Mochila01

Ficheros requeridos: Mochila01.java (Descargar)
Tipo de trabajo: Individual

Se dispone de n objetos, cada uno con un peso (wi) y un beneficio (vi). También se dispone de una mochila en la que se pueden meter dichos objetos, con una capacidad de peso máximo M. Sin perdida de generalidad se supondrá que todos los valores son > 0.

El objetivo consiste en maximizar el valor de los objetos transportados y respetando la limitación de capacidad máxima M

TENDREMOS

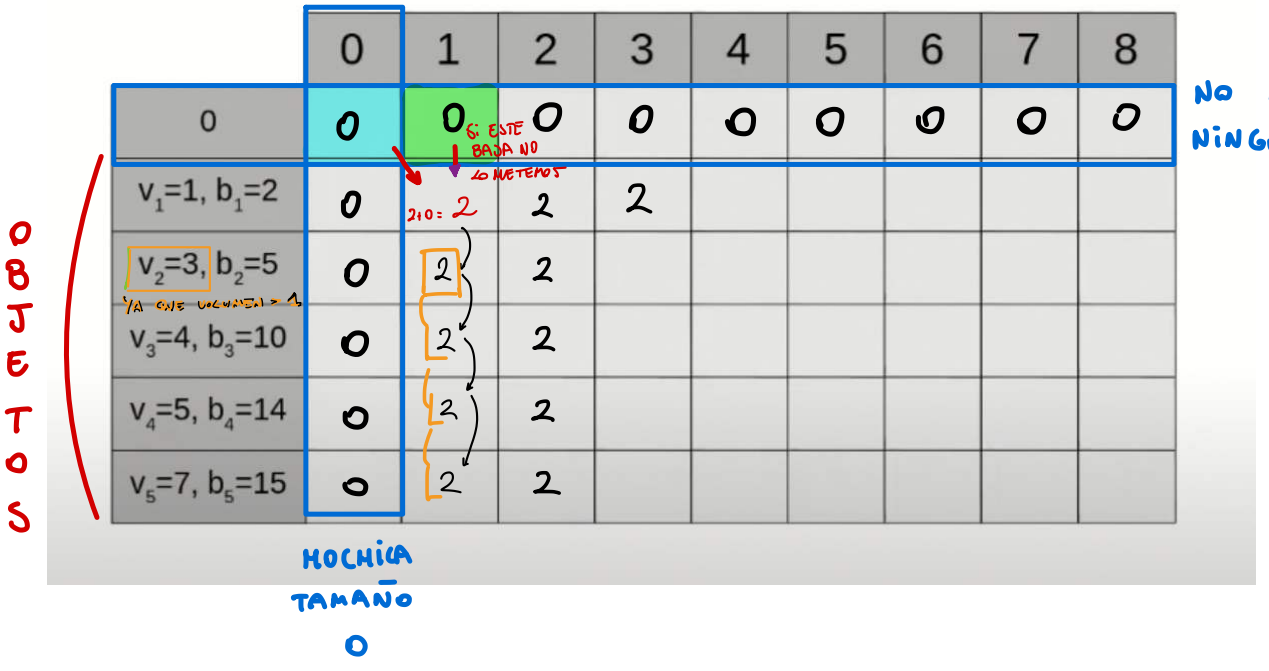
OBJETOS = $X_1 \dots X_n$

VOLUMENES = $V_1 \dots V_n$

BENEFICIOS = $b_1 \dots b_n$

CREAR MATRIZ M[i, j] = BENEFICIO DE LLENAR UNA MOCHILA DE VOLUMEN j CON OBJETOS DESDE X_1 A X_n

EN ESTE CASO TAMAÑO 8



	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
v_1=1, b_1=2	0	2	2	2					
v_2=3, b_2=5	0	2	2						
v_3=4, b_3=10	0	2	2						
v_4=5, b_4=14	0	2	2						
v_5=7, b_5=15	0	2	2						

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
v_1=1, b_1=2	0	2	2	2	2				
v_2=3, b_2=5	0	2	2	5					
v_3=4, b_3=10	0	2	2	5					
v_4=5, b_4=14	0	2	2	5					
v_5=7, b_5=15	0	2	2	5					

NO METER NINGÚN OBJETO

HOCHILA TAMAÑO 0

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
v_1=1, b_1=2	0	2	2	2	2				
v_2=3, b_2=5	0	2	2	5	7				
v_3=4, b_3=10	0	2	2	5	10				
v_4=5, b_4=14	0	2	2	5	10				
v_5=7, b_5=15	0	2	2	5	10				

NO METER NINGÚN OBJETO

COMO $b_3 = 10$ $10 > 7$

HOCHILA TAMAÑO 0

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
v_1=1, b_1=2	0	2	2	2	2	2			
v_2=3, b_2=5	0	2	2	5	7	7			
v_3=4, b_3=10	0	2	2	5	10	12			
~~v_4=5, b_4=14~~	0	2	2	5	10	14			
~~v_5=7, b_5=15~~	0	2	2	5	10	14			

NO METER NINGÚN OBJETO

HOCHILA TAMAÑO

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

NO METER NINGÚN OBJETO

NO METEMOS ULTIMO OBJETO

HOCHICA TAMAÑO 0

b_5 15 < 16

↓ 5° ELEMENTO

DESDE 1° ELEMENTO HASTA EL 4°

	0	1	2	3	4	5	6	7	8
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	17
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

NO METER NINGÚN OBJETO

HOCHICA TAMAÑO

TERMINAMOS

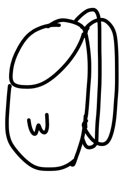
	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
$v_1=1, b_1=2$	0	2	2	2	2	2	2	2	2
✓ $v_2=3, b_2=5$	0	2	2	5	7	7	7	7	7
~~$v_3=4, b_3=10$~~	0	2	2	5	10	12	12	15	17
✓ $v_4=5, b_4=14$	0	2	2	5	10	14	16	16	19
~~$v_5=7, b_5=15$~~	0	2	2	5	10	14	16	16	19

NO METER NINGÚN OBJETO

IGUALES

MOCHILA TAMAÑO

BENEFICIO MÁXIMO 19 = $b_4 = 14 + b_2 = 5$



P [p_i] (i) → SERÍA NUESTRO VOLUMEN DEL EJEMPLO

b [b_i] (i) → BENEFICIO

$$BM(i, c) = \begin{cases} b_0 & i = 0 \wedge c \geq p_0 \\ 0 & i = 0 \wedge c < p_0 \\ BM(i-1, c) & c < p_i \\ \max\left(BM(i-1, c-p_i) + b_i,\ BM(i-1, c) + 0\right) & \end{cases}$$

i → OBJETO i-ésimo

c → CAPACIDAD

peso (p_i); NO SE METE ($+0$)

óptimo, óptimo → óptimo

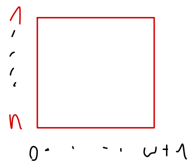
$BM(n-1, W)$

$n-1$ → ÚLTIMO OBJETO

W → CAPACIDAD TOTAL DE LA MOCHILA

ARRAY PESO — ARRAY BENEFICIO — CAPACIDAD MOCHILA

```java
public static Res mochila01(Integer[] peso, Integer[] beneficio, int W) {
    // A REALIZAR POR EL ALUMNO
    int n = peso.length;                 // LONGITUD DEL ARRAY
    int[][] BM = new int [n][W+1];       // CREANDO MATRIZ [n][w+1]
    for(int i = 0; i<n; i++){            // RECORRE LOS DOS ARRAYS
        for (int c=0 ; c <= W; c++){
            if(i == 0 && c >= peso[0]) BM[i][c] = beneficio[0];   // CASO BASE
            else if ( i == 0) BM[0][c] = 0;
            else if ( c< peso[i]) BM [i][c] = BM[i-1][c];         // ¿ANTERIOR?
            else BM [i][c] = Math.max(BM [i-1][c-peso[i]]+beneficio[i], BM[i-1][c]);
        }
    }
```



$$BM(i, c) = \begin{cases} b_0 & i=0 \wedge c \geq p_0 \\ 0 & i=0 \wedge c < p_0 \\ BM(i-1, c) & c < p_i \\ \max\left(BM(i-1, c-p_i)+b_i,\ BM(i-1, c)+0\right) & \end{cases}$$

i: OBJETO iÉSIMO — c: CAPACIDAD

¿DE ESTE CASO? (c < p_i) → NO SE METE

peso

óptimo — óptimo — óptimo

MARCHA ATRÁS

```java
        int[] sol = new int[n];   // ARRAY
        int i = n-1;              // longitud - 1

        int c = W;                // CAPACIDAD
        while (i>0){
            if (BM[i][c] == BM[i-1][c]){   // VA VIENDO DE DONDE SE COGEN
                sol[i] = 0;                // Y SI C SIGUE SIENDO IGUAL
                i--;
            }
            else{
                c-=peso[i];                // → c = c - peso[i]
                sol[i] = 1;
                i--;
            }
        }

        if(c>=peso[0]) sol[0] =1;
        else sol[0] = 0;

        return new Res(BM[n-1][W],sol);
    }
}//end class Mochila01
```

ESCRIBIR CÓDIGO

```
public static Res mochila01(Integer[] peso, Integer[] beneficio, int capacidad)

    int n = peso.length; // o podría poner beneficio.length
    int[][] BM = new int[n][capacidad+1];
    for(int i=0; i<n; i++){
        for(int c=0; c<=capacidad; c++){
            if(i==0 && c>=peso[0]) BM[i][c] = beneficio[0];
            else if(i==0) BM[0][c] = 0;
            else if(c<peso[i]) BM[i][c] = BM[i-1][c];
```