Análisis y Diseño de Algoritmos
Algoritmos Voraces

Gilberto Gutiérrez

Departamento de Ciencias de la Computación y TI
Universidad del Bío-Bío
Chillán / Chile

June 16, 2016

Contenido

El problema de dar cambio

1: **ConjuntoMonedas DevolverCambio(n)** {Da el cambio de n unidades utilizando el menor número posible de monedas. C especifica un arreglo con las denominaciones}
2: $C = \{20000, 10000, 5000, 2000, 1000, 500, 100, 50, 10, 5, 1\}$
3: $S \leftarrow \emptyset$ {Conjunto que contendrá la solución}
4: $s \leftarrow 0$ {s es la suma de los elementos de S}
5: **while** $s \neq n$ **do**
6: $x \leftarrow$ el mayor elemento de C tal que $s + x \leq n$
7: **if** No existe ese elemento **then**
8: **return** "No existe solución"
9: **end if**
10: $S \leftarrow S \cup \{$ una moneda de valor $x\}$
11: $s \leftarrow s + x$
12: **end while**
13: **return** S

Características Generales

- Problema de optimización (mínima cantidad de monedas)
- Se dispone de un conjunto de candidatos(Monedas)
- Conforme avanza el algoritmo se mantienen dos conjuntos
  - Conjunto con candidatos considerados y seleccionados
  - Conjunto con candidatos considerados y rechazados
- Se consideran 4 funciones
  - Verifica si un conjunto de candidatos es solución (ignorando si es óptima). ¿Suman las monedas seleccionadas la cantidad que hay que pagar?
  - Verifica si una solución es factible. $s + x \leq n$
  - Función de selección. Elige el candidato más prometedor de los candidatos que no han sido seleccionados o rechazados.
  - Función objetivo. Da el valor de la solución. Número de monedas. No aparece explícitamente en el algoritmo.
- La función de selección suele estar relacionada con la función objetivo.
- Nunca cambian de opinión.

Esquema general Algoritmos Voraces

1: **voraz(Conjunto** C**)** $\{C$ Conjunto de candidatos$\}$
2: $S \leftarrow \emptyset$ {Conjunto que contendrá la solución}
3: **while** $C \neq \emptyset$ *and not solucion*(S) **do**
4: $x \leftarrow$ *seleccionar*(C)
5: $C \leftarrow C - \{x\}$
6: **if** *factible*$(S \cup \{x\})$ **then**
7: $S \leftarrow S \cup \{x\}$
8: **end if**
9: **end while**
10: **if** *solucion*(S) **then**
11: **return** S
12: **else**
13: **return** "No hay solución"
14: **end if**

El problema de la Mochila

Nos dan n objetos y una mochila. Para $i = 1, 2, \ldots, n$, el objeto i tiene un peso positivo w_i y un valor (económico) v_i. La mochila puede llevar un peso que no sobrepase W. El objetivo es llenar la mochila de tal manera que se maximice el valor de los objetos transportados, respetando la limitación de capacidad impuesta. Suponga que es posible romper los objetos en trozos más pequeños de modo que podemos decidir llevar sólo una fracción x_i del objeto i, es decir, $0 \leq x_i \leq 1$.

1. Diseñe un algoritmo voraz para determinar los objetos que es conveniente cargar en la mochila
2. Indique por qué cree que su algoritmo es voraz ?. ¿ Dónde está lo "voraz" ?
3. Compare su solución con la entregada por el profesor.
4. Asocie las características generales de los algoritmos voraces a las características particulares de este problema. Identifique el conjunto de candidatos y las diferentes funciones.

Solución problema de la Mochila

La función objetivo que queremos optimizar es:

maximizar $\sum_{i=1}^{n} x_i v_i$ con la restricción $\sum_{i=1}^{n} x_i w_i \leq W$, donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

1: **Mochila(**$w[1 \ldots n], v[1 \ldots n], W$**)**
2: Sea $x[1 \ldots n]$ $\{x$ es un vector para almacenar la fracción de cada objeto que va en la mochila$\}$
3: $x[i] \leftarrow 0$ para todo $1 \leq i \leq n$
4: $peso \leftarrow 0$
5: **while** $peso < W$ **do**
6: $\quad i \leftarrow$ el mejor objeto restante
7: $\quad$ **if** $(peso + w[i]) \leq W$ **then**
8: $\quad\quad x[i] \leftarrow 1$
9: $\quad\quad peso \leftarrow peso + w[i]$
10: $\quad$ **else**
11: $\quad\quad x[i] \leftarrow (W - peso)/w[i]$
12: $\quad\quad peso \leftarrow W$
13: $\quad$ **end if**
14: **end while**
15: **return** x

Funciones de selección para el problema de la mochila

1. El objeto más valioso (Máx v_i)
2. El más liviano (Mín w_i)
3. El objeto cuyo valor por unidad de peso sea el mayor (Máx $\frac{v_i}{w_i}$)

Ejemplo: $n = 5, w = 100$

w	10	20	30	40	50
v	20	30	66	40	60
$\frac{v}{w}$	2.0	1.5	2.2	1.0	1.2

Resultados con cada una de las funciones de seleción.

Seleccionar:	x_i					Valor
Máx v_i	0	0	1	0.5	1	146
Mín w_i	1	1	1	1	0	156
Máx $\frac{v_i}{w_i}$	1	1	1	0	0.8	164