## El problema de la mochila

- Suponga que tiene $n$ objetos con volúmenes $v_1, v_2, \ldots, v_n$ y una mochila de volumen $s$.
- ¿Existirá un subconjunto de los objetos cuyo volumen total sea exactamente s?
- Existen muchas variantes del problema, por ejemplo:
  - los objetos tienen costos y se quiere maximizar el costo,
  - se desea maximizar el volumen que se llena de la mochila,
  - hay una cierta cantidad de cada tipo de objeto, etc.



## Algoritmo de divide y vencerás

- Sea $f(i,j)$ la respuesta a la pregunta ¿existe un subconjunto de los primeros $i$ objetos tal que su volumen total sea $j$?
- ¿Cuándo es $f(i,j)$ verdadera?
  - Si $i = 0$ y $j = 0$ (caso base).
  - Si $i \geq 1$ y $f(i-1,j)$ es verdadera (no se usa el objeto $j$).
  - Si $i \geq 1$, $j \geq v_i$ y $f(i-1, j-v_i)$ es verdadera.
- En todos los demás casos $f(i,j)$ es falsa.
- Es fácil ver que este algoritmo se tarda $\Theta(2^n)$ en calcular $f(n,s)$.

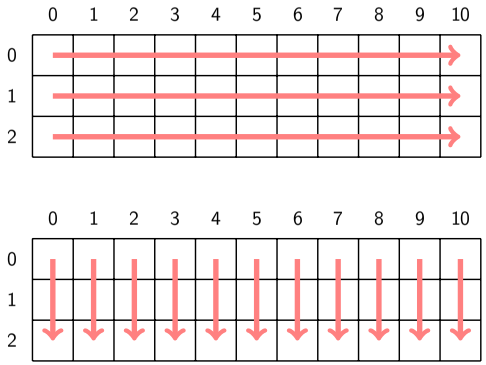


## Algoritmo de programación dinámica

- Almacenemos el valor de $f(i,j)$ en la entrada $F_{i,j}$ de una tabla.
- Observe que para calcular $F_{i,j}$ se necesita saber cuánto valen $F_{i-1,j}$ y $F_{i-1,j-v_i}$.
- En otras palabras, se necesita saber cuánto valen $F_{i-1,0}, F_{i-1,1}, \ldots, F_{i-1,j}$.
- Se puede escoger un orden adecuado para calcular las entradas de $P$ de modo que cada cálculo sólo depende de otros cálculos ya hechos.
- Hay varias posibilidades para ese orden.
- Todas ellas se pueden ejecutar en tiempo $O(ns)$.
- Con cuidado sólo necesitarán espacio $O(n)$.



## Dos formas de llenar la tabla $F$






## Problema de la mochila

- Recordemos que el problema de la mochila consiste en decidir si existe algún subconjunto de $\{s_1, s_2, \ldots, s_n\}$ que sume $S$.
- Usemos una cadena de bits $A[1, \ldots, n]$ para resolver este problema.
- $A[i] = 1$ significará que usaremos el objeto $i$ y $A[i] = 0$ que no.
- Si revisamos todas las cadenas binarias podemos resolver el problema.
- Pero además podemos hacer una poda. Si $S$ es la suma restante entonces:
  - hacer $A[m] = 0$ siempre será legal pero
  - hacer $A[m] = 1$ sólo será legal si $s_m \leq S$.



## Algoritmo

- Para imprimir todas las soluciones llame a mochila$(n, S)$.
- Este algoritmo toma tiempo $O(2^n)$ (mas el tiempo que se tarde en imprimir las $t$ soluciones $O(tn)$).

**Procedimiento mochila$(m, \ell)$**

- Si $m = 0$ entonces:
  - Si $\ell = 0$ entonces imprime$(A)$.
- Si no:
  - $A[m] = 0$ y mochila$(m-1, \ell)$.
  - Si $s_m \leq \ell$ entonces
    - $A[m] = 1$ y mochila$(m-1, \ell - s_m)$.



## Problema continuo de la mochila

- Este problema es parecido a uno que vimos antes.
  1. Se tienen $n$ objetos $A_1, A_2, \ldots, A_n$.
  2. Se tiene una mochila de volumen $S$.
  3. El objeto $A_i$ tiene volumen $v_i$.
  4. El objeto $A_i$ tiene peso $p_i$.
  5. Una fracción $0 \leq x_i \leq 1$ del objeto $A_i$ tiene volumen $x_i v_i$ y peso $x_i p_i$.
- Se desea llenar la mochila tanto como sea posible usando fracciones de los objetos de modo que el peso sea el mínimo posible.



## Ejemplo

- Suponga que $S = 20$, que los volúmenes son $v_1 = 18$, $v_2 = 10$ y $v_3 = 15$ y que los pesos son $p_1 = 25$, $p_2 = 15$ y $p_3 = 24$.
- Algunas posibilidades para las fracciones son:

| $(x_1, x_2, x_3)$ | volumen total | peso total |
|---|---|---|
| (1, 1/5, 0) | 20 | 28.0 |
| (1, 0, 2/15) | 20 | 28.2 |
| (0, 1, 2/3) | 20 | 31.0 |
| (0, 1/2, 1) | 20 | 31.5 |

- La primera opción es la mejor de las que llevamos.
- ¿Pero cómo saber si es la mejor de todas?



## Algoritmo glotón (informal)

- Defina la densidad del objeto $A_i$ como $p_i/v_i$.
- Use tanto como sea posible de los objetos de baja densidad.
- En otras palabras, procese los objetos en orden de densidad creciente.
- Si todo cabe, úselo todo.
- Si no, llene el espacio restante con una fracción del objeto actual.
- Descarte el resto.