Problema

27.10.2011

- Perché è impossibile un algoritmo con fattore di conversione $\frac{6}{5}$

Vogliamo rispondere alla domanda introdotta nella lezione del 13.10.2011 a proposito di algoritmi che convertano stringhe finite di 0 e 1 in stringhe finite che non contengono 2 zeri consecutivi.

Fissiamo un intero $m \geq 1$. Chiamiamo S_m l'insieme delle stringhe lunghe al più m, e V_m l'insieme delle stringhe valide lunghe al più m.
Un algoritmo che trasformi una stringa lunga al più m in una stringa valida lunga al più m definisce una funzione

$$f : S_m \longrightarrow V_m$$

che deve essere <u>iniettiva</u>: altrimenti 2 stringhe diverse sarebbero codificate nella stessa stringa valida e non sarebbe possibile tramite la codifica inversa recuperare la stringa di partenza. Quindi è necessario che

$$|S_m| \leq |V_m|$$

dove $|A|$ indica il numero degli elementi dell'insieme A.
Sappiamo che:

$$|S_m| = \sum_{k=1}^{m} 2^k = 2\sum_{k=1}^{m-1} 2^k = 2(2^m - 1)$$

$$|V_m| = \sum_{k=1}^{m} v_k = \sum_{k=1}^{m} (a\alpha^k + b\beta^k)$$

dove $\alpha = \frac{1+\sqrt{5}}{2}$, $\beta = \frac{1-\sqrt{5}}{2}$, $a = \frac{5+3\sqrt{5}}{10}$, $b = \frac{5-3\sqrt{5}}{10}$. Quindi

$$|V_m| = a\alpha \frac{\alpha^m - 1}{\alpha - 1} + b\beta \frac{1-\beta^m}{1-\beta}$$

Diciamo che un algoritmo di conversione ha rendimento $\leq \gamma$ se manda ogni elemento di S_m in un elemento di $V_{\lfloor \gamma m \rfloor}$ per m sufficientemente grande (abbiamo aggiunto la parte intera poiché γm non è necessariamente un intero). Deve quindi risultare

$$\frac{|V_{\lfloor \gamma m \rfloor}|}{|S_m|} \geq 1 \quad \text{definitivemente}$$

$$\frac{|V_{\lfloor \gamma m \rfloor}|}{|S_m|} = \frac{1}{2(2^m - 1)} \left(a\alpha \frac{\alpha^{\lfloor \gamma m \rfloor} - 1}{\alpha - 1} + b\beta \frac{1 - \beta^{\lfloor \gamma m \rfloor}}{1 - \beta} \right)$$

Tenendo conto del fatto che $\alpha > 1$ e $|\beta| < 1$ si ha

$$\frac{|V_{\lfloor \gamma m \rfloor}|}{|S_m|} = \frac{a\alpha}{2(\alpha - 1)} \frac{\alpha^{\lfloor \gamma m \rfloor}}{2^m} + \varepsilon_m \quad \text{dove } \varepsilon_m \to 0$$

Stimando $\left(\frac{\alpha^\gamma}{2}\right)^m = \frac{\alpha^{\gamma m}}{2^m} \leq \frac{\alpha^{\lfloor \gamma m \rfloor}}{2^m} \leq \frac{\alpha^{\gamma m + 1}}{2^m} = \alpha \cdot \left(\frac{\alpha^\gamma}{2}\right)^m$

vediamo che se $\frac{\alpha^\gamma}{2} < 1$ allora $\frac{|V_{\lfloor \gamma m \rfloor}|}{|S_m|} \to 0$, contraddicendo il fatto che questo rapporto deve essere ≥ 1 def.

Deduciamo che

$$\frac{\alpha^\gamma}{2} \geq 1$$

ossia $\alpha^\gamma \geq 2 \iff \gamma \log_2 \alpha \geq 1 \iff \gamma \geq \frac{1}{\log_2 \alpha}$

Con $\alpha = \frac{1 + \sqrt{5}}{2}$ troviamo $\gamma \geq \frac{1}{\log_2 \alpha} \simeq 1,4404$

Quindi il rendimento $\frac{6}{5} = 1,2$ non è realizzabile

Algoritmo ottimale

Consideriamo la successione di Fibonacci

$$\begin{cases} F_{n+2} = F_n + F_{n+1} & \forall n \geq 0 \\ F_0 = 1 \ , \ F_1 = 1 \end{cases}$$

$(F_n) = (1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \dots)$

Fatto Ogni numero intero $n \geq 1$ è somma di numeri di Fibonacci distinti e mai consecutivi

Per esempio: $30 = 21 + 9 = 21 + 8 + 1$

$88 = 55 + 33 = 55 + 21 + 11 = 55 + 21 + 8 + 3$

In generale, dato $n \geq 1$ sia F_k il più grande numero di Fibonacci $\leq n$, ossia

$$F_k \leq n < F_{k+1}$$

Se $F_k = n$ abbiamo concluso, altrimenti consideriamo il numero $n - F_k$. Risulta

$$n - F_k < F_{k+1} - F_k = F_{k-1} + F_k - F_k = F_{k-1}$$

quindi $n - F_k$ è strettamente più piccolo del $(k-1)$-esimo numero di Fibonacci. Perciò, se F_h è il più grande numero di Fibonacci $\leq n - F_k$, risulta $h < k-1$.

Possiamo iterare il procedimento, che ci porta a scrivere n come somma di numeri di Fibonacci distinti e non consecutivi.

Ecco allora un possibile algoritmo di codifica:

$S = (s_1 s_2 \dots s_m)$, $s_j \in \{0,1\}$, stringa da inviare

$\downarrow$

$N = \sum_{j=1}^{m} s_j 2^{j-1} \in \{0, 1, 2, \dots, 2^m - 1\}$

$\downarrow$

$N = \sum_{j=1}^{m} t_j F_j$, $t_j \in \{0,1\}$ dove 2 t_j consecutivi non sono mai entrambi 1

$\downarrow$

$t = (t_1 t_2 \dots t_m)$

$\downarrow$

$\bar{t} = ((1-t_1)(1-t_2) \dots (1-t_m))$ non ha 2 zeri consecutivi ed è la stringa che viene trasmessa

Esercizio Descrivere la procedura di decodifica, che a $\bar{t}$ associa s.

Mostrare che $\forall \gamma > \frac{1}{\log_2 \alpha}$ questo algoritmo ha rendimento $\leq \gamma$

Esercizio Determinare, se esiste, il limite di $\frac{u_{n+1}}{u_n}$, dove (u_n) è definita da

$$\begin{cases} u_{n+2} = 4(u_{n+1} - u_n) \\ u_0 = 3 \quad , \quad u_1 = 8 \end{cases}$$

Cerco soluzione dell'equazione ricorsiva della forma $u_n = t^n$: $t^{n+2} = 4(t^{n+1} - t^n) \iff t^2 - 4t + 4 = 0$

$\iff t = 2$

Verifico che anche $u_n = n \cdot 2^n$ è soluzione:

$u_{n+2} = (n+2) 2^{n+2}$, $4(u_{n+1} - u_n) = 4((n+1)2^{n+1} - n 2^n)$

$= 4(2n+2-n) 2^n = 4(n+2)2^n = (n+2)2^{n+2}$

Quindi la combinazione lineare

$$u_n = a 2^n + b n 2^n$$

è soluzione. Imponendo $u_0 = 3$, $u_1 = 8$ si trova

$$\begin{cases} a = 3 \\ 2a + 2b = 8 \end{cases} \iff \begin{cases} a = 3 \\ b = 1 \end{cases}$$

Quindi $u_n = 3 \cdot 2^n + n 2^n$ Allora

$$\frac{u_{n+1}}{u_n} = \frac{3 \cdot 2^{n+1} + (n+1)2^{n+1}}{3 \cdot 2^n + n 2^n} = \frac{6 + 2(n+1)}{3+n} = \frac{2n+8}{n+3}$$

$$= \frac{2 + 8/n}{1 + 3/n} \longrightarrow 2 \quad \text{per } n \to \infty$$

Si sarebbe anche potuto procedere così: definiamo $r_n = \frac{u_{n+1}}{u_n}$. Allora $r_{n+1} = \frac{u_{n+2}}{u_{n+1}} = \frac{4(u_{n+1} - u_n)}{u_{n+1}} = 4 - \frac{4}{r_n}$

Quindi r_n è determinato dalla ricorsione $\begin{cases} r_{n+1} = 4 - \frac{4}{r_n} \\ r_0 = \frac{u_1}{u_0} = \frac{8}{3} \end{cases}$

e si può studiare il comportamento al limite di (r_n) con il metodo già visto (si noti che sapendo che r_n converge, il limite deve essere una soluzione di $L = 4 - \frac{4}{L}$, ossia $L = 2$).