Analisis y Diseño de Algoritmos

Juan Gutiérrez

June 2, 2022

Intervalos disjuntos

Problema Max-Intervalos-Disjuntos. Dada una secuencia de intervalos cerrados en la recta, encontrar un subconjunto de intervalos compatibles dos a dos de tamaño máximo.

Intervalos disjuntos

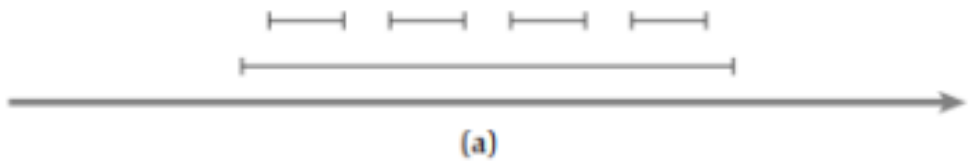

Figure 1: Tomada del libro Kleinberg, Algorithm Design

Intervalos disjuntos

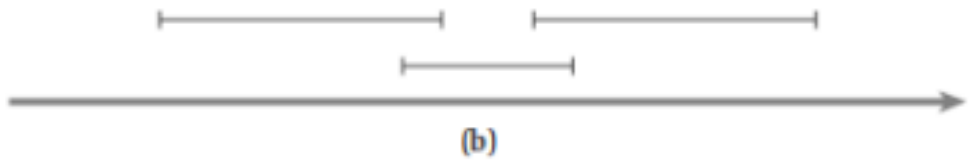

Figure 2: Tomada del libro Kleinberg, Algorithm Design

Intervalos disjuntos

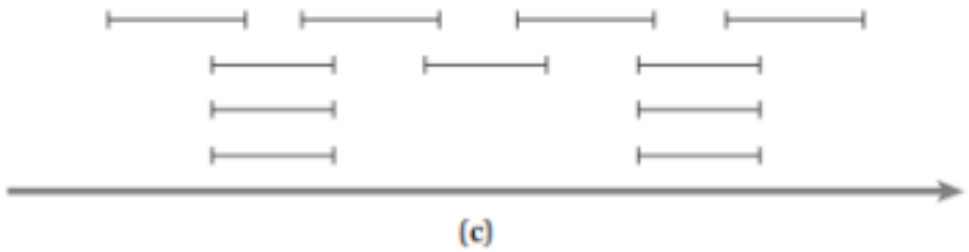

Figure 3: Tomada del libro Kleinberg, Algorithm Design

Intervalos disjuntos

Recibe: un conjunto $\mathcal{I} = \{[s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n]\}$ de intervalos
Devuelve: un subconjunto de intervalos compatibles dos a dos

MAX-INTERVALOS-DISJ($\mathcal{I}$)

1: $A = \emptyset$
2: **while** $\mathcal{I} \neq \emptyset$
3: Sea $[s_i, f_i] \in \mathcal{I}$ tal que f_i es mínimo
4: $A = A \cup \{[s_i, f_i]\}$
5: $\mathcal{I} = \mathcal{I} \setminus \{[s_k, f_k] : [s_k, f_k] \cap [s_i, f_i] \neq \emptyset\}$
6: **return** A

Intervalos disjuntos

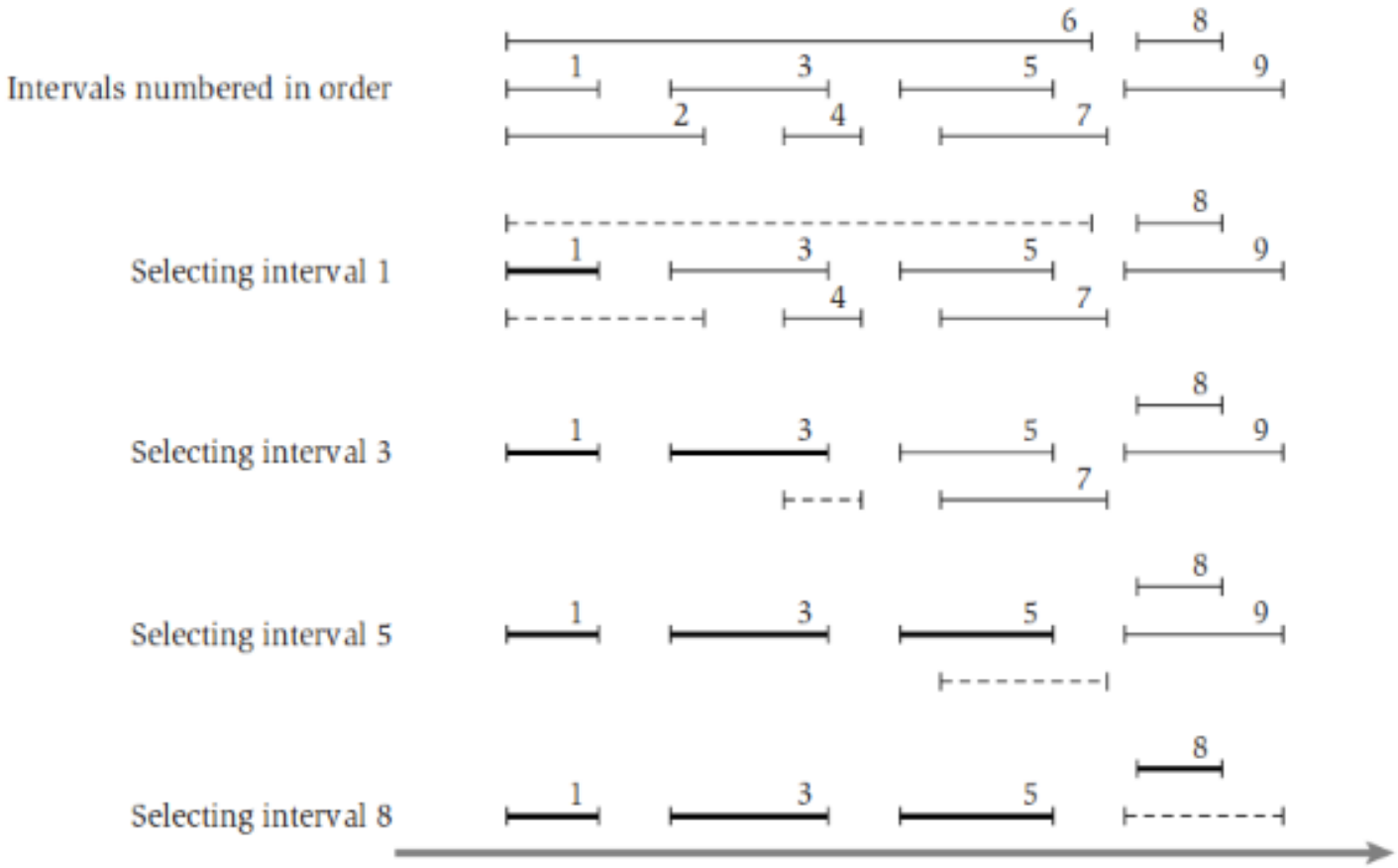

Figure 4: Tomada del libro Kleinberg, Algorithm Design

Elementos de la estrategia Voraz

1. *Elección voraz:* debemos demostrar que siempre existe una solución óptima que contiene a la eleccción voraz
2. *Subestructura óptima*: debemos demostrar que la subsolución dejada es óptima para el subproblema dejado por la elección voraz

Elementos de la estrategia Voraz

Problema Max-Intervalos-Disjuntos. Dada una secuencia de intervalos cerrados en la recta, encontrar un subconjunto de intervalos compatibles dos a dos.

Elementos de la estrategia Voraz

Recibe: un conjunto $\mathcal{I} = \{[s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n]\}$ de intervalos, ordenados de manera creciente por punta final
Devuelve: un subconjunto de intervalos compatibles dos a dos

MAX-INTERVALOS-DISJ-REC($\mathcal{I}$)
1: **if** $\mathcal{I} = \emptyset$
2: return $\emptyset$
3: $\mathcal{I}' = \mathcal{I} \setminus \{[s_i, f_i] : s_i \leq f_1\}$
4: **return** $\{[s_1, f_1]\} \cup$ MAX-INTERVALOS-DISJ-REC($\mathcal{I}'$)

Elementos de la estrategia Voraz

Lema 3.1 (Elección voraz). *Existe una solución óptima para el problema que contiene el intervalo* $[s_1, f_1]$.

Elementos de la estrategia Voraz

Lema 3.2 (Subestructura óptima). *Si X es una solución óptima al problema que contiene a $[s_1, f_1]$ entonces $X \setminus \{[s_1, f_1]\}$ es una solución óptima al subproblema dejado por la elección voraz.*

Mochila fraccionaria

Problema Mochila-entera. Dado un conjunto $\{1, 2, \ldots, n\}$ de items cada uno con un peso natural w_i, un valor natural v_i y un número natural W, encontrar un subconjunto de items cuya suma de valores es la mayor posible, pero menor o igual a W.

Mochila fraccionaria

Problema Mochila-fraccionaria. Dado un conjunto $\{1, 2, \ldots, n\}$ de items cada uno con un peso natural w_i, un valor natural v_i y un número natural W, encontrar un vector de racionales entre 0 y 1 $(x_1, x_2, \ldots, x_n)$ que maximize $\sum_{i=1}^{n} x_i v_i$ sobre la restricción $\sum_{i=1}^{n} x_i w_i \leq W$

Mochila fraccionaria

Recibe: Una instancia v, w, W del problema MOCHILA-FRACCIONARIA
Devuelve: Una solución óptima para dicha instancia
MOCHILAFRACCIONARIA-GREEDY(v, w, W)

```
1: for j = n to 1
2:     if w[j] ≤ W
3:         x_j = 1
4:         W = W − w[j]
5:     else
6:         x_j = W/w[j]
```

Mochila fraccionaria

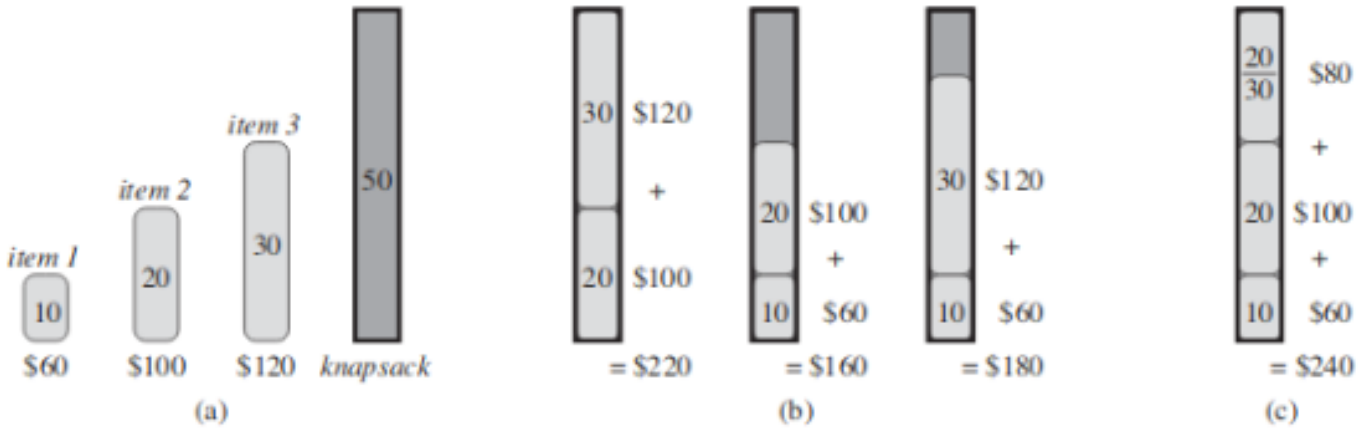

Figure 8: Tomada del libro Cormen, Introduction to Algorithms

Mochila fraccionaria

Lema 4.1 (Elección voraz). *Existe una solución óptima* $(x_1, x_2, \ldots, x_n)$ *al problema tal que* $x_n = \min\{1, W/w_n\}$

Mochila fraccionaria

Lema 4.2 (Subestructura óptima). *Si* $(x_1, x_2, \ldots, x_n)$ *es una solución óptima al problema con* $x_n = \min\{1, W/w_i\}$, *entonces* $(x_1, x_2, \ldots, x_{n-1})$ *es una solución óptima al subproblema dejado con* $W = W - x_n w_n$.

Árboles de Huffman

	a	b	c	d	e	f
Frequency (in thousands)	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101
Variable-length codeword	0	101	100	111	1101	1100

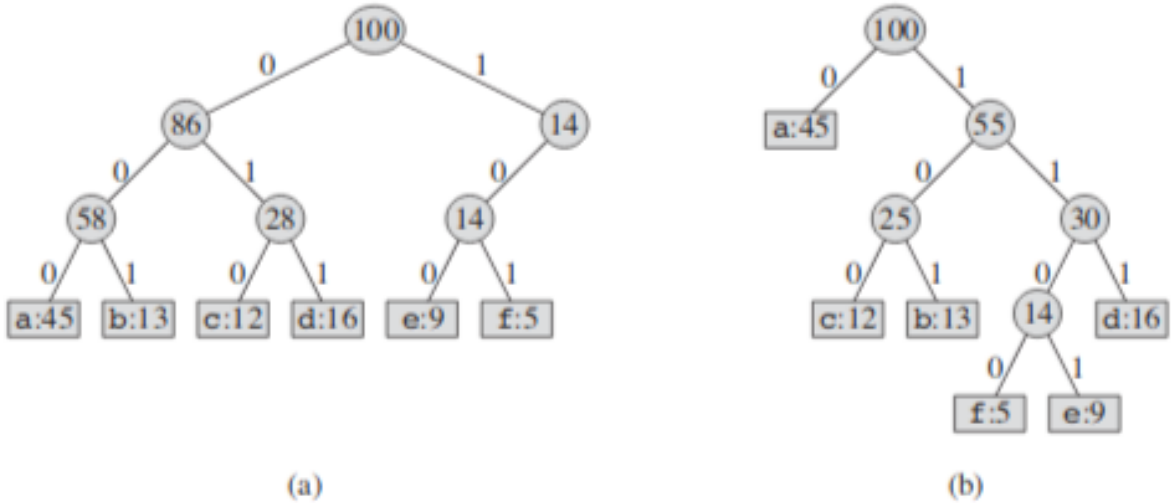

Figure 9: Tomada del libro Cormen, Introduction to Algorithms

Árboles de Huffman

Problema 5.1. *(Problema de compresión) Dado un archivo de caracteres, encontrar una tabla de códigos libre de prefijos que produzca un archivo codificado de tamaño mínimo.*

Árboles de Huffman

Sea $S = \{1, 2, \ldots, n\}$. Un *árbol de Huffman* respecto a S es cualquier colección Π de subconjuntos de S que cumple las siguientes propiedades.

1. para cada X y cada Y en Π, se tiene que $X \cap Y = \emptyset$, o $X \subseteq Y$ o $Y \subseteq X$,
2. $S \in \Pi$,
3. $\{\} \notin \Pi$,
4. todo elemento no minimal en Π, es la unión de otros dos elementos en Π.

Árboles de Huffman

$$p(\Pi) = \sum_{X \in \Pi - \{S\}} p(X).$$

Árboles de Huffman

Recibe: Un conjunto S, una ponderación p de S y una partición Γ de S
Devuelve: Un árbol de Huffman óptimo (con peso mínimo) que tiene a Γ como conjunto de hojas

HUFFMAN(S, p, Γ)
1: **if** $|\Gamma| = 1$
2: **return** Γ
3: Sea X un elemento en Γ con ponderación mínima
4: $\Gamma = \Gamma - X$
5: Sea Y un elemento en Γ con ponderación mínima
6: $\Gamma = \Gamma - Y$
7: $\Gamma = \Gamma \cup \{X \cup Y\}$
8: **return** $\{X, Y\} \cup$ HUFFMAN(S, p, Γ)

Ejemplo: sea $S = \{1, 2, \ldots, 6\}$, $p(1) = 45, p(2) = 13, p(3) = 12, p(4) = 16, p(5) = 9, p(6) = 5$ y $\Gamma = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}$.

El algoritmo produce el árbol $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{6, 5\}, \{6, 5, 4\}, \{2, 3\}, \{2, 3, 4, 5, 6\}, \{1, 2, 3, 4, 5, 6\}\}$. Su peso es 224.

Árboles de Huffman

Lema 5.1. *(Elección voraz) Sean X e Y dos hojas con ponderación mínima. Existe un árbol de Huffman óptimo Π que tiene a X e Y como hojas hermanas de profundidad máxima.*

Árboles de Huffman

Recibe: Un conjunto S, una ponderación p de S
Devuelve: Un árbol de Huffman óptimo (con peso mínimo) que tiene a los elementos de S como conjunto de hojas

HUFFMAN-FILAPRIORIDADES(S, p)

```
1:  n = |S|
2:  Q = INICIAR-FP()
3:  for i = 1 to n
4:      z.peso = p(i)
5:      z.left = NIL
6:      z.rigth = NIL
7:      INSERT-FP(Q, z)
8:  for i = 1 to n − 1
9:      x = EXTRAERMIN-FP(Q)
10:     y = EXTRAERMIN-FP(Q)
11:     z.left = x
12:     z.rigth = y
13:     z.peso = x.peso + y.peso
14:     INSERT-FP(Q, z)
15: return EXTRAERMIN-FP(Q)
```

Gracias