

PROBLEMI NP-COMPLETI

Un problema Π si dice NP-completo se

$$\Pi \in NP, \forall \Pi' \in NP. \Pi' \leq_p \Pi$$

Se $\Pi \in P$ allora $P = NP$

Per dimostrare che un problema appartiene a NP

è sufficiente eribire un certificato polinomiale

TEOREMA di COOK SAT è NP-completo

Dunque, si dimostra che un problema α è NP completo

$$\alpha \in NP, SAT \leq_p \alpha$$

Oggi. $SAT \leq_p CLIQUE \Rightarrow CLIQUE$ è NP completo

$CLIQUE \leq_p SAT \Leftarrow SAT$ è NP completo

SAT e CLIQUE sono problemi equivalenti

(nb nella ricerca della soluzione esatta)

PROBLEMI Co-P, Co-NP

Un problema complementare accetta le istante rifiutate da Π

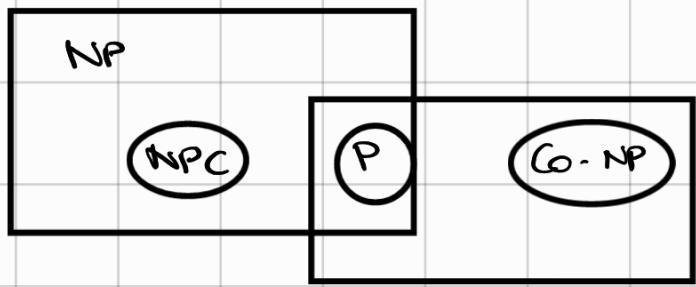
La classe Co-P è la classe dei problemi decisionali Π'

per cui $Co\Pi' \in P$, vale da $P = Co-P$

Co-NP

Co- $\Pi' \in NP$

Si congettura che $NP \neq Co-NP$
Se è vero allora $P \neq NP$



SEQUENZE CASUALI

In crittografia si vengono creare per:

- ① implementare algoritmi randomitati;
- ② generare chiavi segrete dei cifrari.

Esempio: $|h_1|=20 = |h_2|$

$$h_1 = 1 \dots 1$$

$$h_2 = 0110 \dots 1011 \dots$$

Supponendo che $p(0) = p(1) = \frac{1}{2}$,

$$p(h_1) = p(h_2) = \left(\frac{1}{2}\right)^{20}$$

Teoria algorithmica dell'informazione - Kolmogorov

Idea: Una sequenza binaria h è calcolata se non esiste un algoritmo

A di generazione la cui rappresentazione binaria sia più corta di h

Esempio: h = sequenza 1: n-1

A_h = «genera n volte 1»

$$|A_h|=n$$

$$|A_h| = k + \Theta(\log n)$$

Se h non presenta evidenti regolarità allora A deve contenere al suo interno a restituirla in output

$$\Rightarrow |A_1| = k + \Theta(n)$$

FORMALIZZAZIONE MATEMATICA

Ω sistemi di calcolo sui numeri dati (dobbiamo essere in grado di definirli)

Definizione Complessità di Kolmogorov di h in S_i

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

Programma i di S_i che genera h

Dalla teoria dei sistemi di calcolo si conosce un sistema S_u detto "universale" (è in grado di simulare gli altri sistemi di calcolo)

Esempio: p programma che genera $h \in S$:

$$S_i(p) = h$$

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

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

non dipende da h

Cerchiamo di valutare la complessità di Kolmogorov:

$$K_{S_u}(h) \quad ? \quad K_{S_i}(h)$$

$\forall i, \forall h \quad K_{S_u}(h) \leq K_{S_i}(h) + c$ con c che non dipende da h

L'è il limite inferiore

La complessità di Kolmogorov di una sequenza h

$$K(h) := K_{S_u}(h)$$

infatti $\forall S_i \quad K_{S_i}(h) \geq K_{S_u}(h)$ a meno di una costante

CASUALITÀ di KOLMOGOROV

Una sequenza h è casuale se $K(h) \geq |h| - \lceil \log_2(|h|) \rceil$

ESISTENZA di SEQUenze CASUALI

Si considera sequenza h binaria f.c. $|h| = n$

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

$$T = \#\{ \text{sequenze binarie lunghe } n \text{ non casuali} \}$$

Si deve dimostrare che $T < S'$

Per definizione:

le T sequenze sono generate da programmi lunghi al più $n - \lceil \log_2(n) \rceil$

$$N' = \#\{ \text{sequenze binarie lunghe al più } n - \lceil \log_2(n) \rceil \}$$

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

Risulta dunque:

$$T \leq N' < S'$$

$$\Rightarrow T < S'$$

\Leftrightarrow le sequenze casuali sono in

numero minore rispetto a tutte le

sequenze

$$\Rightarrow \Gamma_h$$

$$\text{ess. } \lim_{n \rightarrow \infty} \frac{T}{S'} = \lim_{n \rightarrow \infty} \frac{2^{\frac{n - \lceil \log_2(n) \rceil}{1}} - 1}{2^n} = 0.$$

Il problema di decidere se una sequenza è casuale secondo Kolmogorov è indecidibile.

Dico: Supponiamo per dimostrazione che esiste un algoritmo

$$\text{RANDOM}(h) = \begin{cases} 1 & \text{se } h \text{ è casuale} \\ 0 & \text{alt} \end{cases}$$

Costruiamo l'algoritmo PARADOSSO

PARADOSSO:

for binary $h \leftarrow 1$ to ∞ do: // genera le sequenze

// binarie in ordine di lunghezza

// crescente

if $(|h| - \lceil \log_2(h) \rceil) > |P|$

else $\text{RANDOM}(h) == 1 \rightarrow$ return h

done.

PARADOSSO restituisce la prima sequenza casuale di lunghezza

h tale che $|h| - \lceil \log_2(h) \rceil > |P|$

$|P| := |\text{PARADOSSO}| + |\text{RANDOM}|$

E' una costante, non dipende da h

(h occorre come solo nome di variabile in PARADOSSO)

① $|h| - \lceil \log_2(h) \rceil > |P|$ | E' sicuramente verificabile, ved. dim. prec.

P e' un programma breve, genera $h \Leftrightarrow h$ non e' casuale

② $\text{RANDOM}(h) == 1$ | E' sicuramente verificabile $\Leftrightarrow h$ e' casuale

ASSURDO $\Rightarrow \neg h$