Diseño y Análsis de Algoritmos INF-413

Sergio Hernández
PhD computer science

Universidad Católica del Maule.
shernandez@ucm.cl

Introducción

- Las entradas del sistema son el conjunto de items $X = \{x_1, \ldots, x_n\}$, los pesos de cada item $W = \{w_1, \ldots, w_n\}$ y el valor de cada item $V = \{v_1, \ldots, v_n\}$ y w_{max} la capacidad de la mochila.

Introducción

- Las entradas del sistema son el conjunto de items $X = \{x_1, \ldots, x_n\}$, los pesos de cada item $W = \{w_1, \ldots, w_n\}$ y el valor de cada item $V = \{v_1, \ldots, v_n\}$ y w_{max} la capacidad de la mochila.

Definición

Si hacemos x_i sea una variable que indica la presencia del item i. El problema de la mochila consiste en solucionar el siguiente problema de optimización:

$$\sum_{i}^{n} x_i v_i = F(X) \tag{1}$$

$$\sum_{i}^{n} x_i w_i \leq w_{max} \tag{2}$$

Ejemplo

Supongamos que contamos con una mochila que tiene capacidad maxima $w_{max} = 4[kg]$ y tenemos a disposicion los siguientes items:

Item	Peso [kg]	Valor [$]	Valor/Peso
Guitarra	1	1500	1500
Laptop	3	2000	666.7
Stereo	4	3000	750

Programacion dinamica Mochila 0/1

- El algoritmo de fuerza bruta requiere calcular 2^n posibles estados y luego elegir entre los que cumplen con la restriccion.

Programacion dinamica Mochila 0/1

- El algoritmo de fuerza bruta requiere calcular 2^n posibles estados y luego elegir entre los que cumplen con la restriccion.
- El problema de la mochila 0/1 tiene la propiedad de substructura optima ya que la solucion optima puede ser creada a partir de las soluciones optimas de los sub-problemas.

Programacion dinamica Mochila 0/1

- El algoritmo de fuerza bruta requiere calcular 2^n posibles estados y luego elegir entre los que cumplen con la restriccion.
- El problema de la mochila 0/1 tiene la propiedad de substructura optima ya que la solucion optima puede ser creada a partir de las soluciones optimas de los sub-problemas.
- Como todos los problemas de programacion dinamica, podemos escribir los sub-problemas optimos en una tabla con elementos y pesos maximos $c[i][j]$.

$$c[i][j] = \text{MAX} \begin{cases} c[i-1][j] \\ v_i + c[i-1][j-w_i] \end{cases}$$

Subestructura Optima

Item	Peso [kg]	Valor [$]	Valor/Peso
Guitarra	1	1500	1500
Laptop	3	2000	666.7
Stereo	4	3000	750

Item	1 [kg]	2 [kg]]	3 [kg]	4 [kg]
Guitarra				
Stereo				
Laptop				