UADE

Exposición de experto

Temas a desarrollar

1	Definición problema mochila
2	Síntesis

Problema de la mochila

Problema de la mochila *Greedy*

- El problema consiste en que se tienen **n** objetos y una mochila. Para ***i = 1; 2; ..n***, el objeto ***i*** tiene un peso positivo ***Pi*** y un valor positivo ***Vi***. La mochila puede llevar un peso que no sobrepase **P**. El objetivo es llenar la mochila de tal manera que se maximice el valor de los objetos transportados, respetando la limitación de capacidad impuesta.

- Los objetos pueden ser fraccionados, si una fracción ***Xi*** (0 <= Xi <= 1) del objeto ***i*** es ubicada en la mochila contribuye en ***Xi * Pi*** al peso total de la mochila y en ***Xi * Vi*** al valor de la carga.

Problema de la mochila *Greedy* - Ejemplo

Supongamos que tenemos cuatro objetos y una mochila con capacidad *P* = 17. Los valores de los objetos son (v1,v2,v3,v4) = (5,2,7,4) y los pesos de los objetos son (p1,p2,p3,p4) = (9,5,3,6).
Algunas combinaciones para completar la mochila son:

x1	x2	x3	x4	xi*p1	xi*vi
1	1/2	1/2	2/3	17,00	15,33
1	1	1	0	17,00	7,00
1	0	1	5/6	17,00	28,33
8/9	0	1	1	17,00	32,44

La opción que maximiza respeta la relación *v/p* de mayor a menor:

	Valor				Peso			
	5	2	7	4	9	5	3	6
v/p	0,5556	0,4	2,3333	0,6667				

Problema de la mochila *Greedy* - Estrategia

Conjunto candidatos	Los diferentes objetos aún disponibles a utilizar
Función selección	Seleccionar el objeto de mayor relación *v/p*
Función factibilidad	Validar que el objeto no supere el peso de la mochila, sino ingresar una fracción del mismo
Función solución	Verificar que la mochila se haya completado o se hayan utilizado todos los objetos
Función objetivo	Maximizar el beneficio obtenido en base a los objetos ingresados en la mochila

Problema de la mochila *Greedy* - Algoritmo

ALGORITMO MOCHILA

Entrada: O: Vector<Objeto>, p: entero
Salida: R: Vector<real>

$Ordenar(O)$ //según la razón valor/peso
para $i = 0$ **hasta** $n - 1$

$R[i] \leftarrow 0$
fin para
$suma \leftarrow 0$
$objeto \leftarrow 0$
mientras $suma < p$
$R[objeto] \leftarrow MIN(1, (p - suma)/O[objeto].peso)$
$suma \leftarrow suma + MIN(1, (p - suma)/O[objeto], peso) * O[objeto].peso$
$objeto \leftarrow objeto + 1$
fin mientras
devolver R

Problema de la mochila *Greedy* – Complejidad temporal

ALGORITMO MOCHILA

Entrada: O: Vector<Objeto>, p: entero
Salida: R: Vector<real>

$\quad Ordenar(O)$ //según la razón valor/peso

$\quad$ para $i = 0$ hasta $n - 1$

$\quad\quad R[i] \leftarrow 0$

$\quad$ fin para

$\quad suma \leftarrow 0$

$\quad objeto \leftarrow 0$

$\quad$ mientras $suma < p$

$\quad\quad R[objeto] \leftarrow MIN(1, (p - suma)/O[objeto].peso)$

$\quad\quad suma \leftarrow suma + MIN(1, (p - suma)/O[objeto], peso) * O[objeto].peso$

$\quad\quad objeto \leftarrow objeto + 1$

$\quad$ fin mientras

$\quad$ devolver R

$\boxed{\Theta(n \log(n))}$

La cantidad máxima que itera es utilizando todos los objetos

$\boxed{\mathcal{O}(n)}$

Complejidad del algoritmo $\boxed{\Theta(n \log(n))}$

Problema de la mochila *Greedy* – Correctitud

- La solución del problema de la mochila por *Greedy* para elementos que se pueden fraccionar es una solución correcta.

- Se van a ver otras variantes del problema que se deben resolver con otras técnicas de diseño de algoritmo, para el caso de que los elementos no se puedan fraccionar.

Nota: BRASSARD, Gilles. *Fundamentals of algorithmics.* Prentice Hall, 1996. ISBN: 9780133350685

Síntesis

Síntesis

- El problema consiste en que se tienen **n** objetos y una mochila. Para ***i = 1; 2; ..n***, el objeto ***i*** tiene un peso positivo ***Pi*** y un valor positivo ***Vi***. La mochila puede llevar un peso que no sobrepase **P**. El objetivo es llenar la mochila de tal manera que se maximice el valor de los objetos transportados, respetando la limitación de capacidad impuesta. Los objetos se pueden fraccionar.

- El criterio de selección consiste en ir utilizando el elemento con mayor relación *v/p*.

- La complejidad temporal es de *O(n log n),* siendo *n* la cantidad de objetos disponibles para seleccionar y completar la mochila.

- El algoritmo diseñado a partir del criterio de selección definido es óptimo en todos los casos, siempre considerando que se pueden fraccionar los elementos.

Bibliografía

BRASSARD, Gilles. *Fundamentals of algorithmics.* Upper Saddle River: Prentice Hall, 1996. ISBN: 9780133350685

¡Muchas gracias!