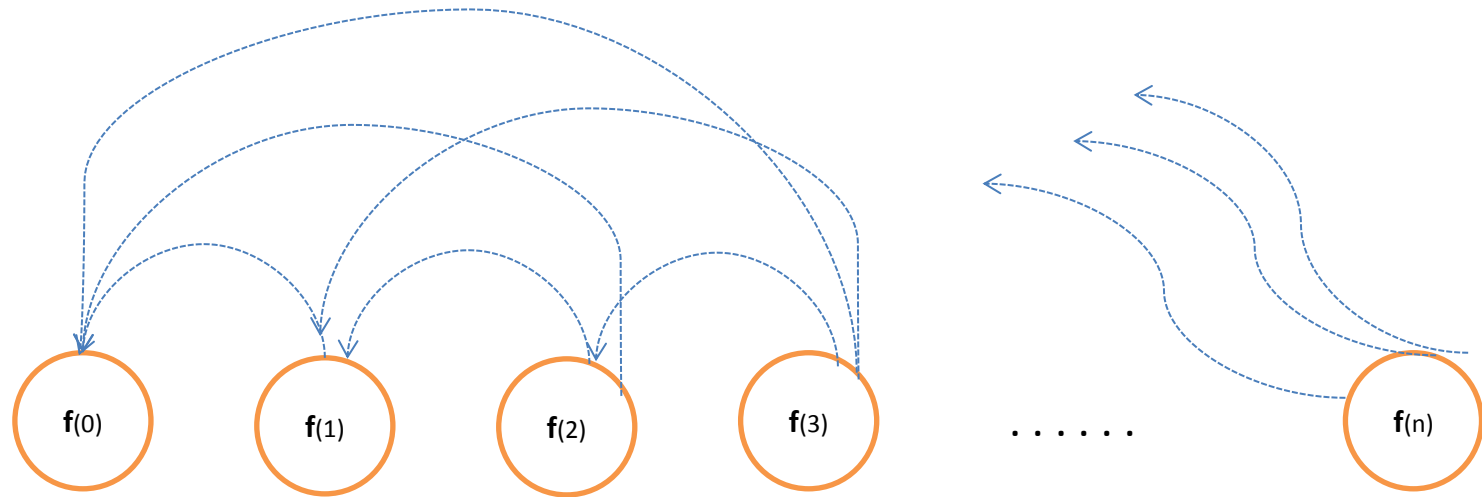


Prácticas Tema 3:

Programación Dinámica



1. Programación Dinámica

- La programación recursiva comienza desde el último nivel hasta el caso base: en ocasiones se repiten llamadas a un mismo nivel, y se repite la computación.
- La programación dinámica parte del caso base, y resuelve hacia delante:
 - Resuelve los siguientes niveles en función de los casos anteriores.
 - Almacena en una tabla la resolución de un nivel para no repetir cálculos.

1. Programación Dinámica

- Para resolver un problema, éste debe plantearse:
 - En forma de grafo, donde un estado supone que se ha pasado por los anteriores estados.
 - La función objetivo es la evaluación de un estado, que depende de la función resuelta de uno o más estados anteriores.
 - El camino de un nodo a otro debe ser óptimo: minimizar o maximizar la función objetivo.

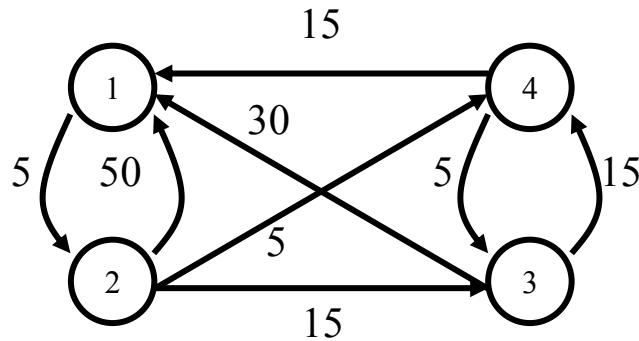
1. Programación Dinámica

- Los problemas que veremos pueden englobarse en 3 tipos:
 - Camino más corto entre 2 estados
 - Floyd
 - Traducción de un idioma a otro en cadena.
 - Insertar sí o no, una vez cada elemento
 - Llenado de Mochila, CD y Pendrive
 - Insertar sí o no, varias veces cada elemento
 - Devolver el cambio
 - Insertar sí o no, una fracción de un elemento
 - Inversión óptima
 - Abono de campos

Problemas: 'Camino más corto entre 2 estados'

- El algoritmo de Floyd dice que si el Camino a $A \rightarrow B$ es más largo que $A \rightarrow C + C \rightarrow B$, entonces elijamos $A \rightarrow C \rightarrow B$.
- Para resolver un problema por Floyd para un grafo con n nodos, necesitamos :
 - Matriz $L[n][n]$ con el peso de las aristas entre cada par de nodos.
 - Matriz $D[n][n]$
 - Comienza siendo $D=L$
 - Evoluciona n veces calculando:
 - $D[i,j]=\min\{D[i,j], D[i,k]+D[k,j]\}$
 - Donde k es el id del nodo intermedio, $k=1\dots n$

Camino más corto entre 2 nodos



- $D_0 = L =$

| n/n | 1 | 2 | 3 | 4 |
|-----|----|----------|----------|----------|
| 1 | 0 | 5 | ∞ | ∞ |
| 2 | 50 | 0 | 15 | 5 |
| 3 | 30 | ∞ | 0 | 15 |
| 4 | 15 | ∞ | 5 | 0 |

- $D_{k=1} =$

| n/n | 1 | 2 | 3 | 4 |
|-----|----|-----------|----------|----------|
| 1 | 0 | 5 | ∞ | ∞ |
| 2 | 50 | 0 | 15 | 5 |
| 3 | 30 | 35 | 0 | 15 |
| 4 | 15 | 20 | 5 | 0 |

Ejemplo: $D[1,2] = \min \{D[1,2], D[1,1]+D[1,2] = \min\{5, 0+5\}=5\}$

Ejemplo: $D[3,2] = \min \{D[3,2], D[3,1]+D[1,2] = \min\{\infty, 30+5\}=35\}$

(Truco para ir más rápido si se hace a mano: si $k=1$, la fila y columna 1 no hace falta comprobar si cambian.)

- $D_{k=2} =$

| n/n | 1 | 2 | 3 | 4 |
|-----|----|----|-----------|-----------|
| 1 | 0 | 5 | 20 | 10 |
| 2 | 50 | 0 | 15 | 5 |
| 3 | 30 | 35 | 0 | 15 |
| 4 | 15 | 20 | 5 | 0 |

Ejemplo: $D[1,3] = \min \{D[1,3], D[1,2]+D[2,3] = \min\{\infty, 5+15\}=20\}$

(En este caso no hace falta comprobar la fila y columna 2 porque $k=2$...)

- $D_{k=3} =$

| n/n | 1 | 2 | 3 | 4 |
|-----|-----------|----|----|----|
| 1 | 0 | 5 | 20 | 10 |
| 2 | 45 | 0 | 15 | 5 |
| 3 | 30 | 35 | 0 | 15 |
| 4 | 15 | 20 | 5 | 0 |

- $D_{k=4} =$

| n/n | 1 | 2 | 3 | 4 |
|-----|-----------|----|-----------|----|
| 1 | 0 | 5 | 15 | 10 |
| 2 | 20 | 0 | 10 | 5 |
| 3 | 30 | 35 | 0 | 15 |
| 4 | 15 | 20 | 5 | 0 |

Recuperar el camino más corto

- Con lo anterior solo se calcula coste más corto entre 2 nodos.
- Para recuperar el camino entre 2 nodos, hay que mantener en el proceso una matriz $P[n,n]$:
 - Se inicia a 0.
 - Si en la iteración k , el camino más corto entre los nodos i,j es a través del nodo k , entonces $P[i,j]=k$.

Recuperar el camino más corto

• $P_0 =$

| n/ n | 1 | 2 | 3 | 4 |
|---------|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 |

• $P_1 =$

| n/ n | 1 | 2 | 3 | 4 |
|---------|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 |

• $P_2 =$

| n/ n | 1 | 2 | 3 | 4 |
|---------|---|---|---|---|
| 1 | 0 | 0 | 2 | 2 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 |

• $P_3 =$

| n/ n | 1 | 2 | 3 | 4 |
|---------|---|---|---|---|
| 1 | 0 | 0 | 2 | 2 |
| 2 | 3 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 |

• $P_4 =$

| n/ n | 1 | 2 | 3 | 4 |
|---------|---|---|---|---|
| 1 | 0 | 0 | 4 | 2 |
| 2 | 4 | 0 | 4 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 |

¿Camino más corto entre 1 y 3?:

- $P[1,3]=4 \rightarrow$ el camino más corto de 1 a 3 pasa por 4

- $P[1,4]=2 \rightarrow$ el camino más corto de 1 a 4 pasa por 2

- $P[1,2]=0 \rightarrow$ el camino más corto de 1 a 2 es directo

$1 \rightarrow 2 \rightarrow 4 \rightarrow 3$ con coste $5+5+5=15$. (comprobar $D[1,3]=15$)

Minimizar coste de traducción

- En el departamento de una empresa de traducciones se puede hacer traducciones de textos entre varios idiomas.
- Se pueden alquilar diccionarios a precios distintos dependiendo del par ordenado de idiomas que traduce.
- Se pide proporcionar un algoritmo que calcule el **coste** más barato en cuanto a alquiler de diccionarios, para realizar cada traducción posible.

Ej: Minimizar coste de traducción

- Puede verse como buscar el camino más barato (corto) entre 2 idiomas (nodos).
- → **aplicación directa de Floyd.**
- **Diseño de la solución:**
 - Matriz $N \times N$ (N idiomas)
 - Al inicializar:
 - $M[i,j] = 0$, si $i=j$
 - $M[i,j] = \infty$, si no hay enlace directo
 - $M[i,j] = \text{coste de traducir } i \rightarrow j$, e.o.c.
 - Iteraciones para $k=1, \dots, N$, haciendo en cada iteración:
 - $M[i,j] = \min \{M[i,j], M[i,k] + M[k,j]\}$
 - Si se pidiera devolver el conjunto de idiomas, utilizar el método de recuperar el camino más corto con la matriz P.

Problemas '*insertar sí o no*'

¿Avanzar un estado/insertar 1 objeto aporta valor a mi función de evaluación?

Ejercicios I

• Problema de la mochila sin fraccionar

- Meter N objetos en una mochila con capacidad W
- Tenemos
 - Vector $B[N]$ que indica el Beneficio de meter cada objeto
 - Vector $P[N]$ que indica el Peso de cada objeto
- Diseñar $V[N+1, W+1]$, donde $V[i, j]$ indique el beneficio máximo de **meter o intentar meter** i objetos en una mochila con capacidad j .
 - $V[i, j] = 0$, si $i=0$ ó $j=0$
 - $V[i, j] = \max\{V[i-1, j], B[i] + V[i-1, j-P[i]]\} = \max\{\text{no meter objeto } i, \text{ sí meter objeto } i\}$
 - $V[i, j] = -\infty$, para $j < 0$

| N/W | 0 | 1 | 2 | 3 | ... | W |
|-----|---|---|---|---|-----|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | | | | | |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| ... | 0 | | | | | |
| N | 0 | | | | | |

$$V[1,1] = \max\{V[0,1], B[1] + V[0,1-P[1]]\} = \max\{0, B[1]\} = B[1]$$



- Para recuperar, además del beneficio, los objetos seleccionados:
 - Comenzar en $V[N+1, W+1]$
 - Hacer hasta $j=i=0$:
 - Si $V[i, j] = V[i-1, j]$ entonces ir a $V[i-1, j]$
 - Si $V[i, j] = V[i-1, j-P[i]] + B[i]$:
 - coger objeto i
 - Ir a $V[i-1, j-P[i]]$

Ejercicios I

- Problema del llenado de CD sobrando menor espacio posible
- Tenemos:
 - Disco de capacidad C a llenar.
 - Vector de archivos L[N] con el tamaño de cada archivo.
- Diseño de la solución:
 - El espacio ocupado también es el beneficio obtenido
 - Matriz M[N+1,C+1], donde M[i,j] indica el tamaño ocupado (\approx beneficio), considerando hasta el fichero i, al intentar ocupar la cantidad j
 - $M[i,j] = 0$, si $i=0$ ó $j=0$
 - $M[i,j] = \mathbf{max}\{M[i-1,j], L[i]+M[i-1,j-L[i]]\} = \max\{\text{no grabar } i, \text{ sí grabar } i\}$
 - $M[i,j] = -\infty$ para $j < 0$

- Hacer para tamaños {100,200,300} MB, Capacidad=500

| N/C | 0 | 1 | 2 | 3 | 4 | 5 |
|-----|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | | | | | |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |

$$M[1,1] = \max\{M[0,1], L[1] + M[0,0]\} = \max\{0, 1+0\} = 1 \quad \dots M[1,j] = 1$$

$$M[2,1] = \max\{M[1,1], L[2] + M[1,-1]\} = \max\{1, -\infty\} = 1 \quad M[2,2] = \max\{M[1,2], L[2] + M[1,0]\} = \max\{1, 2+0\} = 2$$

$$M[2,3] = \max\{M[1,3], L[2] + M[1,1]\} = \max\{1, 2+1\} = 3 \quad M[2,4] = \max\{M[1,4], L[2] + M[1,2]\} = \max\{1, 2+1\} = 3$$

$$M[2,5] = \max\{M[1,5], L[2] + M[1,3]\} = \max\{1, 2+1\} = 3$$

$$M[3,1] = \max\{M[2,1], L[3] + M[2,-2]\} = \max\{1, 3-\infty\} = 1 \quad M[3,2] = \max\{M[2,2], L[3] + M[2,-1]\} = \max\{2, 3-\infty\} = 2$$

$$M[3,3] = \max\{M[2,3], L[3] + M[2,0]\} = \max\{3, 3+0\} = 3 \quad M[3,4] = \max\{M[2,4], L[3] + M[2,1]\} = \max\{3, 3+1\} = 4$$

$$M[3,5] = \max\{M[2,5], L[3] + M[2,2]\} = \max\{3, 3+2\} = 5$$

| N/C | 0 | 1 | 2 | 3 | 4 | 5 |
|-----|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 2 | 3 | 3 | 3 |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 |

Recuperar los archivos insertados igual que en la mochila:

-Si $M[i,j]=M[i-1,j]$ entonces ir a $M[i-1,j]$

-Si $M[i,j]=M[i-1, j-L[i]] + L[i]$:

-coger objeto i

-Ir a $M[i-1, j-L[i]]$

- $M[3,5] = M[2,5-3] + 3 \rightarrow$ cojo archivo 3

- $M[2,2] = M[1,2-2] + 2 \rightarrow$ cojo archivo 2

- $M[1,0] = M[1-1,0] \rightarrow$ ir a $M[1-1,0]$

- $M[0,0] \rightarrow i=j=0$ criterio de parada

\rightarrow Se han seleccionado los archivos 3 y 2

Ejercicios I

- Problema del llenado de CD con menor nº de ficheros
- Para el caso de $L=\{1,2,3\}$ y $C=4$,
 - El algoritmo anterior llenaría los 4 (ver $M[3,4]$ en el ej. anterior) con 1+3
 - Pero ahora se pide utilizar el menor posible, así que habría que devolver el fichero de tamaño 3, y sobraría 1 unidad de capacidad.
- Diseño de la solución:
 - Se ordenan de mayor a menor tamaño $L=\{3,2,1\}$
 - Matriz $M[N+1,C+1]$, donde $M[i,j]$ indica el tamaño ocupado (\approx beneficio), considerando hasta el fichero i , al intentar ocupar la cantidad j
 - $M[i,j] = 0$, si $j=0$ ó $j<i$
 - $M[i,j] = -\infty$ para $j<0$
 - $M[1,j] = 1$, si $j \geq L[1]$
 - $M[i,j] = \max\{M[i-1,j], 1+M[i-1,j-L[i]]\} = \max\{\text{no grabar } i, \text{ sí grabar } i\}$

| N/C | 0 | 1 | 2 | 3 | 4 |
|-----|---|---|---|---|---|
| 3 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | | | |
| 1 | 0 | | | | |

$$M[2,2] = \max\{M[1,2], 1 + M[1,2-2]\} = \max\{0, 1+0\} = 1$$

$$M[2,3] = \max\{M[1,3], 1 + M[1,3-2]\} = \max\{1, 1+0\} = 1$$

$$M[2,4] = \max\{M[1,4], 1 + M[1,4-2]\} = \max\{1, 1+0\} = 1$$

... (todas las siguientes también dan 1...)

Hacerlo para capacidad hasta 7 si se quiere ver mejor que en efecto así se fuerza

Un comportamiento voraz que no es óptimo.

Problemas ‘insertar sí o no varias veces’

Ejercicios I – Devolver el cambio con N tipos de monedas

- Tenemos:
 - $D[N]$ vector con valor de cada moneda
 - C cantidad a pagar
- Diseño de la solución:
 - Matriz $M[N][C+1]$: $M[i,j]$ es la suma del número de monedas necesarias para pagar la cantidad j , con monedas de tipo 1 hasta i .
 - Iniciar $M[i,0]=0$, para todo i .
 - $M[i,j] = \min \{M[i-1,j], 1 + M[i, j - D[i]]\} = \min\{\text{sin usar monedas de tipo } i, \text{ usando al menos una moneda de tipo } i\}$

- Devolver el cambio con N tipos de monedas
- $C=8$; $N=\{1,4,6\}$

| N/C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|---|---|---|---|---|---|---|---|---|
| D[1]=1 | 0 | | | | | | | | |
| D[2]=4 | 0 | | | | | | | | |
| D[3]=6 | 0 | | | | | | | | |

$$M[1,1]=\min\{M[0,1], 1 + M[1,0]\}=\min\{\infty, 1+0\}=1$$

$$M[1,2]=\min\{M[0,2], 1 + M[1,1]\}=\min\{\infty, 1+1\}=2$$

$$\dots M[1,j]=j$$

$$M[2,1]=\min\{M[1,1], 1 + M[2,-3]\}=\min\{1, 1+\text{no existe}\}=1$$

$$M[2,2]=\min\{M[1,2], 1 + M[2,-2]\}=\min\{2, 1+\text{no existe}\}=2$$

$$M[2,3]=\min\{M[1,3], 1 + M[2,-1]\}=\min\{3, 1+\text{no existe}\}=3$$

$$M[2,4]=\min\{M[1,4], 1 + M[2,0]\}=\min\{4, 1+0\}=1$$

$$M[2,5]=\min\{M[1,5], 1 + M[2,1]\}=\min\{5, 1+1\}=2$$

$$M[2,6]=\min\{M[1,6], 1 + M[2,2]\}=\min\{6, 1+2\}=3$$

$$M[2,7]=\min\{M[1,7], 1 + M[2,3]\}=\min\{7, 1+3\}=4$$

$$M[2,8]=\min\{M[1,8], 1 + M[2,4]\}=\min\{8, 1+1\}=2$$

$$M[3,1]=\min\{M[2,1], 1 + M[3,-5]\}=\min\{1, 1+\text{no existe}\}=1$$

....

| N/C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|---|---|---|---|---|---|---|---|---|
| D[1]=1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| D[2]=4 | 0 | 1 | 2 | 3 | 1 | 2 | 3 | 4 | 2 |
| D[3]=6 | 0 | 1 | 2 | 3 | 1 | 2 | 1 | 2 | 2 |

Para pagar $C=8$ con monedas de tipo 1, 4 y 6: hacen falta 2 monedas

| N/C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|---|---|---|---|---|---|---|---|---|
| D[1]=1 | 0 | 1 | 2 | 4 | 5 | 6 | 7 | 8 | 8 |
| D[2]=4 | 0 | 1 | 2 | 3 | 1 | 2 | 3 | 4 | 2 |
| D[3]=6 | 0 | 1 | 2 | 3 | 1 | 2 | 1 | 2 | 2 |

¿Y si nos preguntan, además de cuántas, qué monedas son?

- Desde $M[N, C+1]$, hasta $M[_, 0]$:
 - Si $M[i, j] = M[i-1, j]$:
 - no hacen falta monedas de tipo i
 - seguir por $M[i-1, j]$
 - Si $M[i, j] = 1 + M[i, j-D[i]]$:
 - 1 moneda i
 - Seguir por $M[i, j-D[i]]$
 - Si $M[i, j] = M[i-1, j] = 1 + M[i, j-D[i]] = M[i, j-D[i]]$,
 - Coger moneda i ó $i-1$

Para pagar $C=8$:

- $M[3, 8] = M[2, 8] \rightarrow$ no hacen falta monedas de tipo 3, y sigo mirando por $M[2, 8]$
- $M[2, 8] = 1 + M[2, 8-4] = 1 + M[2, 4] \rightarrow$ 1 moneda de tipo 2, y sigo mirando por $M[2, 8-4]$
- $M[2, 4] = 1 + M[2, 4-4] \rightarrow$ 1 moneda de tipo 2, y sigo mirando por $M[2, 4-4]$
- $M[2, 0] \rightarrow j=0$, criterio de parada.
- \rightarrow 2 monedas de tipo 2 (2 monedas de valor $D[2]=4$).

Problemas '*insertar fracción de una cantidad o elemento*'

Ejercicios II – Inversión Óptima en B=3 bancos, disponiendo de cantidades de dinero desde 0 hasta D=4

- Tenemos:
 - Matriz $BC[B][D+1]$, donde $BC[i,j]$ es el beneficio obtenido al invertir una cantidad j en el banco i .
 - Encontrar el reparto óptimo del dinero
- Diseño de la solución:
 - Iniciar Matriz $M[i,j]=0$ si $i=0$ ó $j=0$
 - Matriz $M[B+1][D+1]$, donde $M[i,j]$ tomará el valor que maximice $BC[i,k] + M[i-1, j-k]$; para $k=0\dots j$.
 - Por lo anterior, se puede iniciar $M[1,j] = BC[1,j]$ para todo j

BC=

| Cantidad Banco | 0 | 1 | 2 | 3 | 4 |
|----------------|---|---|---|---|---|
| {} | 0 | 0 | 0 | 0 | 0 |
| {B1} | 0 | 2 | 5 | 6 | 7 |
| {B2} | 0 | 1 | 3 | 6 | 7 |
| {B3} | 0 | 1 | 4 | 5 | 8 |

M=

| Cantidad Banco | 0 | 1 | 2 | 3 | 4 |
|----------------|---|---|---|---|---|
| {} | 0 | 0 | 0 | 0 | 0 |
| {B1} | 0 | 2 | 5 | 6 | 7 |
| {B1,B2} | 0 | | | | |
| {B1,B2,B3} | 0 | | | | |

$$M[2,4] = \max \begin{cases} k=0 & BC[2,0] + M[1,4] = 0+7 \\ k=1 & BC[2,1] + M[1,3] = 1+6 \\ k=2 & BC[2,2] + M[1,2] = 3+5 \\ k=3 & BC[2,3] + M[1,1] = 6+2 \\ k=4 & BC[2,4] + M[1,0] = 7+0 \end{cases} = 8$$

$$M[3,4] = \max \begin{cases} k=0 & BC[3,0] + M[2,4] = 0+7 \\ k=1 & BC[3,1] + M[2,3] = 1+6 \\ k=2 & BC[3,2] + M[2,2] = 4+5 \\ k=3 & BC[3,3] + M[2,1] = 5+2 \\ k=4 & BC[3,4] + M[2,0] = 8+0 \end{cases} = 9$$

... el resto de $M[i,j]$ da igual que $M[1,j]$

→ Máximo beneficio que se puede obtener al utilizar 4 millones repartidos en los 3 bancos: 9

→ Si nos fijamos, sería el resultado de invertir 2 millones en B1 y otros 2 millones en B3

- Ejercicio Examen 5 Junio 2011 y 6 Julio 2011: repartir 6 sacos de abono en 4 campos, teniendo la tabla CS de beneficio obtenido en cada campo con 0 a 4 sacos de abono.

CS=

| Sacos | 0 | 1 | 2 | 3 | 4 |
|--------|---|---|---|---|---|
| Campos | | | | | |
| {} | 0 | 0 | 0 | 0 | 0 |
| {C1} | 0 | 4 | 3 | 5 | 7 |
| {C2} | 0 | 5 | 3 | 2 | 3 |
| {C3} | 0 | 3 | 4 | 3 | 4 |
| {C4} | 0 | 4 | 3 | 4 | 6 |

M=

| Sacos | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---------------|---|---|---|---|---|---|---|
| Campos | | | | | | | |
| {} | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| {C1} | 0 | 4 | 3 | 5 | 7 | 7 | 7 |
| {C1,C2} | 0 | | | | | | |
| {C1,C2,C3} | 0 | | | | | | |
| {C1,C2,C3,C4} | 0 | | | | | | |

Como no tenemos información para 5 y 6 sacos, usamos la información de 4.

Diseño de la solución:

- Iniciar Matriz $M[i,j]=0$ si $i=0$ ó $j=0$
- Matriz $M[C+1][S+1]$, donde $M[i,j]$ tomará el valor que maximice $CS[i,k] + M[i-1, j-k]$; para $k=0 \dots \min\{j,4\}$.
- Por lo anterior, se puede iniciar $M[1,j] = CS[1,\min\{j,4\}]$ para todo j