Problema de la mochila 0-1

(0,1-Knapsack problem)

Definición del problema

- Se tiene una colección de objetos numerados i=1,..,n.
- Los objetos **son indivisibles**, tienen un valor v_i y un peso w_i.
- Se tiene “una mochila” de capacidad W.
- Se pide escoger el subconjunto de objetos que no superen la capacidad de la mochila y que maximizen el valor total sin fraccionar los objetos.

Formulación del problema

Maximizar

$$\sum_{i=1}^{n} x_i v_i$$

Sujeto a

$$\begin{aligned} \sum_{i=1}^{n} x_i w_i &\leq W \\ x_i &\in \{0,1\} \end{aligned}$$

Nota:
La diferencia con la versión fraccional del problema son los valores permitidos de las variables de decisión.
En la versión fraccional:

$$x_i \in [0,1]$$

El algoritmo greedy no encuentra la respuesta óptima

Ejemplo

Consideremos la siguiente colección de objetos, con la restricción del peso total menor a 10:

	w	v	v/w
Obj1	2	80	40.00
Obj2	3	70	23.33
Obj3	4	60	15.00
Obj4	2	28	14.00
Obj5	3	35	11.67

La solución greedy devolvería los 3 primeros objetos, con peso total 9 y valor 210.

Principio de optimalidad

- El principio de optimalidad aplica a este problema:
  Dado un objeto X de peso w, la mejor solución está entre la solución anterior sin incluir a X, o la mejor solución para una mochila de capacidad W-w unida a X.

Solución dinámica

- Considerando un objeto a la vez, se determinan las mejores soluciones para una mochila de capacidad de 0..W.
- La matriz A[1..n,0..W] nos da el valor de la mejor solución incluyendo los objetos entre 1..i, que no sobrepasen un peso w.
- Por el principio de optimalidad, tenemos que los elementos de A cumplen:

$$a_{ij} = \max\{a_{i-1,j}, a_{i-1,j-w_i} + v_i\}$$

Solución dinámica

```
función knapsackBinario(W, w[1..n], v[1..n])
    A ← matriz[0..n,0..W]
    para i←0 hasta W
        A[0,i]←0
    para i←1 hasta n
        para j←0 hasta W
            si j=0
                A[i,j]←0
            si j-w[i]<0
                A[i,j]←A[i-1,j]
            si j-w[i]>=0
                A[i,j]←max(A[i-1,j],v[i]+A[i-1,j-w[i]])
    devolver A[n,W]
```

Ejemplo

- La matriz A para el ejemplo anterior es:

	w	v	0	1	2	3	4	5	6	7	8	9	10
Obj3	4	60	0	0	0	0	60	60	60	60	60	60	60
Obj2	3	70	0	0	0	70	70	70	70	130	130	130	130
Obj1	2	80	0	0	80	80	80	150	150	150	150	210	210
Obj4	2	28	0	0	80	80	108	150	150	178	178	210	210
Obj5	3	35	0	0	80	80	108	150	150	178	185	210	**213**

De la posición A[n,W] se concluye que la solución óptima tiene un valor de 213.

- Ejercicio: Qué objetos hacen parte de la solución óptima?

Ejercicios

1) Observar que en la matriz A los cambios ocurren en las columnas que son la suma de pesos de una combinación de objetos.
2) Dado un conjunto de n objetos, cuantas combinaciones se pueden hacer con ellos? (Sugerencia: Tener presente que cuentan las combinaciones de 1 objeto, de 2 objetos y así hasta de n objetos).
3) Hacer un algoritmo que enumere todas las combinaciones de objetos y determine su peso.
4) Proponer una implementación de `knapsackBinario` que utilice una matriz que solo utilice el número de columnas encontrado en (2).
5) Cuál es el orden asintótico del tiempo requerido por este nuevo algoritmo?