Huffman, HuTucker y Cie

Note Title 7/30/2009

I Huffman

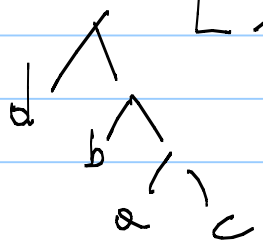
$[n] = \{1, \dots, n\}$

Input $(p_i)_{i \in [n]}$

Output $(c_i)_{i \in [n]}$, $(l_i)_{i \in [n]}$

libre de prefijos (codigos binarios)

a, b, c, d
.1 .3 .2 .4
.3
.6
1



① Shannon: teoría de la información

cantidad de información de un símbolo c_i es $\log_2 \frac{1}{p_i}$

$H =$ Entropía$(p_1, \dots, p_n) = \sum p_i \log_2 \frac{1}{p_i}$

$\Rightarrow$ Cota inferior sobre el costo de | transmisión / codificación | de una secuencia de símbolos.

② Rendimiento del codigo de Huffman

Pregunta: Huffman es optimo ?

No es optimo

$Perf(Huffman) = n(1 + H(p_1, \ldots, p_n))$

Mejor caso: $\forall i \quad p_i = \frac{1}{2^{h_i}}$

Peor caso: $\exists i \quad p_i \gg .5$

③ Hu Tucker. "Alphabetic Codes"

Input: $(p_1, \ldots, p_n)$

orden sobre el alfabeto

Output $(c_1, \ldots, c_n)$ tal que

$c_1, \ldots, c_n$ son en el mismo orden

libres de prefijos

Algoritmo

Reglas ① Unificar hojas solamente si: están contiguas.

② Unificar nodos internos solamente si no hay hojas "no mezcladas" a dentro.

Algoritmo: – construir el arbol con estas reglas
– girar los nodos internos de manera a reordenar las hojas.

$\Rightarrow$ Perf (Hu Tucker) $= n(2 + H(p_1, \ldots, p_n))$

Ejemplos

① Peor caso $\quad p_i = \frac{1}{n} \quad \forall i$

$$H = \sum p_i \lg \frac{1}{p_i} = \frac{n}{n} \lg n$$

② Mejor caso

$$\begin{cases} p_i = \frac{1}{2^i} & \forall i \in [n-1] \\ p_n = \frac{1}{2^{n-1}} \end{cases}$$

$$H = \sum p_i \lg \frac{1}{p_i}$$

$$= \sum_{i=1}^{n-1} \frac{1}{2^i} \lg \frac{1}{1/2^i} + \frac{1}{2^{n-1}} \lg \frac{1}{1/2^{n-1}}$$

$$= \sum_{i=1}^{n-1} \frac{i}{2^i} + \frac{n-1}{2^{n-1}} \geq \sum_{i=1}^{\infty} \frac{i}{2^i} \geq \sum_{i=1}^{n-1} \frac{i}{2^i}$$

$$\sum_{i=1}^{\infty} i c^i = \frac{c}{1-c} \quad \forall c<1 \Rightarrow \sum_{i=1}^{\infty} \frac{i}{2^i} = \frac{1/2}{1-1/2} = \frac{1}{2-1} = 1$$

Mostralo con integrales

Pregunta: Relación a la busqueda?

Secuencia de simbolos ordenados pore frecuencia. Buscar para un simbolo secuencialmente

Rendimiento : $\sum i p_i$

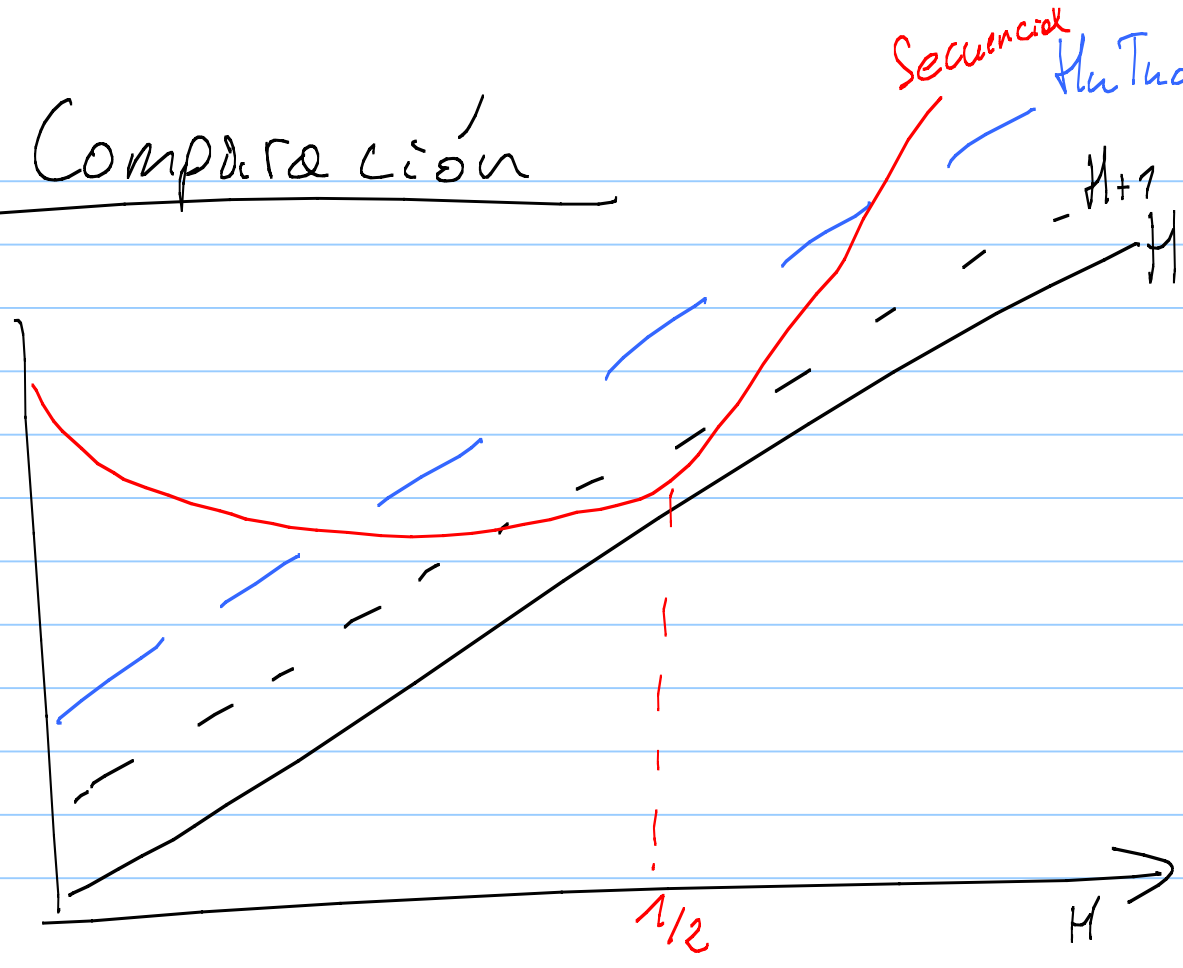
$p_1 \geq p_2 \geq \cdots \geq p_n$

Peor caso $p_i = \frac{1}{n} \rightsquigarrow \sum \frac{i}{n} = \frac{\frac{\sum i}{n(n+1)}}{2}$

Mejor caso $\begin{cases} p_i = \frac{1}{2^i} & i \in [n-1] \\ p_{n-1} = \frac{1}{2^{n-1}} \end{cases}$

$$\sum_{i=1}^{n-1} i\, p_i \leq \sum_{i=1}^{\infty} \frac{i}{2^i} = 1$$

4) Comparación



(5) Conclusión

1_ Un algoritmo de busqueda es suggerando un codigo. Esta relacion es importante para obtener cotas inferiores de complejidad

2_ A veces, un codigo puede suggerir un algoritmo tambien

Tarea:

Buscar la solución de $\sum \frac{i}{2^i}$