INF221 – Algoritmos y Complejidad

Ayudantía #4
Algoritmos Voraces

Aldo Berrios Valenzuela

30 de agosto de 2016

1. Algoritmos Voraces

Los algoritmos voraces encuentran una solución óptima, pero no necesariamente la *más* óptima.

Para demostrar que un algoritmo es voraz tenemos que demostrar que:

- *Greedy Choice:* Para toda instancia[1] P, tal que hay una solución óptima que incluye $\hat{p}$; este es el primer elemento elegido.

  `/* Dibujo feo: un circulo deforme que contiene` P `e inscrito a este tiene otro circulo` P'`. Leyenda: cada vez vamos escogiendo` $\hat{p}$ `tales que van achicando nuestro problema cada vez más. */`

- *Inductive Structure:* Dada la elección de $\hat{p}$, queda un subproblema P' (menor que el anterior), tal que si Π' es solución óptima de P', $\{\hat{p}\} \cup \Pi'$ es solución viable de P. `/* en el fondo, cada vez que sacamos un` $\hat{p}$ `para formar un nuevo` P'`, el resultado siempre será el mismo problema que tenemos que resolver */`

- *Optimal Subestructure:* Si Π' es solución óptima para P' y $\hat{p}$ es la elección voraz para P, entonces $\{\hat{p}\} \cup \Pi'$ es solución óptima para P.

Ejemplo 1.1 (Árbol de Compensión de Hufmann)**.** Es un algoritmo que utiliza `mp3` para comprimir datos. Nos dan una tabla con frecuencia y símbolo (Cuadro <>) Y el arbol generado es:

Símbolo	Frecuencia
A	70 millones
B	5 millones
C	20 millones
D	37 millones

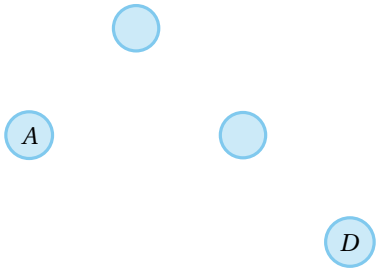
`/* Colocar arbol: valores más repetidos están más arriba */`

Y luego el código de ese arbol está dado por el cuadro <>: Luego, buscamos el costo del arbol:

$$\text{Cost} = \sum_{i=1}^{n} f_i \text{Depth}_i \tag{1.1}$$

El árbol inicial es:

[1] cuando hablamos de instancia es como decir un "momento" del problema



Símbolo	Código
A	0
B	100
C	101
D	11

```
/* Arbol inicial */
```

En cada paso del algoritmo vamos sumando los menores para formar una expresión y llegar al grafo.

$$\underbrace{C}_{20}, \underbrace{B}_{3}, \underbrace{D}_{37}, \underbrace{A}_{70}$$
$$(C+B), D, A$$
$$((C+B)+D), A$$
$$(((C+B)+D)+A)$$

```
/* Grafo: En el fondo, empezamos con los elementos que están dentro del mismo paréntesis.
   Por otro lado, todos aquellos que estén dentro del mismo paréntesis tienen al mismo
                                        padre. */
```

Ahora que sabe como funciona el árbol de compensación de Hufmann, demuestre que es óptimo. ■

Demostración. Demostramos cada uno de los puntos de algoritmo voraz:

- *Greedy choice:*

  C y B son la mejor opción para el nivel más bajo. Opción:

  $$x \cdot 1 + y \cdot 2 + \underbrace{23}_{C+B} \cdot 3$$ `/* 1, 2, y 3 representan el nivel en el arbol */`

- *Inductive structure:*

  ```
  /* Explicar el tema de juntar los nodos, y ver que el problema que nos queda será el mismo
  que teníamos inicialmente, pero reducido. Si continuamos haciendo esta jugarreta siempre
  tendremos el mismo problema. Por lo tanto, cumple con la estructura óptima. */
  ```

- *Optimal Subestructure:*

□

Ejemplo 1.2 (Algoritmo de Kruskal)**.** Genera un arbol recubridor mínimo:

```
/* colocar ambos grafos: original y recubridor mínimo */
```

Demuestre que es óptimo. ■

Demostración. Demostramos cada uno de los puntos de algoritmo voraz:

- *Greedy choice:*

□