dev 1
    sino
        dev Fib(n-1) + Fib(n-2)
    fsi
ffun
```

```
fun FibDin(n: entero): entero
var
    i,suma: entero
    t: tabla[0..n] de entero
fvar
    si n ≤ 1 entonces
        dev 1
    sino
        t[0] ← 1
        t[1] ← 1
        para i ← 2 hasta n hacer
            t[i] ← t[i-1] + t[i-2]
        fpara
    fsi
ffun
```

Añadir:

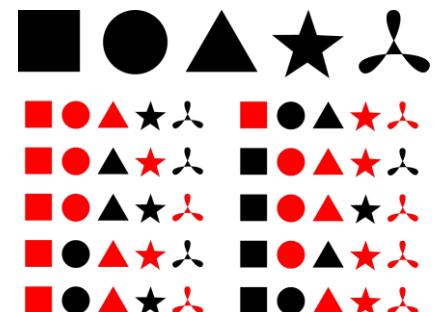
dev t[n]

Los coeficientes binomiales

- Coeficientes binomiales, números combinatorios o combinaciones:
$$\binom{n}{k}$$
- Se utilizan para calcular combinaciones de elementos (número de formas de extraer distintos subconjuntos dado un conjunto, sin importar su orden)
 - Ejemplo: ¿Formas de escoger 3 elementos de 5?

$$\binom{5}{3} = 10$$

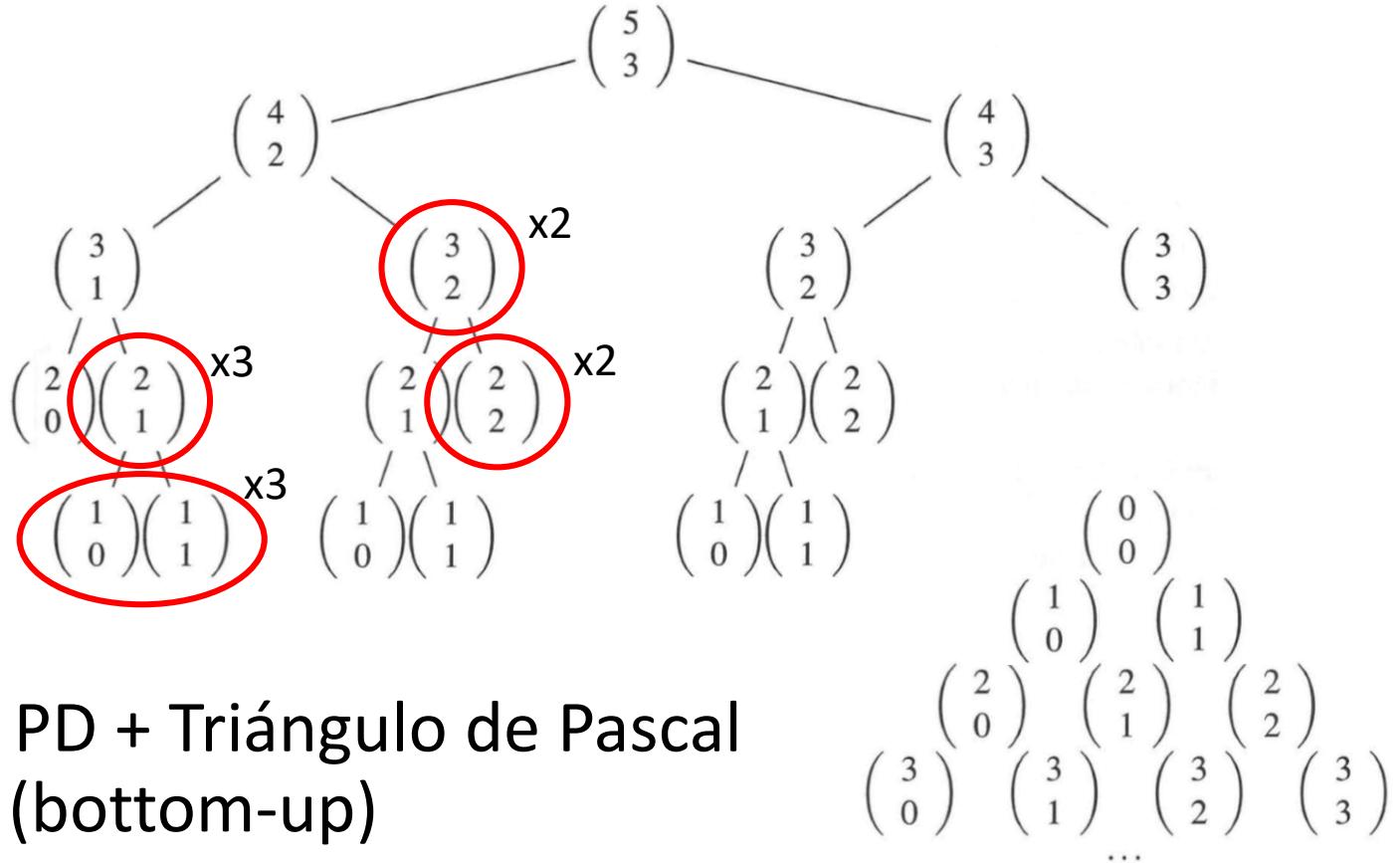
opciones marcadas en rojo



- De complejidad exponencial (versión recursiva) a $O(nk)$ con PD

Los coeficientes binomiales

- Definición: $\binom{n}{k} = \frac{n!}{k!(n-k)!} = \begin{cases} 1 & \text{si } k=0 \text{ ó } k=n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{si } 0 < k < n \end{cases}$
- Ejemplo:



```

fun CoefBin(n,k: entero): entero
    var
        i,j: entero
        t: matriz[0..n, 0..k] de entero
    fvar
        si k ≤ 0 ∨ k = n entonces
            dev 1
        sino
            para i ← 0 hasta n hacer t[i]
                para i ← 1 hasta n hacer t[i]
                para i ← 2 hasta k hacer t[i]
                para i ← 3 hasta n hacer
                    para j ← 2 hasta i-1 hacer
                        si j ≤ k entonces
                            t[i,j] ← t[i-1,j-1] +
                        fsi
                    fpara
                fpara
            fpara
        fsi
    ffun
    Añadir:
    dev t[n, k]

```

omiales

si $k = 0$ ó $k = n$

$$+ \binom{n-1}{k} \quad \text{si } 0 < k < n$$

$$\begin{pmatrix} 5 \\ 3 \end{pmatrix} \xrightarrow{\hspace{1cm}} \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \quad \quad \quad \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

$$\times 2$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Pascal

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$\begin{pmatrix} 2 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$
$$\begin{pmatrix} 3 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 3 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 3 \\ 2 \end{pmatrix} \quad \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

- Solución: PD + Triángulo de Pascal
(bottom-up)

```

fun CoefBin(n,k: entero): entero
    var
        i,j: entero
        t: matriz[0..n, 0..k] de entero
    fvar
        si k ≤ 0 ∨ k = n entonces
            dev 1
        sino
            para i ← 0 hasta n hacer t[i,0] ← 1 fpara
            para i ← 1 hasta n hacer t[i,1] ← i fpara
            para i ← 2 hasta k hacer t[i,i] ← 1 fpara
            para i ← 3 hasta n hacer
                para j ← 2 hasta i-1 hacer
                    si j ≤ k entonces
                        t[i,j] ← t[i-1,j-1] + t[i-1,j]
                    fsi
                fpara
            fpara
            ffun
    
```

Añadir:
dev t[n, k]

- Solución: PD (bo)

binomiales

$$1 \choose 1 \Big) + {n-1 \choose k} \quad \begin{array}{l} \text{si } k=0 \text{ ó } k=n \\ \text{si } 0 < k < n \end{array}$$

$${5 \choose 3} \longrightarrow \begin{array}{c} {4 \choose 3} \\ / \quad \backslash \\ {3 \choose 2} \quad {3 \choose 3} \end{array}$$

	0	1	2	3	...	$k-1$	k
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
...		
$n-1$	$C(n-1,k-1)$	$C(n-1,k)$
n	$C(n-1,k-1) + C(n-1,k)$

fun CoefBin(n,k: entero): entero

var

i,j: entero

t: matriz[0..n, 0..k] de entero

fvar

si $k \leq 0 \vee k = n$ **entonces**

dev 1

sino

para i $\leftarrow 0$ **hasta** n **hacer** t[i,0] $\leftarrow 1$ **fpara**

para i $\leftarrow 1$ **hasta** n **hacer** t[i,1] $\leftarrow i$ **fpara**

para i $\leftarrow 2$ **hasta** k **hacer** t[i,i] $\leftarrow 1$ **fpara**

para i $\leftarrow 3$ **hasta** n **hacer**

para j $\leftarrow 2$ **hasta** i-1 **hacer**

si j $\leq k$ **entonces**

t[i,j] \leftarrow t[i-1,j-1] + t[i-1,j]

fsi

fpara

fpara Añadir:

dev t[n, k]

ffun

- Solución: PD (bo)

binomiales

$$\binom{n}{k} = \begin{cases} 1 & \text{si } k = 0 \text{ ó } k = n \\ \binom{n-1}{k} + \binom{n-1}{k-1} & \text{si } 0 < k < n \end{cases}$$

$$\binom{5}{3} \rightarrow \binom{4}{3} \rightarrow \binom{3}{2} \rightarrow \binom{3}{2}$$

	0	1	2	3	...	$k-1$	k
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
...		
$n-1$	$C(n-1, k-1)$	$C(n-1, k)$
n	$C(n-1, k-1) + C(n-1, k)$

```

fun CoefBin(n,k: entero): entero
    var
        i,j: entero
        t: matriz[0..n, 0..k] de entero
    fvar
        si k ≤ 0 ∨ k = n entonces
            dev 1
        sino
            para i ← 0 hasta n hacer t[i,0] ← 1 fpara
            para i ← 1 hasta n hacer t[i,1] ← i fpara
            para i ← 2 hasta k hacer t[i,i] ← 1 fpara
            para i ← 3 hasta n hacer
                para j ← 2 hasta i-1 hacer
                    si j ≤ k entonces
                        t[i,j] ← t[i-1,j-1] + t[i-1,j]
                    fsi
                fpara
            fpara
        fsi
    ffun

```

Añadir:

dev t[n, k]

- Solución: PD (bo)

binomiales

$$\binom{n}{k} = \begin{cases} 1 & \text{si } k = 0 \text{ ó } k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{si } 0 < k < n \end{cases}$$

$$\binom{5}{3} = \binom{4}{3} + \binom{3}{2}$$

$$\binom{4}{3} = \binom{3}{2} + \binom{3}{2}$$

	0	1	2	3	...	$k-1$	k
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
...		
$n-1$	$C(n-1,k-1)$	$C(n-1,k)$
n	$C(n-1,k-1) + C(n-1,k)$

Solución

Devolución del cambio

- Con N tipos de monedas
(imposible con algoritmo voraz)
- Coste O(NC)
- Construcción de la tabla
 - $T[i,j] = \text{Nº monedas de tipo } x_i \text{ o menos para la cantidad } C_j$

	0	1	2	3	4	5	6	7	8	9	10	11	12
$x_1 = 1$	0	1	2	3	4	5	6	7	8	9	10	11	12
$x_2 = 6$	0	1	2	3	4	5	1	2	3	4	5	6	2
$x_3 = 10$	0	1	2	3	4	5	1	2	3	4	1	2	2

- Posibilidad de usar un algoritmo voraz ($O(C)$) para saber las monedas usadas

```
tipo Tabla = matriz[1..N, 0..C] de entero
tipo Vector = matriz[0..N] de entero
fun DarCambio(C: entero, moneda: Vector): Tabla
    var
        t: Tabla
        i,j: entero
    fvar
    para i ← 1 hasta N hacer
        t[i,0] ← 0
    fpara
    para j ← 1 hasta C hacer
        para i ← 1 hasta N hacer
            si i = 1 ∧ moneda[i] > j entonces
                t[i,j] ← ∞
            sino
                si i = 1 entonces
                    t[i,j] ← 1 + t[1,j-moneda[i]]
                sino
                    si j < moneda[i] entonces
                        t[i,j] ← t[i-1,j]
                    sino
                        t[i,j] ← min(t[i-1,j], t[i,j-moneda[i]] + 1)
                fsi
            fsi
        fpara
        dev t
ffun
```

Devolución del cambio

- Con N tipos (imposible combinar)
- Coste $O(NC)$

```

tipo Tabla = matriz[1..N, 0..C] de entero
tipo Vector = matriz[0..N] de entero
fun DarCambio(C: entero, moneda: Vector): Tabla
    var

```

Ejemplo:

- Monedas = {1, 4, 6}
- C = 8

$$T[i,j] = t[i-1,j] \text{ si } x_i > j$$

$$T[i,j] = \min (t[i-1,j] , t[i, j-x_i] + 1) \text{ si } x_i \leq j$$

para i ← 1 hasta N hacer

 t[i, 0] ← 0 para j ← 1 hasta C hacer

 t[i, j] ← +∞ para k ← 1 hasta N hacer

 si $x_k \leq j$ entonces $t[i, j] ← \min (t[i, j], t[i, j - x_k] + 1)$

 fin si

 fin para k

 fin para j

fin para i

T	0	1	2	3	4	5	6	7	8
$x_1=1$ $\{1\}$	0								
$x_2=4$ $\{1,4\}$	0								
$x_3=6$ $\{1,4,6\}$	0								

algoritmo voraz ($O(C)$)
para saber las
monedas usadas

ISI
fpara
fpara
dev t
ffun

+ 1)

Devolución del cambio

- Con N tipos
(imposible combinar)
 - Coste $O(N^N)$

tipo Tabla = matriz[1..N, 0..C] de entero

tipo Vector = matriz[0..N] de entero

```
fun DarCambio(C: entero, moneda: Vector): Tabla  
    var
```

Ejemplo:

- Monedas = {1, 4, 6}
 - $C = 8$

$$T[i,j] = t[i-1,j] \text{ si } x_i > j$$

$$T[i,j] = \min (t[i-1,j] , t[i, j-x_i] + 1) \text{ si } x_i \leq j$$

para $i \leftarrow 1$ hasta N hacer

T	0	1	2	3	4	5	6	7	8
$x_1=1$ $\{1\}$	0	1 +1 1	2 +1 2	3 +1 3					
$x_2=4$ $\{1,4\}$	0	1 1	2 2	3 3					
$x_3=6$ $\{1,4,6\}$	0	1 1	2 2	3 3					

algoritmo voraz ($O(C)$)
para saber las
monedas usadas

IS
fpara
fpara
dev t
ffun

Devolución del cambio

- Con N tipos
(imposible combinar)
 - Coste $O(N^N)$

tipo Tabla = matriz[1..N, 0..C] de entero

tipo Vector = matriz[0..N] de enteros

```
fun DarCambio(C: entero, moneda: Vector): Tabla  
var
```

Ejemplo:

- Monedas = {1, 4, 6}
 - $C = 8$

$$T[i,j] = t[i-1,j] \text{ si } x_i > j$$

$$T[i,j] = \min (t[i-1,j] , t[i, j-x_i] + 1) \text{ si } x_i \leq j$$

para $i \leftarrow 1$ hasta N hacer

T	0	1	2	3	4	5	6	7	8
$x_1=1$ $\{1\}$	0	1	2	3	4				
$x_2=4$ $\{1,4\}$	0	1	2	3	1				
$x_3=6$ $\{1,4,6\}$	0	1	2	3	1				

algoritmo voraz ($O(C)$)
para saber las
monedas usadas

IS
fpara
fpara
dev t
ffun

Devolución del cambio

- Con N tipos (imposible con 1 tipo)
- Coste O(NC)

```

tipo Tabla = matriz[1..N, 0..C] de entero
tipo Vector = matriz[0..N] de entero
fun DarCambio(C: entero, moneda: Vector): Tabla
    var

```

Ejemplo:

- Monedas = {1, 4, 6}
- C = 8

$$T[i,j] = t[i-1,j] \text{ si } x_i > j$$

$$T[i,j] = \min (t[i-1,j] , t[i, j-x_i] + 1) \text{ si } x_i \leq j$$

para i ← 1 hasta N hacer

 t[i, 0] ← 0 para j ← 1 hasta C hacer

 t[i, j] ← min (t[i-1, j], t[i, j-x_i] + 1)

T	0	1	2	3	4	5	6	7	8
$x_1=1$ $\{1\}$	0	1	2	3	4	5			
$x_2=4$ $\{1,4\}$	0	1	2	3	1	2			
$x_3=6$ $\{1,4,6\}$	0	1	2	3	1	2			

algoritmo voraz (O(C))
para saber las
monedas usadas

ISI
fpara
fpara
dev t
ffun

+ 1)

Devolución del cambio

- Con N tipos (imposible con voraz)
- Coste O(NC)

```

tipo Tabla = matriz[1..N, 0..C] de entero
tipo Vector = matriz[0..N] de entero
fun DarCambio(C: entero, moneda: Vector): Tabla
    var

```

Ejemplo:

- Monedas = {1, 4, 6}
- C = 8

$$T[i,j] = t[i-1,j] \text{ si } x_i > j$$

$$T[i,j] = \min (t[i-1,j] , t[i, j-x_i] + 1) \text{ si } x_i \leq j$$

para i ← 1 hasta N hacer

 t[i, 0] ← 0 para j ← 0 hasta C hacer

 t[i, j] ← min (t[i-1, j], t[i, j-x_i] + 1)

T	0	1	2	3	4	5	6	7	8
$x_1=1$ $\{1\}$	0	1	2	3	4	5	6		
$x_2=4$ $\{1,4\}$	0	1	2	3	1	2	3		
$x_3=6$ $\{1,4,6\}$	0	1	2	3	1	2	1		

algoritmo voraz (O(C))
para saber las
monedas usadas

ISI
fpara
fpara
dev t
ffun

+ 1)

Devolución del cambio

- Con N tipos
(imposible cor)
- Coste O(N)

```
tipo Tabla = matriz[1..N, 0..C] de entero  
tipo Vector = matriz[0..N] de entero  
fun DarCambio(C: entero, moneda: Vector): Tabla  
    var
```

Ejemplo:

- Monedas = {1, 4, 6}
 - C = 8

$$T[i,j] = t[i-1,j] \text{ si } x_i > j$$

$$T[i,j] = \min (t[i-1,j] , t[i, j-x_i] + 1) \text{ si } x_i \leq j$$

T	0	1	2	3	4	5	6	7	8
$x_1=1$ $\{1\}$	0	1	2	3	4	5	6	7	
$x_2=4$ $\{1,4\}$	0	1	2	3	1	2	3	4	
$x_3=6$ $\{1,4,6\}$	0	1	2	3	1	2	1	2	

algoritmo voraz ($O(C)$)
para saber las
monedas usadas

ISI
fpara
fpara
dev t
ffun

Devolución del cambio

- Con N tipos (imposible combinar)
- Coste $O(N^N)$

```
tipo Tabla = matriz[1..N, 0..C] de entero  
tipo Vector = matriz[0..N] de entero  
fun DarCambio(C: entero, moneda: Vector): Tabla  
    var
```

Ejemplo:

- Monedas = {1, 4, 6}
 - C = 8

$$T[i,j] = t[i-1,j] \text{ si } x_i > j$$

$$T[i,j] = \min (t[i-1,j] , t[i, j-x_i] + 1) \text{ si } x_i \leq j$$

T	0	1	2	3	4	5	6	7	8
$x_1=1$ $\{1\}$	0	1	2	3	4	5	6	7	8
$x_2=4$ $\{1,4\}$	0	1	2	3	1	2	3	4	2
$x_3=6$ $\{1,4,6\}$	0	1	2	3	1	2	1	2	2

The diagram shows three blue curved arrows indicating transitions between states. One arrow points from state 7 to state 8, labeled '+1'. Another arrow points from state 1 to state 2, labeled '+1'. A third arrow points from state 2 to state 2, also labeled '+1'.

algoritmo voraz ($O(C)$)
para saber las
monedas usadas

IS
fpara
fpara
dev t
ffun

Ejercicio de examen

2. Sea el problema de la devolución de cambio con monedas de valores 1,6 y 10 solucionado con programación dinámica para pagar una cantidad de 12 unidades. Identifica cuál de las siguientes respuestas correspondería al contenido de la tabla de resultados parciales de cantidades en la fila correspondiente a la moneda de valor 6, si dichas monedas se consideran por orden creciente de valores:
- (a) 0 1 2 3 4 5 6 2 3 4 5 6 3
 - (b) 0 1 2 3 4 5 6 2 3 4 5 6 7
 - (c) 0 1 2 3 4 5 1 2 3 4 5 6 2
 - (d) Ninguna de las anteriores.

Ejercicio de examen

2. Sea el problema de la devolución de cambio con monedas de valores 1,6 y 10 solucionado con programación dinámica para pagar una cantidad de 12 unidades. Identifica cuál de las siguientes respuestas correspondería al contenido de la tabla de resultados parciales de cantidades en la fila correspondiente a la moneda de valor 6, si dichas monedas se consideran por orden creciente de valores:

- (a) 0 1 2 3 4 5 6 2 3 4 5 6 3
(b) 0 1 2 3 4 5 6 2 3 4 5 6 7
(c) 0 1 2 3 4 5 1 2 3 4 5 6 2
(d) Ninguna de las anteriores.

$$T[i,j] = t[i-1,j] \text{ si } x_i > j$$

$$T[i,j] = \min (t[i-1,j] , t[i, j-x_i] + 1) \text{ si } x_i \leq j$$

Ejercicio de examen

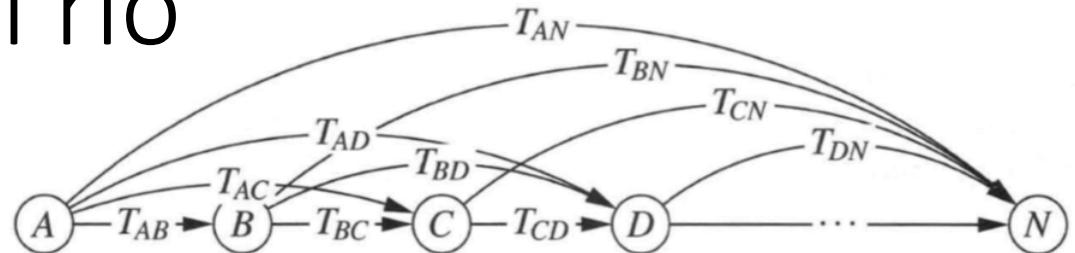
2. Sea el problema de la devolución de cambio con monedas de valores 1,6 y 10 solucionado con programación dinámica para pagar una cantidad de 12 unidades. Identifica cuál de las siguientes respuestas correspondería al contenido de la tabla de resultados parciales de cantidades en la fila correspondiente a la moneda de valor 6, si dichas monedas se consideran por orden creciente de valores:

- (a) 0 1 2 3 4 5 6 2 3 4 5 6 3
(b) 0 1 2 3 4 5 6 2 3 4 5 6 7
(c) 0 1 2 3 4 5 1 2 3 4 5 6 2
(d) Ninguna de las anteriores.



	0	1	2	3	4	5	6	7	8	9	10	11	12
$x_1 = 1$	0	1	2	3	4	5	6	7	8	9	10	11	12
$x_1 = 6$	0	1	2	3	4	5	1	2	3	4	5	6	2
$x_1 = 10$	0	1	2	3	4	5	1	2	3	4	1	2	2

El viaje por el río

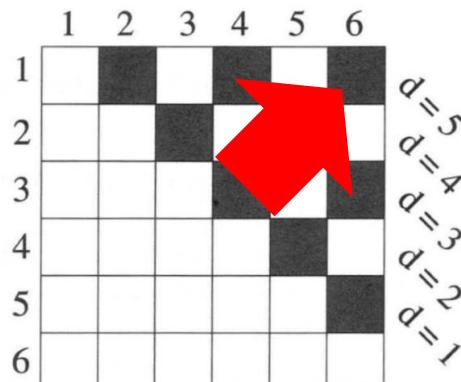


- ¿Cuál es la forma más barata de ir de uno de los embarcaderos a cualquier otro que esté río abajo?

$$C(i, j) = \begin{cases} 0 & \text{si } i = j \\ \min_{i < k \leq j} \{T[i, k] + C(k, j)\} & \text{si } i < j \end{cases}$$

T: Tabla de precios
C: Tabla de costes mínimos

- Construcción de la tabla C por diagonales

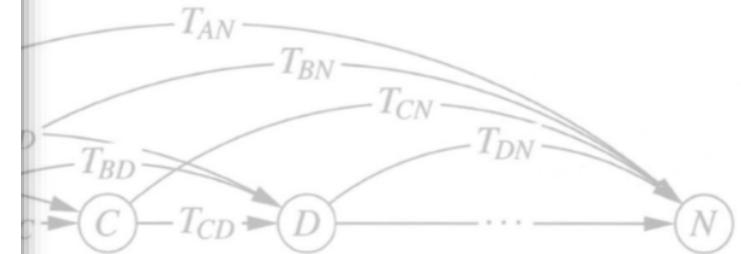


```

tipo Tabla = matriz[1..N,1..N] de entero
fun ViajeRío(T: Tabla, N: entero, C: Tabla)
    var
        i,diag: entero
    fvar
    para i  $\leftarrow$  1 hasta N hacer
        C[i,i]  $\leftarrow$  0
    fpara
    para diag  $\leftarrow$  1 hasta N-1 hacer
        para i  $\leftarrow$  1 hasta N-diag hacer
            C[i,i+diag]  $\leftarrow$  MinMultiple(C,i,i+diag)
        fpara
    fpara
ffun

```

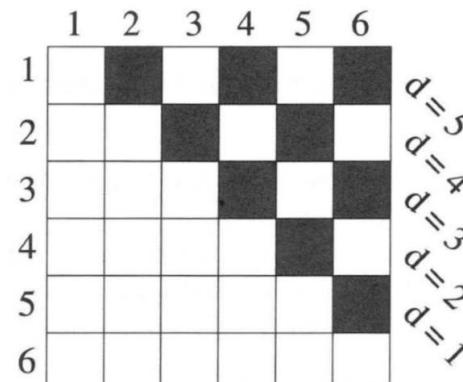
CONSTRUCCIÓN DE LA TABLA C POR diagonales



de ir de uno de los
o que esté río abajo?

si $i = j$
si $i < j$

T: Tabla de precios
C: Tabla de costes
mínimos



```

tipo Tabla = matriz[1..N,1..N] de entero
fun ViajeRío(T: Tabla, N: entero, C: Tabla)

```

var

i,diag: entero

fvar

para i \leftarrow 1 hasta N **hacer**

C[i,i] \leftarrow 0

fpara

para diag \leftarrow 1 hasta N-1 **hacer**

para i \leftarrow 1 hasta N-diag **hacer**

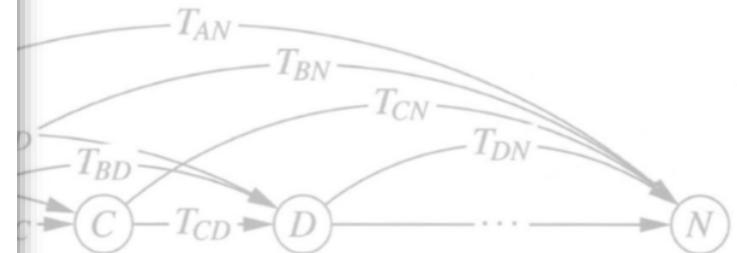
C[i,i+diag] \leftarrow MinMultiple(C,i,i+diag)

fpara

fpara

ffun

CONSTRUCCIÓN DE LA TABLA C POR



de ir de uno de los
o que esté río abajo?

si $i = j$

si $i < j$

T: Tabla de precios

C: Tabla de costes

mínimos

diagonales

	1	2	3	4	5	6
1	0					
2		0				
3			0			
4				0		
5					0	
6						0

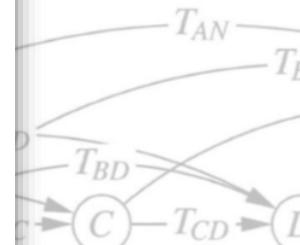
$d=5$
 $d=4$
 $d=3$
 $d=2$
 $d=1$

```

tipo Tabla = matriz[1..N,1..N] de entero
fun ViajeRio(T: Tabla, N: entero, C: Tabla)
    var
        i,diag: entero
    fvar
    para i  $\leftarrow$  1 hasta N hacer
        C[i,i]  $\leftarrow$  0
    fpara
    para diag  $\leftarrow$  1 hasta N-1 hacer
        para i  $\leftarrow$  1 hasta N-diag hacer
            C[i,i+diag]  $\leftarrow$  MinMultiple(C,i,i+diag)
        fpara
    fpara
ffun

```

diag	i	i+diag
1	1	2
1	2	3
1	3	4
1	4	5
1	5	6
2	1	3
2	2	4
2	3	5
2	4	6
3	1	4
3	2	5
3	3	6
4	1	5
4	2	6
5	1	6

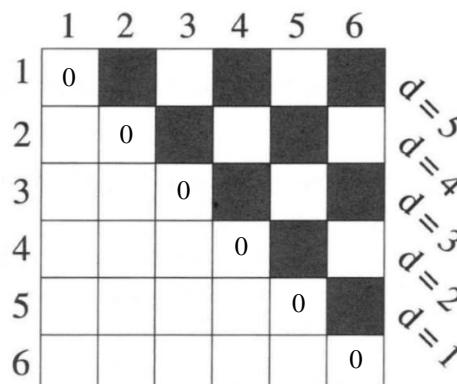


de ir de un
que esté

si $i = j$

si $i < j$

diagonale



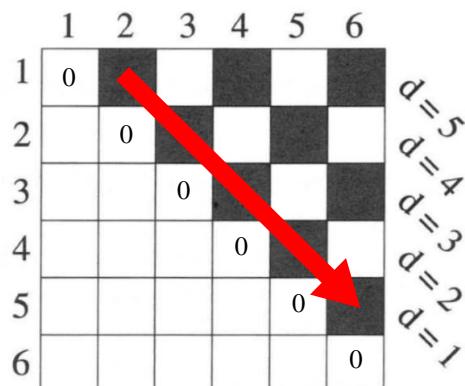
```

tipo Tabla = matriz[1..N,1..N] de entero
fun ViajeRio(T: Tabla, N: entero, C: Tabla)
    var
        i,diag: entero
    fvar
    para i  $\leftarrow$  1 hasta N hacer
        C[i,i]  $\leftarrow$  0
    fpara
    para diag  $\leftarrow$  1 hasta N-1 hacer
        para i  $\leftarrow$  1 hasta N-diag hacer
            C[i,i+diag]  $\leftarrow$  MinMultiple(C,i,i+diag)
        fpara
    fpara
ffun

```

The diagram illustrates a 6x6 grid with nodes labeled A, B, C, and D. Arrows indicate transitions between nodes: T_{AN} from A to N, T_{BA} from B to A, T_{BD} from B to D, and T_{CD} from C to D. Below the grid is a table with columns labeled "diag", "i", and "i+diag". The table rows are numbered 1 through 5. The first five rows have three columns, while the last row has two columns. The data is as follows:

diag	i	i+diag
1	1	2
1	2	3
1	3	4
1	4	5
1	5	6
2	1	3
2	2	4
2	3	5
2	4	6
3	1	4
3	2	5
3	3	6
4	1	5
4	2	6
5	1	6



```

tipo Tabla = matriz[1..N,1..N] de entero
fun ViajeRio(T: Tabla, N: entero, C: Tabla)
    var
        i,diag: entero
    fvar
    para i  $\leftarrow$  1 hasta N hacer
        C[i,i]  $\leftarrow$  0
    fpara
    para diag  $\leftarrow$  1 hasta N-1 hacer
        para i  $\leftarrow$  1 hasta N-diag hacer
            C[i,i+diag]  $\leftarrow$  MinMultiple(C,i,i+diag)
    fpara
    fpara
ffun

```

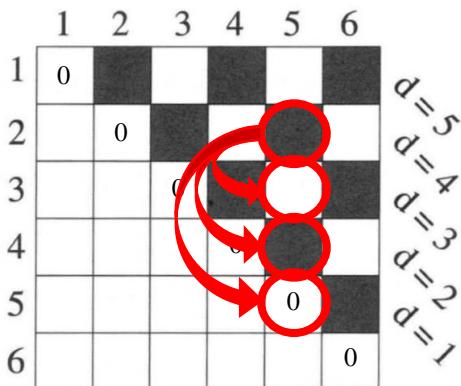


Diagram illustrating the construction of a matrix T for the Traveling Salesman Problem (TSP). The matrix T is an N x N matrix where N=6. The columns are labeled 'diag', 'i', and 'i+diag'. The matrix values are:

diag	i	i+diag
1	1	2
1	2	3
1	3	4
1	4	5
1	5	6
2	1	3
2	2	4
2	3	5
2	4	6
3	1	4
3	2	5

The value 5 is circled in red, corresponding to the entry T[4,2].

```

fun MinMultiple(C: Tabla, i: entero, j: entero): entero
    var
        k, minimo: entero
    fvar
        minimo  $\leftarrow$   $\infty$ 
    para k  $\leftarrow$  i+1 hasta j hacer
        minimo  $\leftarrow$  min(minimo, C[k,j] + T[i,k])
    fpara
    dev minimo
ffun

```

Añadir T como 1er parámetro en declaración y llamada a MinMultiple

```

tipo Tabla = matriz[1..N,1..N] de entero
fun ViajeRio(T: Tabla, N: entero, C: Tabla)
  var
    i,diag: entero
  fvar
  para i  $\leftarrow$  1 hasta N hacer
    C[i,i]  $\leftarrow$  0
  fpara
  para diag  $\leftarrow$  1 hasta N-1 hacer
    para i  $\leftarrow$  1 hasta N-diag hacer
      C[i,i+diag]  $\leftarrow$  MinMultiple(C,i,i+diag)
    fpara
  fpara
ffun

```

Posibilidad de crear una versión que guarde las escalas

Coste $O(N^3)$
(2 bucles anidados más llamada)

diag	i	i+diag
1	1	2
1	2	3
1	3	4
1	4	5
1	5	6
2	1	3
2	2	4
2	3	5
2	4	6
3	1	4
3	2	5

de ir de un punto que esté

si $i = j$

fun MinMultiple(C: Tabla, i: entero, j: entero): entero

var

k, minimo: entero

fvar

minimo $\leftarrow \infty$

para k \leftarrow i+1 hasta j **hacer**

minimo \leftarrow min(minimo, C[k,j] + T[i,k])

fpara

dev minimo

ffun

Añadir T como 1er parámetro en declaración y llamada a MinMultiple

La mochila

- Se dispone de n objetos con un volumen entero v_i y un beneficio positivo b_i , y una mochila con una capacidad máxima V (o W)
- Objetivo: llenar la mochila de manera que se maximice el beneficio total considerando el volumen máximo:

$$\text{maximizar } \sum_{i=0}^n x_i b_i \text{ cumpliendo } \sum_{i=0}^n x_i v_i \leq V$$

$x_i = 0$ ó 1
NO
FRACTICIONABLES

- Ecuación de recurrencia:

$$\text{mochila}(i, W) = \begin{cases} 0 & \text{si } i = 0 \text{ y } W \geq 0 \\ -\infty & \text{si } W < 0 \\ \text{mochila}(i - 1, W) & \text{si } i > 0 \text{ y } v_i > W \\ \max\{\text{mochila}(i - 1, W), \\ \quad b_i + \text{mochila}(i - 1, W - v_i)\} & \text{si } i > 0 \text{ y } v_i \leq W \end{cases}$$

máximo
beneficio para
un volumen libre
 W considerando
los objetos entre
1 e i , siendo $i \leq n$

La mochila

- Construimos la tabla M (objetos x volumen)
 - Sólo es posible si los volúmenes son enteros
 - Para calcular $M[i, j]$ necesitamos 2 posiciones de la fila $i-1$
 - Si solo queremos el beneficio (no los objetos) basta con un vector
 - El valor $M[n, V]$ de la última fila nos da la solución

- Ejemplo:

Límite de volumen	0	1	2	3	4	5	6	7	8
posición 0	0	0	0	0	0	0	0	0	0
$v_1 = 1, b_1 = 2$	0	2	2	2	2	2	2	2	2
$v_2 = 3, b_2 = 5$	0	2	2	5	7	7	7	7	7
$v_3 = 4, b_3 = 10$	0	2	2	5	10	12	12	15	15
$v_4 = 5, b_4 = 14$	0	2	2	5	10	14	16	16	19
$v_5 = 7, b_5 = 15$	0	2	2	5	10	14	16	16	19



$$\text{mochila}(i-1, W) \quad v_i > W$$

$$\max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} \quad v_i \leq W$$

- Coste: $O(nV)$
- Posibilidad de recorrer la tabla y construir un vector binario indicando los objetos seleccionados

La mochila

- ¿Cómo saber los objetos incluidos?

- Empezamos en $t[n, W]$
- Si $t[i, j] == t[i-1, j]$: el objeto i **NO** se incluye y seguir en $t[i-1, j]$
- Si $t[i, j] != t[i-1, j]$: el objeto i **SI** se incluye y seguir en $t[i-1, j-v_i]$

Límite de volumen	0	1	2	3	4	5	6	7	8
posición 0	0	0	0	0	0	0	0	0	0
$v_1 = 1, b_1 = 2$	0	2	2	2	2	2	2	2	2
$v_2 = 3, b_2 = 5$	0	2	2	5	7	7	7	7	7
$v_3 = 4, b_3 = 10$	0	2	2	5	10	12	12	15	15
$v_4 = 5, b_4 = 14$	0	2	2	5	10	14	16	16	19
$v_5 = 7, b_5 = 15$	0	2	2	5	10	14	16	16	19

(, , ,) \Rightarrow (, , , 0) \Rightarrow (, , , 1, 0) \Rightarrow (, , 0, 1, 0) \Rightarrow (, 1, 0, 1, 0) \Rightarrow (0, 1, 0, 1, 0)

La mochila

```
tipo Tabla = matriz[0..n,0..V] de entero
tipo Vector = matriz[0..n] de entero
fun MochilaEntera(vol:Vector, ben:Vector, n: entero, V: entero, M:Tabla)
    var
        i,j: entero
    fvar
    para i ← 1 hasta n hacer
        M[i,0] ← 0
    fpara
    para j ← 1 hasta V hacer
        M[0,j] ← 0
    fpara
    para i ← 1 hasta n hacer
        para j ← 1 hasta V hacer
            si vol[i] > j entonces
                M[i,j] ← M[i-1,j]
            sino
                M[i,j] ← max(M[i-1,j], M[i-1,j-vol[i]] + ben[i])
            fsi
        fpara
    fpara
ffun
```

Ejercicio de examen

2.- Sea el problema de la mochila en su versión de objetos no fraccionables solucionado con programación dinámica. Supongamos que se dispone de 5 objetos con pesos: 1,3,4,5,7 y beneficios: 2,5,10,14,15 respectivamente, y un volumen máximo de 8. Identifica cuál de las siguientes respuestas correspondería al contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 5, si dichos objetos se consideran en orden creciente de pesos.

- a. 0 2 2 5 10 12 12 15 15
- b. 0 2 2 5 10 14 16 16 19
- c. 0 2 2 5 10 12 14 16 19
- d. Ninguna de las anteriores.

Ejercicio de examen

2.- Sea el problema de la mochila en su versión de objetos no fraccionables solucionado con programación dinámica. Supongamos que se dispone de 5 objetos con pesos: 1,3,4,5,7 y beneficios: 2,5,10,14,15 respectivamente, y un volumen máximo de 8. Identifica cuál de las siguientes respuestas correspondería al contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 5, si dichos objetos se consideran en orden creciente de pesos.

- a. 0 2 2 5 10 12 12 15 15
b. 0 2 2 5 10 14 16 16 19
c. 0 2 2 5 10 12 14 16 19
d. Ninguna de las anteriores.

Límite de volumen	0	1	2	3	4	5	6	7	8
posición 0	0	0	0	0	0	0	0	0	0
$v_1 = 1, b_1 = 2$	0	2	2	2	2	2	2	2	2
$v_2 = 3, b_2 = 5$	0	2	2	5	7	7	7	7	7
$v_3 = 4, b_3 = 10$	0	2	2	5	10	12	12	15	15
$v_4 = 5, b_4 = 14$	0	2	2	5	10	14	16	16	19
$v_5 = 7, b_5 = 15$	0	2	2	5	10	14	16	16	19

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.
- a) 0 1 6 7 7 18 19 24 25 25 28 29
 - b) 0 1 6 7 7 18 22 24 28 29 29 40
 - c) 0 1 6 7 7 18 22 28 29 34 35 40
 - d) Ninguna de las anteriores.

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
b) 0 1 6 7 7 18 22 24 28 29 29 40
c) 0 1 6 7 7 18 22 28 29 34 35 40
d) Ninguna de las anteriores.

$$\max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} \quad v_i \leq W$$

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
b) 0 1 6 7 7 18 22 24 28 29 29 40
c) 0 1 6 7 7 18 22 28 29 34 35 40
d) Ninguna de las anteriores.

$$\max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} \quad v_i \leq W$$

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
b) 0 1 6 7 7 18 22 24 28 29 29 40
c) 0 1 6 7 7 18 22 28 29 34 35 40
d) Ninguna de las anteriores.

$$\max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} \quad v_i \leq W$$

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
b) 0 1 6 7 7 18 22 24 28 29 29 40
c) 0 1 6 7 7 18 22 28 29 34 35 40
d) Ninguna de las anteriores.

$$\max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} \quad v_i \leq W$$

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
b) 0 1 6 7 7 18 22 24 28 29 29 40
c) 0 1 6 7 7 18 22 28 29 34 35 40
d) Ninguna de las anteriores.

$$\max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} \quad v_i \leq W$$

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll} \text{mochila}(i-1, W) & v_i > W \\ \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7							
6/22	0	1	6	7	7							
7/28	0	1	6	7	7							

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll}
 \text{mochila}(i-1, W) & v_i > W \\
 \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W
 \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18						
6/22	0	1	6	7	7							
7/28	0	1	6	7	7							

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

	$mochila(i - 1, W)$	$v_i > W$
	$\max\{mochila(i - 1, W), b_i + mochila(i - 1, W - v_i)\}$	$v_i \leq W$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19					
6/22	0	1	6	7	7							
7/28	0	1	6	7	7							

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

	$mochila(i - 1, W)$	$v_i > W$
	$\max\{mochila(i - 1, W), b_i + mochila(i - 1, W - v_i)\}$	$v_i \leq W$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24				
6/22	0	1	6	7	7							
7/28	0	1	6	7	7							

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll}
 \text{mochila}(i-1, W) & v_i > W \\
 \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W
 \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25			
6/22	0	1	6	7	7							
7/28	0	1	6	7	7							

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll} \text{mochila}(i-1, W) & v_i > W \\ \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18						
7/28	0	1	6	7	7	18						

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll}
 \text{mochila}(i-1, W) & v_i > W \\
 \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W
 \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22					
7/28	0	1	6	7	7	18						

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll}
 \text{mochila}(i-1, W) & v_i > W \\
 \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W
 \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24				
7/28	0	1	6	7	7	18						

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll}
 \text{mochila}(i-1, W) & v_i > W \\
 \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W
 \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28			
7/28	0	1	6	7	7	18						

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

	$mochila(i - 1, W)$	$v_i > W$
	$\max\{mochila(i - 1, W), b_i + mochila(i - 1, W - v_i)\}$	$v_i \leq W$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28	29		
7/28	0	1	6	7	7	18						

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll}
 \text{mochila}(i-1, W) & v_i > W \\
 \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W
 \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28	29	29	40
7/28	0	1	6	7	7	18						

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll}
 \text{mochila}(i-1, W) & v_i > W \\
 \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W
 \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28	29	29	40
7/28	0	1	6	7	7	18	22					

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll}
 \text{mochila}(i-1, W) & v_i > W \\
 \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W
 \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28	29	29	40
7/28	0	1	6	7	7	18	22	28				

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll} \text{mochila}(i-1, W) & v_i > W \\ \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28	29	29	40
7/28	0	1	6	7	7	18	22	28	29			

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll} \text{mochila}(i-1, W) & v_i > W \\ \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28	29	29	40
7/28	0	1	6	7	7	18	22	28	29	34		

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
- b) 0 1 6 7 7 18 22 24 28 29 29 40
- c) 0 1 6 7 7 18 22 28 29 34 35 40
- d) Ninguna de las anteriores.

$$\begin{array}{ll}
 \text{mochila}(i-1, W) & v_i > W \\
 \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W
 \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28	29	29	40
7/28	0	1	6	7	7	18	22	28	29	34	35	

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

-
- a) 0 1 6 7 7 18 19 24 25 25 28 29
 - b) 0 1 6 7 7 18 22 24 28 29 29 40
 - c) 0 1 6 7 7 18 22 28 29 34 35 40
 - d) Ninguna de las anteriores.

$$\begin{array}{ll} \text{mochila}(i-1, W) & v_i > W \\ \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W \end{array}$$

v/b	0	1	2	3	4	5	6	7	8	9	10	11
-	0	0	0	0	0	0	0	0	0	0	0	0
1/1	0	1	1	1	1	1	1	1	1	1	1	1
2/6	0	1	6	7	7	7	7	7	7	7	7	7
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28	29	29	40
7/28	0	1	6	7	7	18	22	28	29	34	35	40

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

- a) 0 1 6 7 7 18 19 24 25 25 28 29
b) 0 1 6 7 7 18 22 24 28 29 29 40
c) 0 1 6 7 7 18 22 28 29 34 35 40
d) Ninguna de las anteriores.

$$\begin{array}{ll} \text{mochila}(i-1, W) & v_i > W \\ \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W \end{array}$$

v/b	0	1	2	3	4	5	6	?	?	?	?	?
-	0	0	0	0	0	0	0	,	,	,	,	,
1/1	0	1	1	1	1	1	1	,	,	,	,	,
2/6	0	1	6	7	7	7	7	,	,	,	,	,
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28	29	29	40
7/28	0	1	6	7	7	18	22	28	29	34	35	40

¿Qué objetos hemos introducido?

Ejercicio de examen

3. Sea el problema de la mochila en su versión de objetos no fraccionables solucionado mediante programación dinámica. Suponga que se dispone de 5 objetos con volúmenes $\{1,2,5,6,7\}$ y que aportan unos beneficios de $\{1,6,18,22,28\}$, respectivamente. Suponga también que dispone de una mochila con una capacidad máxima de 11. Indique cuál sería el contenido de la tabla de resultados parciales en la fila correspondiente al objeto de peso 7, si dichos objetos se consideran en orden creciente de pesos.

-
- a) 0 1 6 7 7 18 19 24 25 25 28 29
 - b) 0 1 6 7 7 18 22 24 28 29 29 40
 - c) 0 1 6 7 7 18 22 28 29 34 35 40
 - d) Ninguna de las anteriores.

$$\begin{array}{ll}
 \text{mochila}(i-1, W) & v_i > W \\
 \max\{\text{mochila}(i-1, W), b_i + \text{mochila}(i-1, W - v_i)\} & v_i \leq W
 \end{array}$$

v/b	0	1	2	3	4	5	6					
-	0	0	0	0	0	0	0					
1/1	0	1	1	1	1	1	1					
2/6	0	1	6	7	7	7	7					
5/18	0	1	6	7	7	18	19	24	25	25	25	25
6/22	0	1	6	7	7	18	22	24	28	29	29	40
7/28	0	1	6	7	7	18	22	28	29	34	35	40

¿Qué objetos hemos introducido?

0	0	1	1	0
0	+ 0	+ 18	+ 22	+ 0

0 + 0 + 18 + 22 + 0

Multiplicación asociativa de matrices

- Multiplicar $A (m \times n)$ por $B (n \times p)$ da una matriz de $m \times p$ y requiere $m \times n \times p$ productos escalares

$$\begin{pmatrix} 2 & 1 \\ 0 & -3 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 & 2 \\ 4 & -2 \end{pmatrix} = \begin{pmatrix} 2 \cdot 3 + 1 \cdot 4 & 2 \cdot 2 + 1 \cdot (-2) \\ 0 \cdot 3 + (-3) \cdot 4 & 0 \cdot 2 + (-3) \cdot (-2) \\ 1 \cdot 3 + 2 \cdot 4 & 1 \cdot 2 + 2 \cdot (-2) \end{pmatrix} = \begin{pmatrix} 10 & 2 \\ -12 & 6 \\ 11 & -2 \end{pmatrix}$$

- Objetivo: hallar el número mínimo de multiplicaciones escalares para el producto de una secuencia de matrices
- Ejemplo:

Matriz	Dimensiones	Producto	Coste	Cálculo
A	60×2	$((AB)C)D$	18600	$60 \times 2 \times 30 + 60 \times 30 \times 5 + 60 \times 5 \times 20$
B	2×30	$A((BC)D)$	2900	$2 \times 30 \times 5 + 2 \times 5 \times 20 + 60 \times 2 \times 20$
C	30×5	$(AB)(CD)$	42600	$60 \times 2 \times 30 + 30 \times 5 \times 20 + 60 \times 30 \times 20$
D	5×20	$A(B(CD))$	6600	$30 \times 5 \times 20 + 2 \times 30 \times 20 + 2 \times 20 \times 60$
		$(A(BC))D$	6900	$2 \times 30 \times 5 + 60 \times 2 \times 5 + 60 \times 5 \times 20$

Multiplicación asociativa de matrices

- ¿Y si lo hacemos por fuerza bruta?
¿De cuantas formas podemos agrupar las matrices?
 - *Si hay 1 o 2 matrices...*
solo hay 1 agrupación posible: (A) o (AB)
 - *Si hay 3 matrices...*
hay 2: ((AB)C), (A(BC))
 - *Si hay 4 matrices...*
hay 5: (((AB)C)D)), ((AB)(CD)), ((A(BC))D), (A((BC)D)), (A(B(CD)))
 - *Si hay 5 matrices...*
hay... ¡14 agrupaciones posibles!
 - *Y con más... sigue creciendo exponencialmente!*
Siguen la secuencia de los números de Catalan

n	1	2	3	4	5	10	15
$T(n)$	1	1	2	5	14	4862	2674440

- No es una solución válida

Multiplicación asociativa de matrices

- Ecuación de recurrencia

$$E(i, j) = \begin{cases} \min_{i \leq k < j} \{E(i, k) + E(k + 1, j) + d_{i-1}d_kd_j\} & \text{si } i < j \\ 0 & \text{si } i = j \end{cases}$$

- d_i : Nº columnas de la matriz M_i (número de filas será d_{i-1})
- $E(i, j)$: Nº mínimo de productos escalares para multiplicar las matrices $M_i..M_j$

- Ejemplo: $M1(60x2) \times M2(2x30) \times M3(30x5) \times M4(5x20)$

	j=1	j=2	j=3	j=4
i=1	0			
i=2		0		
i=3			0	
i=4				0

	j=1	j=2	j=3	j=4
i=1	0	3600		
i=2		0	300	
i=3			0	3000
i=4				0

	j=1	j=2	j=3	j=4
i=1	0	3600	900	
i=2		0	300	500
i=3			0	3000
i=4				0

	j=1	j=2	j=3	j=4
i=1	0	3600	900	2900
i=2		0	300	500
i=3			0	3000
i=4				0

Multiplicación asociativa de matrices

$$M1(60 \times 2) \times M2(2 \times 30) \times M3(30 \times 5) \times M4(5 \times 20)$$

- Ecuación:

$$E(i, j) = \sum_{k=1}^N E(i, k) \cdot E(k, j)$$

	$j=1$	$j=2$	$j=3$	$j=4$
$i=1$	0	3600	900	?
$i=2$		0	300	500
$i=3$			0	3000
$i=4$				0

mínimo

$$+ 60 \times 2 \times 20 = 2900$$

$$+ 60 \times 30 \times 20 = 39000$$

$$+ 60 \times 5 \times 20 = 6900$$

- Ejemplo:

	$j=1$	$j=2$	$j=3$	$j=4$
$i=1$	0			
$i=2$	0			
$i=3$		0		
$i=4$		0		

	$j=1$	$j=2$	$j=3$	$j=4$
$i=1$	-	1	1	1
$i=2$		-	2	3
$i=3$			-	3
$i=4$				-

Posibilidad de guardar la posición de los paréntesis:
 Función recursiva que empieza en $[1, N]$ y se bifurca en $[i, k]$ y $[k+1, j]$ siendo k el valor almacenado

$$ABCD \rightarrow (A((BC)D))$$

```

tipo Tabla = matriz[1..N,1..N] de entero
tipo Vector = matriz[0..N] de entero
fun MultMatrices(d: Vector, N: entero, mult: Tabla)

```

var

i,diag: entero

fvar

para i \leftarrow 1 **hasta** N **hacer**

mult[i,i] \leftarrow 0

fpara

para diag \leftarrow 1 **hasta** N-1 **hacer**

para i \leftarrow 1 **hasta** N-diag **hacer**

mult[i,i+diag] \leftarrow MinMultiple(mult,d,i,i+diag)

fpara

fpara

ffun

Ejemplo:

	j=1	j=2	j=3	j=4
i=1	0			
i=2		0		
i=3			0	
i=4				0

va de

Coste O(N³)

(2 bucles anidados más llamada)

Coste espacial O(N²)
(tabla de NxN)

$$d_k d_j \} \quad \begin{array}{l} \text{si } i < j \\ \text{si } i = j \end{array}$$

de filas será d_{i-1})

para multiplicar

fun MinMultiple(mult: Tabla, d: Vector, i: entero, j: entero): entero

var

k, minimo: entero

fvar

minimo \leftarrow ∞

para k \leftarrow i **hasta** j-1 **hacer**

minimo \leftarrow min(minimo, mult[i,k] + mult[k+1,j] + d[i-1]*d[k]*d[j])

fpara

dev minimo

ffun

Ejercicio de examen

6. Sean $A_1..A_6$ matrices de dimensiones 30×35 , 35×15 , 15×5 , 5×10 , 10×20 y 20×25 ; y sea $m[i,j]$ el número mínimo de productos escalares para multiplicar $A_i \dots A_j$; es decir, el subrango $i..j$ con $i,j \in \{1..6\}$, siendo i la abscisa y j la ordenada, y siendo la matriz con los diferentes valores $m[i,j]$ la siguiente:

Antes vimos T
girada 90° a la
dcha.

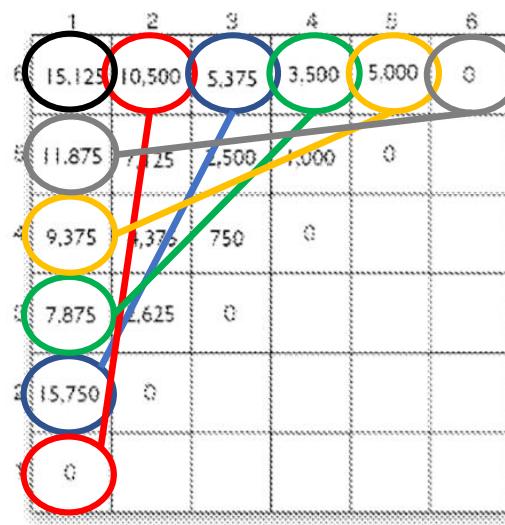
	1	2	3	4	5	6
6	15,125	10,500	5,375	3,500	5,000	0
5	11,875	7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

¿Cuál de las siguientes opciones es la parametrización óptima del producto de las seis matrices $A_1..A_6$?

- (a) $(A_1(A_2A_3)(A_4A_5)A_6)$
- (b) $(A_1A_2)(A_3A_4)(A_5A_6)$
- (c) $(A_1(A_2(A_3A_4)A_5)A_6)$
- (d) Ninguna de las anteriores

Ejercicio de examen

6. Sean $A_1..A_6$ matrices de dimensiones 30×35 , 35×15 , 15×5 , 5×10 , 10×20 y 20×25 ; y sea $m[i,j]$ el número mínimo de productos escalares para multiplicar $A_i \dots A_j$; es decir, el subrango $i..j$ con $i, j \in \{1..6\}$, siendo i la abscisa y j la ordenada, y siendo la matriz con los diferentes valores $m[i,j]$ la siguiente:



T'	1	2	3	4	5	6
6	3	?	?	?	?	-
5	?					-
4	?					-
3	?					-
2	?	-				
1	-					

¿Cuál de las siguientes opciones es la parametrización óptima del producto de las seis matrices $A_1..A_6$?

- (a) $(A_1(A_2A_3)(A_4A_5)A_6)$
- (b) $(A_1A_2)(A_3A_4)(A_5A_6)$
- (c) $(A_1(A_2(A_3A_4)A_5)A_6)$
- (d) Ninguna de las anteriores

- $T'[6,1] = \min(0+10500+30 \cdot 35 \cdot 25, 15750+5375+30 \cdot 15 \cdot 25, 7875+3500+30 \cdot 5 \cdot 25, 9375+5000+30 \cdot 10 \cdot 25, 11875+0+30 \cdot 20 \cdot 25)$
- $T'[6,1] = \min(36750, 32375, 15125, 21875, 26875)$

Ejercicio de examen

6. Sean $A_1..A_6$ matrices de dimensiones 30×35 , 35×15 , 15×5 , 5×10 , 10×20 y 20×25 ; y sea $m[i,j]$ el número mínimo de productos escalares para multiplicar $A_i \dots A_j$; es decir, el subrango $i..j$ con $i, j \in \{1..6\}$, siendo i la abscisa y j la ordenada, y siendo la matriz con los diferentes valores $m[i,j]$ la siguiente:

	1	2	3	4	5	6
6	15,125	10,500	5,375	3,500	5,000	0
5	11,875	7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0	0		
2	15,750	0				
1	0					

T'	1	2	3	4	5	6
6	3			5	?	-
5				?	-	
4					-	
3	1	?		-		
2	?	-				
1	-					

¿Cuál de las siguientes opciones es la parametrización óptima del producto de las seis matrices $A_1..A_6$?

- (a) $(A_1(A_2A_3)(A_4A_5)A_6)$
- (b) $(A_1A_2)(A_3A_4)(A_5A_6)$
- (c) $(A_1(A_2(A_3A_4)A_5)A_6)$
- (d) Ninguna de las anteriores

- $T'[3,1] = \min(0+2625+30 \cdot 35 \cdot 5, 15750+0+30 \cdot 15 \cdot 5)$
- $T'[3,1] = \min(7875, 18000)$
- $T'[6,4] = \min(0+5000+5 \cdot 10 \cdot 25, 1000+0+5 \cdot 20 \cdot 25)$
- $T'[6,4] = \min(6250, 3500)$

Ejercicio de examen

6. Sean $A_1..A_6$ matrices de dimensiones 30×35 , 35×15 , 15×5 , 5×10 , 10×20 y 20×25 ; y sea $m[i,j]$ el número mínimo de productos escalares para multiplicar $A_i \dots A_j$; es decir, el subrango $i..j$ con $i,j \in \{1..6\}$, siendo i la abscisa y j la ordenada, y siendo la matriz con los diferentes valores $m[i,j]$ la siguiente:

	1	2	3	4	5	6
6	15,125	10,500	5,375	3,500	5,000	0
5	11,875	7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

T'	1	2	3	4	5	6
6	3			5		-
5						-
4					-	
3	1			-		
2		-				
1	-					

¿Cuál de las siguientes opciones es la parametrización óptima del producto de las seis matrices $A_1..A_6$?

- (a) $(A_1(A_2A_3)(A_4A_5)A_6)$
(b) $(A_1A_2)(A_3A_4)(A_5A_6)$
(c) $(A_1(A_2(A_3A_4)A_5)A_6)$
(d) Ninguna de las anteriores

$(A_1A_2A_3A_4A_5A_6) \rightarrow$
 $((A_1A_2A_3)(A_4A_5A_6)) \rightarrow$
 $((A_1(A_2A_3))((A_4A_5)A_6))$

Camino de coste mínimo en un grafo dirigido

- Algoritmo de Floyd como alternativa al algoritmo de Dijkstra (ver alg. voraz), los dos con $O(N^3)$
- Ecuación de recurrencia
 - $M(i, j, k)$: Coste mínimo entre los nodos i y j pudiendo pasar por los nodos entre 1 y k
 - $A[i, j]$: el coste de la arista entre i y j

$$M(i, j, k) = \begin{cases} 0 & \text{si } k = 0 \text{ y } i = j \\ A[i, j] & \text{si } k = 0 \text{ y } i \neq j \\ \min(M(i, j, k-1), \\ \quad M(i, k, k-1) + M(k, j, k-1)) & \text{si } k > 0 \end{cases}$$

- Para calcular $M(i, j, k)$ sólo necesitamos los datos tras incluir el nodo $k-1$, así que utilizamos una tabla NxN que se va reescribiendo

Camino grafo

- Algoritmo de Dijkstra
- Ecuaciones

```
tipo Tabla = matriz[1..N,1..N] de entero
fun Floyd(A: Tabla, N: entero, M: Tabla)
    var
        i, j, k: entero
    fvar
    para i ← 1 hasta N hacer
        para j ← 1 hasta N hacer
            M[i,j] ← A[i,j]
        fpara
    fpara
    para k ← 1 hasta N hacer
        para i ← 1 hasta N hacer
            para j ← 1 hasta N hacer
                M[i,j] ← min(M[i,j], M[i,k] + M[k,j])
            fpara
        fpara
    ffun
```

Distancia de edición

- Objetivo: encontrar el número mínimo de cambios para transformar la cadena $X=x_1, \dots, x_n$ en la cadena $Y=y_1, \dots, y_m$ con los siguientes cambios posibles:
 - Borrar un carácter de X
 - Insertar uno de los caracteres de Y en X
 - Sustituir un carácter de X por uno de los de Y
- Ecuación de recurrencia:
 - $C(i, j)$: Número mínimo de cambios para convertir x_1, \dots, x_i en y_1, \dots, y_j

$$C(i, j) = \begin{cases} i & \text{si } j = 0 \\ j & \text{si } i = 0 \\ 1 + \min\{C(i-1, j), C(i, j-1), C(i-1, j-1)\} & \text{si } i \neq 0, j \neq 0, x_i \neq y_j \\ \min\{C(i-1, j) + 1, C(i, j-1) + 1, C(i-1, j-1)\} & \text{si } i \neq 0, j \neq 0, x_i = y_j \end{cases}$$

Distancia de edición

- Objetivo: encontrar el número mínimo de cambios para transformar la cadena $X=x_1, \dots, x_n$ en la cadena $Y=y_1, \dots, y_m$ con los siguientes cambios posibles:
 - Borrar un carácter de X
 - Insertar uno de los caracteres de Y en X
 - Sustituir un carácter de X por uno de los de Y
- Ecuación de recurrencia:
 - $C(i, j)$: Nº mínimo de cambios para convertir x_1, \dots, x_i en y_1, \dots, y_j

$$C(i, j) = \begin{cases} i & \text{borrado} \\ j & \text{inserción} \\ \min\{C(i-1, j), C(i, j-1), C(i-1, j-1)\} & \text{sustitución} \end{cases} \quad \begin{array}{ll} \text{si } j = 0 & \\ \text{si } i = 0 & \\ \text{si } i \neq 0, j \neq 0, x_i \neq y_j & \\ \text{si } i \neq 0, j \neq 0, x_i = y_j & \end{array}$$

```

tipo Tabla = matriz[0..n,0..m] de entero
fun DistanciaEdicion(X: Vector[1..n] de caracter, Y: Vector[1..m] de caracter,
                      n,m: entero, C: Tabla)
    var
        i,j,tmp: entero
    fvar
        para i  $\leftarrow$  0 hasta n hacer
            C[i,0]  $\leftarrow$  i
        fpara
        para j  $\leftarrow$  0 hasta m hacer
            C[0,j]  $\leftarrow$  j
        fpara
        para i  $\leftarrow$  1 hasta n hacer
            para j  $\leftarrow$  1 hasta m hacer
                tmp  $\leftarrow$  min(1 + C[i-1,j], 1 + C[i,j-1])
                si X[i] = Y[j] entonces
                    C[i,j]  $\leftarrow$  min(tmp, C[i-1,j-1])
                sino
                    C[i,j]  $\leftarrow$  min(tmp, C[i-1,j-1]+1)
                fsi
            fpara
        fpara
ffun

```

Coste O(nm)
 (2 bucles anidados)

Coste espacial O(nm)
 (tabla de n x m)

Ejercicio de examen

6. Indique cuál de las siguientes afirmaciones es **falsa** con respecto al esquema de programación dinámica:
- (a) El objetivo de este esquema es la reducción del coste del algoritmo mediante la memorización de soluciones parciales que se necesitan para llegar a la solución final.
 - (b) Es igual de eficiente que el esquema divide y vencerás cuando en este último esquema se dan llamadas recursivas que se repiten en la secuencia de llamadas recursivas.
 - (c) El problema de la multiplicación asociativa de matrices resuelto con programación dinámica tiene un coste temporal de $O(N^3)$, siendo N el número de matrices para multiplicar.
 - (d) El problema de la distancia de edición entre dos cadenas resuelto con programación dinámica tiene un coste temporal de $O(nm)$, siendo n la longitud de una cadena y m la longitud de la otra.

Ejercicio de examen

6. Indique cuál de las siguientes afirmaciones es **falsa** con respecto al esquema de programación dinámica:
- (a) El objetivo de este esquema es la reducción del coste del algoritmo mediante la memorización de soluciones parciales que se necesitan para llegar a la solución final.
 -  (b) Es igual de eficiente que el esquema divide y vencerás cuando en este último esquema se dan llamadas recursivas que se repiten en la secuencia de llamadas recursivas.
 - (c) El problema de la multiplicación asociativa de matrices resuelto con programación dinámica tiene un coste temporal de $O(N^3)$, siendo N el número de matrices para multiplicar.
 - (d) El problema de la distancia de edición entre dos cadenas resuelto con programación dinámica tiene un coste temporal de $O(nm)$, siendo n la longitud de una cadena y m la longitud de la otra.

Resumen de ejemplos

- Fibonacci
 - Almacenar los 2 últimos valores
- Coeficientes binomiales
 - Triángulo de Pascal
- Devolución del cambio (para cualquier tipo de monedas)
 - $T[i,j] = \text{nº monedas de tipo } x_i \text{ o menos para la cantidad } C_j$
- Viaje por el río
 - Ir almacenando el coste en saltos de distancia 1, 2, ...
- Mochila (objetos NO fraccionables)
 - $T[i,j] = \text{beneficio máximo con objetos } 1..i \text{ y volumen libre } j$
- Multiplicación asociativa de matrices
 - $T(i, j) = \text{Nº mínimo de productos escalares para las matrices } M_i..M_j$
- Camino de coste mínimo en un grafo dirigido
 - Alg. de Floyd (alternativa a Dijkstra): consideramos cada vez más nodos
- Distancia de edición
 - $T(i, j) = \text{Nº mínimo de cambios para las posiciones } i \text{ y } j \text{ de las cadenas } x, y$