# ESERCIZI

*) Mostrare che se $P = NP$ allora $NP = NP$-complete (i.e. tutti i linguaggi in $NP$ − tranne $\emptyset$ e $\Sigma^*$ − sono $NP$-completi).

Ovviamente, $NP\text{-complete} \subseteq NP$. Quindi dobbiamo mostrare $NP \subseteq NP\text{-complete}$.
Sia $A \in NP$, devo far vedere $\forall\, L \in NP$ si ha $L \leq_m^p A$.
Siccome $A \neq \emptyset, \Sigma^*$ allora esistono

due istanze $x_{YES}$, $x_{NO}$ t.c. $x_{YES} \in A$
ma $x_{NO} \notin A$.

La riduzione $R$:

- Su input $x$, usa una TM determini-
  nistica per decidere $x \in L$. Questo è
  possibile perché $L \in NP = P$.
- Se la TM accetta, $R(x)$ ritorna $x_{YES}$
  e altrimenti $x_{NO}$.

Chiaramente:

$$x \in L \iff R(x) \in A$$

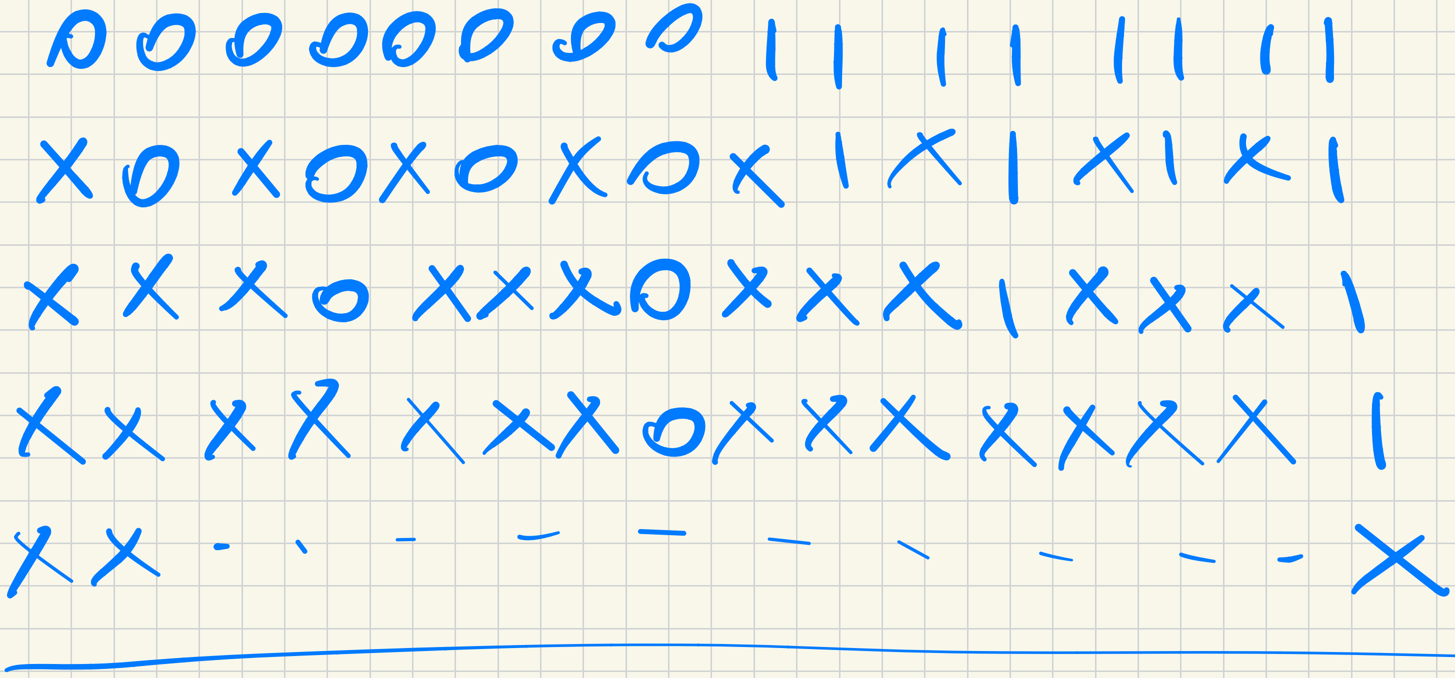
*) Sia $L = \{ 0^k 1^k : k \in \mathbb{N} \}$. Abbiamo visto a lezione che $L \in DTIME(n^2)$. Mostriamo che $L \in DTIME(n \log n)$.
L'idea: Cancellare più simboli allo stesso tempo.
Ad esempio:
- Controllo che input abbia la forma $0^i 1^j$. Se non fosse, rifiuto. Tempo: $O(n)$.
- Ripeto fino a che il nastro contiene almeno uno '0' oppure almeno un '1'.
  - Conto il numero di '0' più il numero

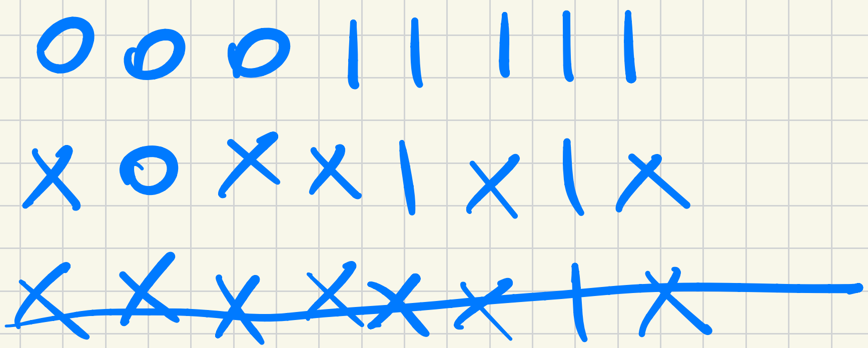
di '1' è ripetuto se fosse dispari.

- Scansiono il nastro sostituendo metà '0' con 'X' (uno sì e uno no e partire dal primo). Allo stesso modo sostituisco metà '1' con 'X'.

- Se rimangono solo 'X' accetto.

Complessità di tempo: Ad ogni iterazione (costo $O(n)$) l'input si dimezza
$\Rightarrow O(n \log n)$.





Riparte perché la

somma è DISPARI

*) Mostrare che se $P = NP$ allora esiste una TM polinomiale che dato $G = (V, E)$ grafo non orientato ritorna una clique massima contenuta nel grafo.

Abbiamo studiato come:

$$CLIQUE = \{\langle G, k \rangle : G = (V, E) \text{ con } k\text{-clique}\}$$

è NP-completo. In particolare, siccome $P = NP$ allora $CLIQUE \in P$. Uso la TM decisore deterministico per calcolare

la dimensione $K_{max}$ della massima clique contenuta in G.
A questo punto:
- Per ciascun nodo $x$ nel grafo definisco $H = (V', E')$ con $V' = V \setminus \{x\}$ e $E'$ di conseguenza.
- Lancio decisore per $\langle H, K_{max} \rangle \in$ CLIQUE.
  Se risponde lascio $x$ e passo al nodo successivo. Altrimenti passo al nodo successivo a partire da H.

Made with Goodnotes

- Quando non posso più rimuovere nodi restituisco in output l'insieme di nodi rimanenti.

Questi costituiscono una Kmax-clique.

*) Mostrare che NP è chiuso rispetto all'operazione "*".

Per definizione: $L \in NP \Rightarrow \exists$ TM $V$

t.c. $\forall x$, $x \in L$ sse $\exists y$ t.c.

$V(x, y) = ACC.$

Per decidere se $x \in L^* = \bigcup_K L^K$:

- Su input $w$, scegliamo non-det. una scomposizione $w = x_1 \dots x_K$

- Per ciascun $x_i$, scegliamo certificato

$y_i$ e controllo se $V(x_i, y_i) = Acc.$

- Se tutti i certificati accettano, allora accetto. Altrimenti rifiuto.

La correttezza è immediata:

- Se $x \in L^*$, allora $x = x_1 \dots x_k$ con $k \in \mathbb{N}$ t.c. $\forall i$ $x_i \in L$. Ma allora esiste un ramo di computazione in cui la NTM accetta.
- Se la NTM accetta, allora esiste uno ramo accettante. Ma questo è

vero sse $x = x_1 \dots x_k$ t.c.
$\forall n \quad x_n \in L$.

*) Si dimostra che $A_{DFA} = \{ \langle M, w \rangle :$
$M \in DFA$ e $M(w) = Acc \} \in L$.

Definisco una TM det. per decidere $A_{DFA}$:

- Su input $\langle M, w \rangle$ sul nastro di input
- Sul nastro di lavoro memorizzo: stato attuale di $M$ ($\log |Q| = O(1)$ spazio) e un contatore per ricordare il prossimo carattere di $w$ da leggere (spazio $O(\log n)$).
- Ad ogni passo scansiono input per det. prossimo stato e aggiorno nastro di lavoro riusando spazio.

*) Dimostrare che $DTIME(2^n) = DTIME(2^{n+1})$ e che $DTIME(2^n) \subsetneq DTIME(2^{2n})$.

La prima uguaglianza è banale: $2^{n+1} = 2 \cdot 2^n = O(2^n)$.

Per quanto riguarda la seconda posso usare il T.H.T.: siccome $2^{2n}$ è tempo costruttibile allora esiste $A \in DTIME(2^{2n})$ ma $A \notin DTIME(g(n))$ t.c. $g(n) = o(2^{2n} / \log 2^{2n})$ e $2^n = O(2^{2n} / \log 2^{2n})$

$\Rightarrow A \notin DTIME(2^n)$.

*) Dimostrare che $NTIME(n) \subsetneq PSPACE$.

Innanzitutto $NTIME(n) \subseteq NSPACE(n)$ perché il tempo limita lo spazio.

Poi $NSPACE(n) \subseteq SPACE(n^2)$ per il teorema di Savitch.

Per S. H. T. : $SPACE(n^2) \subsetneq SPACE(n^3)$

$\subseteq PSPACE$.

*) Si consideri la funzione

$$pad : \Sigma^* \times \mathbb{N} \to \Sigma^* \#^*$$

$$pad(x, l) = x \#^j \quad \text{dove}$$

$$j = \max(0, l - |x|)$$

(i) Mostrare che $\forall A$, $\forall k \in \mathbb{N}$ si ha $pad(A, n^k) \in P$ sse $A \in P$ dove $pad(A, n^k) = \{ pad(x, n^k) : x \in A \}$

(ii) Usare (i) per mostrare che $P \neq SPACE(n)$.

(i) Se $A \in P$, allora $\text{pad}(A, n^k) \in P$ perché posso decidere se $y \in \text{pad}(A, n^k)$ controllando che:

- $y$ abbia la forma $y = x\#^l$ e $|y| = n^k$.
- poi verifico che $x \in A$.

$\Rightarrow$ Complessità di tempo polinomiale in $|y|$ ovvero $\text{pad}(A, n^k) \in P$.

Sia invece $\text{pad}(A, n^k) \in P$. Per verificare re se $x \in A$ creo $y = x\#^\ell$ fino a lunghezza $|x|^k$ e poi vedo la TM det. che decide $y \in \text{pad}(A, n^k)$. Questo richiede tempo polinomiale.

(NN) Sia $P = SPACE(n)$. Per lo S.H.T. $\exists\, A \in SPACE(n^2)$ ma $A \notin SPACE(n)$. Mostriamo che $(N) \Rightarrow A \in SPACE(n)$, $\rightarrow\leftarrow$. Quindi $P \neq SPACE(n)$.

Questo segue dal fatto che

$$\text{pad}(A, n^2) \in SPACE(n)$$

questo perché ho spazio a sufficienza per eseguire la TM che decide A in spazio quadratico.

Ma allora $pad(A, n^2) \in SPACE(n) = P$

e quindi per (n) $A \in P = SPACE(n^2) \rightarrow\leftarrow$

*) Dimostrare che ogni linguaggio PSPACE-hard è anche NP-hard.

Ci ricordiamo: A è PSPACE-hard se $\forall$ $L \in PSPACE$ si ha $L \leq_m^p A$.

mentre A è NP-hard se $\forall$ $L \in NP$ si ha $L \leq_m^p A$.

Ma $NP \subseteq NPSPACE$ perché tempo limita spazio e $NPSPACE \subseteq PSPACE$ per Savitch.

$\Rightarrow$ $\forall$ $L \in NP \subseteq PSPACE$ si ha
$L \leq_m^p A$ perché A è PSPACE-hard

$\Rightarrow$ A è NP - hard!