NP

La domanda fondamentale: SAT $\in$ P ?
3SAT $\in$ P ? Perché è così importante:
NP-completezza, ovvero 3SAT $\in$ P se e solo se P = NP.
Intuizione per NP: Tutti problemi hanno la caratteristica di avere un numero esponenziale di soluzioni, ma controllare se una sol. è corretta è facile.

Esempi:

- 3COL. Il numero di 3-colorazioni è esponenziale in n, ma controllare se una 3-colorazione è corretta richiede tempo polinomiale.
- 3SAT: stesso discorso.
- PATH, 2SAT anche sono in NP e abbiamo visto infatti anche in P!

In generale, per i problemi in NP

- le soluzioni candidate sono codifica.

bile con stringhe di lunghezza poly.

- Verificare una sol. è possibile in tempo polinomiale.

DEF (Verificatore) Una TM V è un VERIFICATORE per linguaggio L se:

- V ha input $\langle x, y \rangle$.
- $\forall x$: $x \in L$ sse $\exists y$ t.c.

$$V(\langle x, y \rangle) = Acc.$$

Nota: questo significa:

- YES CASE: $x \in L \Rightarrow \exists y$ t.c. $V(\langle x, y \rangle) = ACC$

- NO CASE: $x \notin L \Rightarrow \forall y \; V(\langle x, y \rangle) = REJ$

Nota: V ha tempo di esecuzione polinomiale se il suo tempo di esecuzione è $O(|x|^K)$ per $K \in \mathbb{N}$.

Conseguenza: $|y| = poly(n)$ perché

V deve almeno leggere y.

DEF (Verifier-NP). NP è l'insieme di linguaggi L t.c. L ha un verificatore di tempo polinomiale.

Vediamo alcuni esempi.

THM $3COL \in NP$.

Dim. Basta considerare il seguente verificatore V:

- Su input $< x, y >$ dove $x = G$

un grafo.

- Interpreta $y = < c_1, c_2, \dots, c_n >$ dove $n = \#$ vertici in G e $c_i \in \{R, Y, B\}$

- $\forall (i, j) \in E$ RIFIUTA sse $c_i = c_j$. (colorazione del grafo G)

- Se non ho mai rifiutato, ACCETTO.

Innanzitutto, V ha tempo polinomiale in $|x|$. Inoltre:

- YES CASE: $G \in$ 3COL, allora $\exists$

Made with Goodnotes

una 3-colorazione $y=(c_1, \ldots, c_n)$ t.c. $\forall (v,i) \in E$ allora $c_v \neq c_i$. Allora $V(<x, y>) = Acc$.

- NO CASE. Se $V(<x, y>) = Acc$ per qualche $y = (c_1, \ldots, c_n)$ allora y è una 3-colorazione. Ovvero $G \in 3COL$.

ESERCIZIO: $3SAT \in NP$.

Made with Goodnotes

Qual è la relazione tra P ed NP.

THM $P \subseteq NP \subseteq EXP$.

Ricordiamo che $P \neq EXP$ (T.H.T.)

Ma allora $P \neq NP$ o $NP \neq EXP$ o

Tutte e due.

DIM. Sia $L \in P$: $\exists$ M t.c. $L(M) = L$

e M ha tempo polinomiale. Basta

considerare il verificatore che :

- Su input $\langle x, y \rangle$

  Ignora y e ACCETTA se $M(x) = ACC.$

Altra inclusione: se $L \in NP$ esiste V verificatore di tempo polinomiale. Questo significa come già detto che $|y| = O(|x|^k)$ per $k \in \mathbb{N}$ e quindi esiste M che decide L in tempo esponenziale provando tutti gli y. $\boxtimes$

C'è un'altra definizione di NP:
Non-deterministic Polynomial Time.
Già conosciamo la macchina di Turing
non deterministica.
La complessità di tempo di una NTM
è il massimo tempo di esecuzione su
tutti i rami di computazione della NTM.

DEF NTIME$(f(n))$ è l'insieme di
linguaggi L t.c. $\exists$ NTM N per
cui $L(N) = L$ e N ha tempo $O(f(n))$.

DEF $NP = \bigcup_{k \in \mathbb{N}} NTIME(n^k)$

DEF $NEXP = \bigcup_{k \in \mathbb{N}} NTIME(2^{n^k})$

THM $3SAT \in NP$.

DIM. Considero la NTM N che su input $\langle x \rangle = \langle \phi \rangle$ una 3-SAT formula:

- Prova non-deterministicamente tutti i possibili assegnamenti $x_1, \dots, x_n$ e controlla $\phi(x_1 \dots x_n) = 1$

THM Verifier-NP = NP.

DIM. ($\Rightarrow$) Sia L con verificatore V di tempo polinomiale in $|x|$. Devo definire una NTM N t.c. $L(N) = L$.

Come sappiamo, $V(<x, y>)$ dove $|y| = O(|x|^k)$ ovvero $\leq c \cdot x^k$ per qualche costante.

- Individua non-det. y.
- Esegui det. $V(<x, y>)$ su ogni

ramo L ACCETTA sse $V(\langle x, y\rangle) = ACC$

Chiaramente:

$x \in L$ sse almeno un ramo accetta

sse $\exists\, y$ t.c. $V(\langle x, y\rangle) = ACC$

sse $N(x)$ accetta.

($\Leftarrow$) Sia $L \in NP$, ovvero $\exists$ NTM N t.c. $L(N) = L$ e N ha tempo polinomiale. Devo definire V verificatore per L di tempo polinomiale.

Idea: Il certificato y sarà l'insieme delle scelte non det. di N.
Questo è rappresentabile con una stringa di lunghezza polinomiale perché N ha tempo polinomiale.

- Su input $\langle x, y \rangle$
- Simula det. $N(x)$ seguendo le scelte non-det. contenute in y.

Chiaramente: $x \in L$ sse $N(x)$ accetta sse $\exists$ un y insieme di scelte non det. per un N accetta nel relativo ramo di computazione. Ma questo è vero sse $\exists\, y$ t.c. $V(<x, y>)$ = ACC.

NP - completezza. Abbiamo visto $\exists$ linguaggi $L \in NP$ (e.g. 3SAT) che non sembrano essere in P.

Come vedremo questi linguaggi sono NP-completi: $L \in P$ sse $P = NP$.

Come si fa vedere questo? Tramite le RIDUZIONI.

DEF Siano A, B linguaggi. Diremo che $A \leq_m^P B$ se $\exists$ poly-time $R : \Sigma^* \to \Sigma^*$ t.c.

$\forall x \in \Sigma^* : x \in A$ sse $R(x) \in B$.

Nota: È identico alla precedente def. di mapping reduction ma ora R è poly-time computable.