

CRITOGRAFIA

lezione del 27 settembre -

mercoledí ore 16:15



Sequenze casuali in CRIPTOGRAFIA

- 1) Per alimentare algoritmi randominetti
- 2) Per generare le chiavi segrete dei cefrari.

$$h_1 = 111111\ldots 1 \quad |h_1|=20$$

$$h_2 = 01100101110100\ldots \quad |h_2|=20$$

Prob di ottenere h_1 $\left(\frac{1}{2}\right)^{20}$
(lanci di monete)

Prob di ottenere h_2 $\left(\frac{1}{2}\right)^{20}$

↳ Significato algoritmico della casualità

Teoria algorithmica dell'informazione

(≈ 1960)
kolmogorov

Idee:

Una sequenza binaria h è CASUALE se non ammette un algoritmo di generazione A la cui rappresentazione binaria sia più corta della sequenza h

$h = \text{seguenza di } n \text{ bit}$

$\rightarrow A_h = \langle \text{genera } n \text{ bit} \rangle$

$$|A_h| = n$$

Proponiamo di generare h :

for ($i=0$; $i < n$; $i++$)
 print '1'

$$|A_h| = \text{cost} + \Theta(\log n)$$

no discosto con $\Theta(\log n)$ bit con sequenza di n bit

~~se~~

Se h ci appare corrente, o non presente
esistente neppure; l'algoritmo di generazione
lo contiene al suo interno, e le restituisce
in output

$$h_2 = 1001111011 - \dots, 01 \dots$$

$\underbrace{\hspace{10em}}$
 n

P_2 = Programma per h_2 :

print '10011... - -'

$$|P_2| = \text{cost} + \Theta(n)$$

FORMULAZIONE MATEMATICA

- obiettivo: rendere le definizioni di consueto indipendente dal sistema di calcolo adottato
- i sistemi di calcolo sono numerabili (dobbiamo essere in grado di descriverli)

$S_1, S_2, \dots, S_i, \dots$

DEF Complessità di Kolmogorov di h in S_i :

$$k_{S_i}(h) = \min \{ |p| \mid S_i(p) = h \}$$

p programma per S_i , che genera h

$$S_i(p) = h$$

Tra i

Tra i vari sistemi di calcolo, \exists almeno un sistema di calcolo

S_u universale

(in grado di simulare gli altri sistemi di calcolo)

P programma di genere h in S_i :

$$S_i(p) = h$$

$q = \langle i, p \rangle \rightarrow$ programma di genere h in S_u :

$$S_u(q) = S_u(\langle i, p \rangle) = S_i(p) = h$$

$q = \langle i, p \rangle$ genera h in S_u

$$q = \langle i, p \rangle$$

$$|q| = |i| + |p| = \Theta(\log i) + |p|$$

α ≈ 1/2
per scavallo

non dipende da h

$$k_{\text{Su}}(h) ?$$

$$k_i + h$$

$$k_{\text{Su}}(h)$$

$$k_{\text{Si}}(h)$$

$$k_{\text{Su}}(h) \leq k_{\text{Si}}(h) + c$$

non dipende da h

Def.

La complessità secondo Kolmogorov di una sequenza
 h è

$$k(h) = k_{\text{Su}}(h)$$

(infatti, \nexists si, la complessità $k_{\text{Si}}(h)$
è $\geq k_{\text{Su}}(h)$, o meno di una costante)

Def. di casualità secondo Kolmogorov

Una sequenza h è casuale se

$$k(h) \geq |h| - \lceil \log_2 |h| \rceil$$

ESISTENZA di SEQUENZE CASUALI

f_n

Consideriamo sequenze binarie di lunghezza n

$$S = \# \text{ di sequenze binarie lunghe } n = 2^n$$

$$T = \# \text{ di sequenze binarie di lunghezza } n$$

NON casuali

Per dimostrare l'esistenza di seq. Casuali,
devo mostrare che $T < S$

—

Le T sequenze NON casuali sono generate
da prefissi di lunghezza $< n - \lceil \log_2 n \rceil$

$N = \# \text{ sequenze binarie di lunghezza } < n - \lceil \log_2 n \rceil$

(tra queste ci saranno anche le sequenze che
~~diff.~~ codificano i numeri di generazione
 delle T sequenze non corrispondenti)

$$N = \sum_{i=0}^{n - \lceil \log_2 n \rceil - 1} 2^i = \frac{2^{n - \lceil \log_2 n \rceil} - 1}{2 - 1} = \underline{\underline{2^{n - \lceil \log_2 n \rceil} - 1}}$$

$$T \leq N < S \Rightarrow T < S$$



$$\lim_{n \rightarrow \infty} \frac{T}{S} = \lim_{n \rightarrow \infty} \frac{2^{n - \lceil \log_2 n \rceil} - 1}{2^n} = \lim_{n \rightarrow \infty} \left(\frac{1}{2^{\lceil \log_2 n \rceil}} - \frac{1}{2^n} \right) = 0$$

→ LE SEQUENZE CASUALI TOLSTOJO SONO MOLTI ENORMEMENTE
PIÙ NUMEROSE DELLE SEQUENZE NON CASUALI

→ IL PROBLEMA DI STABILIRE SE UNA SEQUENZA È
ARBITRARIA E' CASUALE E' INDECIDIBILE

Dim Per assurdo, esiste un algoritmo

$$\text{Random}(h) = \begin{cases} 1 & h \text{ casuale} \\ 0 & \text{o/w} \end{cases}$$

Così nasce l'algoritmo PARADASSO

Paradosso

fr (binary $h \leftarrow 1 \text{ to } \infty$) ~~for~~)

// genera le
// sequenze binarie
// in ordine di
// lunghezza crescente

$$\text{if } (|h| - \lceil \log_2 |h| \rceil) > |P| \quad \textcircled{1}$$

&

Random(h) = 1) $\textcircled{2}$ return h

}

risolve il primo ~~eq.~~ caso
di lunghezza $|h| + c$
 $|h| - \lceil \log_2 |h| \rceil > |P|$

$$|P| = |\text{Paradosso}| + |\text{Random}|$$

\hookrightarrow è una costante indipendente da h

h occorre nel programma paradosso solo come nome di variabile

1) $\cancel{h} \quad [h] - \lceil \log_2 h \rceil \geq \underline{|P|}$



P é um programa de gerar h bres
 $\Leftrightarrow h$ non é constante

2) $\text{Random}(h) \neq \pm 1$

$\Leftrightarrow h$ é ~~non~~ constante (condição 1)

AND

h é constante (condição 2)

