Capitolo 9:

Trattabilità e intrattabilità dei problemi

La classe P è considerata tradizionalmente come una caratterizzazione formale dei problemi trattabili (risolti efficientemente dai calcolatori).

Tipici problemi di complessità polinomiale sono i problemi di ordinamento, di interrogazione di basi di dati, di ricerca di percorsi minimi in grafi, di programmazione lineare.

Ad oggi, non si sa se tutti i problemi in P hanno complessità equivalente. Ad esempio, alcuni problemi in P possono essere risolti in spazio logaritmico; ma tale fatto vale per tutti i problemi in P?.

Le riducibilità polinomiali non ci aiutano a caratterizzare la complessità di problemi in P (infatti, se $\mathcal{P}_1$ e $\mathcal{P}_2 \in P$, allora $\mathcal{P}_1 \leq_m^p \mathcal{P}_2$ e $\mathcal{P}_2 \leq_m^p \mathcal{P}_1$).

Definizione 1 *Dati due problemi $\mathcal{P}_1$ e $\mathcal{P}_2$, si dice che $\mathcal{P}_1$ è* log-space Karp-riducibile *a $\mathcal{P}_2$ ($\mathcal{P}_1 \leq_m^{logsp} \mathcal{P}_2$) se e solo se $\mathcal{P}_1$ e Karp-riducibile a $\mathcal{P}_2$ e la riduzione è un algoritmo calcolabile in spazio logaritmico.*

LOGSPACE è chiusa rispetto a $\leq_m^{logsp}$.

Se un algoritmo termina utilizzando spazio logaritmico, allora richiede tempo al piú polinomiale; quindi una riduzione log-space è anche polinomiale (ma non vale il contrario).

Esistono problemi "più difficili" (P-completi) rispetto alla riducibilità log-space? Esistono cioè problemi che "rappresentano" tutti i problemi in P??

Corollario immediato:
Se un problema P-completo è risolubile in spazio logaritmico, allora $P = LOGSPACE$.

P-completezza
La P-completezza di un problema di decisione $\mathcal{P} \in P$ è mostrata individuando qualche altro problema P-completo $\mathcal{P}_1$ tale che $\mathcal{P}_1$ è Karp-riducibile in spazio logaritmico a $\mathcal{P}$.

Infatti, si ha:
$\mathcal{P}_2 \leq_m^{logsp} \mathcal{P}_1$, per ogni $\mathcal{P}_2$, essendo $\mathcal{P}_1$ P-completo;
$\mathcal{P}_1 \leq_m^{logsp} \mathcal{P}$ (lo dimostriamo); allora, per transitivitá,
$\mathcal{P}_2 \leq_m^{logsp} \mathcal{P}$, per ogni $\mathcal{P}_2$, e quindi $\mathcal{P}$ è P-completo;

Cerchiamo un problema P-completo iniziale.

Definizione 2 *Un* circuito booleano *è un grafo orientato aciclico in cui ogni nodo ha $d_{in} \leq 2$ archi entranti. In particolare:*
1. i nodi con $d_{in} = 0$ sono detti nodi di input;
2. i nodi con $d_{in} = 1$ sono detti porte NOT;
3. i nodi con $d_{in} = 2$ sono detti porte AND o porte OR.
Tutti i nodi hanno $d_{out} > 0$, eccetto un solo nodo di output.

dimensione del circuito = numero di nodi;
profonditá = lunghezza massima fra input e output.

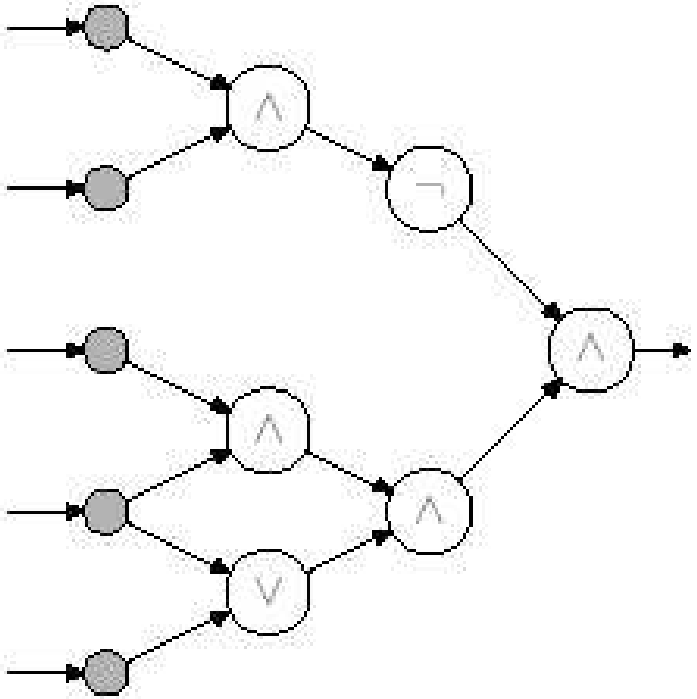
Sia $I(C)$ l'insieme di n nodi di input del circuito C; data una assegnazione di n valori booleani, il circuito calcola i valori booleani in uscita da ogni nodo, applicando su ogni porta l'operatore ad essa associato.

Il circuito calcola la funzione $f_C : \{0,1\}^n \longmapsto \{0,1\}$ tale che $f_C(b_0, \ldots, b_{n-1}) = 1$ sse il circuito restituisce il valore 1 per input $b_0, \ldots, b_{n-1}$.

Definizione 3 *Dato C, con $|I(C)| = n$, il linguaggio deciso da C è l'insieme delle stringhe $w \in \{0,1\}^n$ tale che $f_C(w) = 1$*

Esempio di circuito

5 input, 6 porte



VALORE CALCOLATO DA UN CIRCUITO
ISTANZA: circuito booleano C con nodi di input $|I(C)| = n$; stringa $w \in \{0,1\}^n$.
PREDICATO: si ha $f_C(w) = 1$?

Teorema 4 *Il problema VALORE CALCOLATO DA UN CIRCUITO è P-completo rispetto ad una riduzione log-space*

La classe NP

Molti problemi di decisione non hanno algoritmi polinomiali per la loro soluzione, ma non si sa se sono esponenziali.
Sono peró decidibili in tempo polinomiale da macchine di Turing non deterministiche (ad esempio, SODDISFACIBILITÁ).

Ad ogni problema di decisione $\mathcal{P}$ che chiede se l'istanza $x \in Y_{\mathcal{P}}$ o no, possiamo associare un problema di verifica il quale, data una possibile soluzione s, chiede se s è soluzione per x.

In generale, verificare una soluzione è piú facile che risolvere il problema di decisione nella sua forma originaria??

In particolare, verificare le soluzioni di un problema vuol dire risolvere il problema in modo non deterministico, nel modo seguente:

"una macchina non deterministica genera inizialmente tutte le stringhe che possono essere soluzione del problema, e verifica deterministicamente se ogni stringa è soluzione."

Se le soluzioni hanno dimensione polinomiale allora la fase di generazione non deterministica richiede tempo polinomiale; se la verifica è polinomiale, allora tutta la macchina opera in tempo polinomiale.

Un algoritmo non deterministico è tale che, oltre a tutte le istruzioni usuali del modello scelto, ha un comando di "guess", eseguito in un solo passo di computazione.

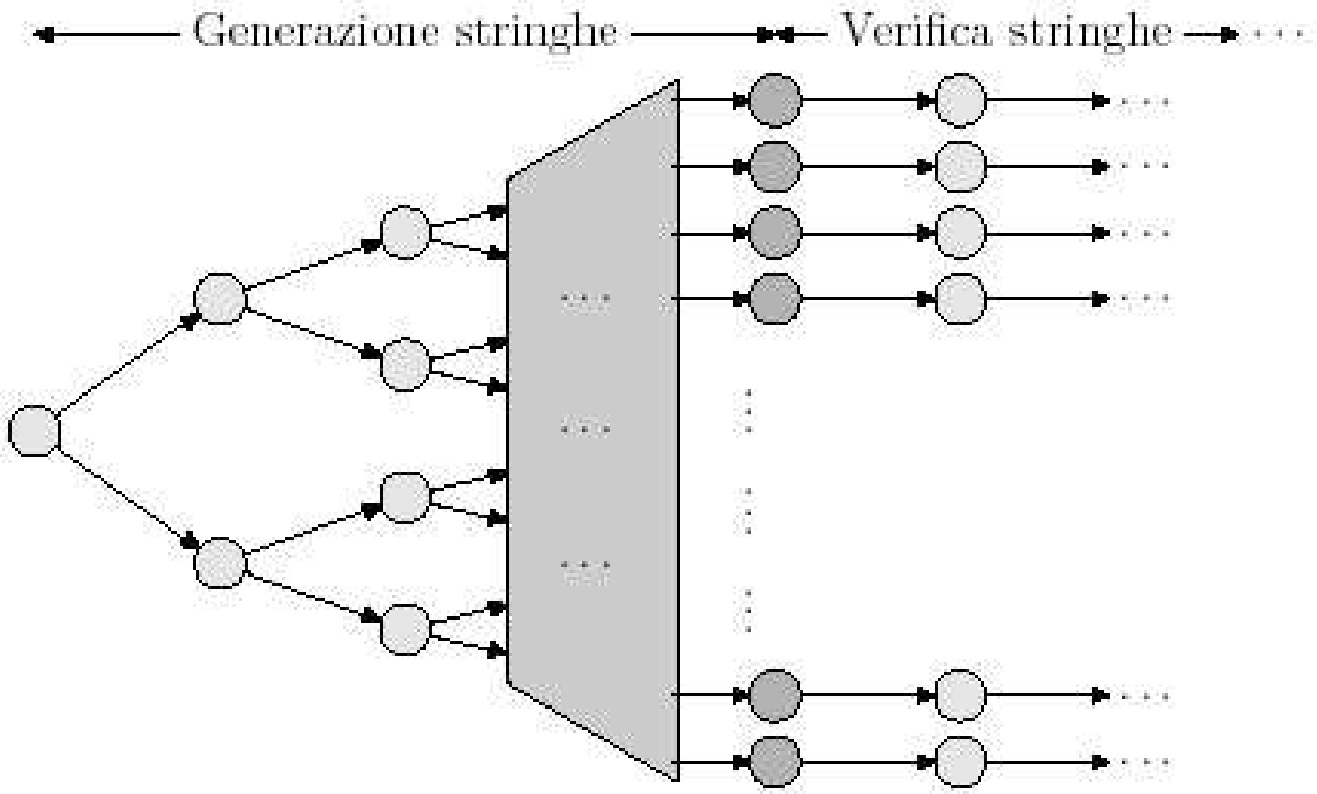
Quindi, puó "indovinare" come continuare la computazione nell'insieme finito delle possibili continuazioni della computazione svolta.

Definizione 5 *Dato un problema di decisione $\mathcal{P}$, un algoritmo non deterministico $\mathcal{A}$* risolve *$\mathcal{P}$ se per ogni istanza $x \in Y_{\mathcal{P}}$, esiste almeno una sequenza di guess tale che $\mathcal{A}$ restituisce il valore VERO, e per ogni $x \notin Y_{\mathcal{P}}$, non esiste alcuna sequenza di guess tale che $\mathcal{A}$ restituisce il valore VERO.*

La classe NP è ridefinita come la classe di tutti i problemi risolti da algoritmi deterministici che operano in tempo polinomiale a partire da guess di lunghezza polinomiale.

"$P = NP$?" vuol dire chiedersi se verificare efficientemente la soluzione di un problema di decisione è equivalente a risolvere efficientemente lo stesso problema.

Schema di soluzione di un problema in NP mediante macchina di Turing non deterministica



Algoritmo non deterministico per SODDISFACIBILITÀ

input X: insieme di variabili booleane, $(c_1, \ldots, c_n)$: insieme di clausole su V;
output boolean;
begin
  guess una assegnazione $f : X \mapsto \{\text{TRUE,FALSE}\}$;
  for each clausola $c_i \in \mathcal{F}$ **do**
    if non esiste alcun $t_{i,j} \in c_i$ soddisfatto da f
      then return NO;
  return VERO
end.

NP-completezza

Un problema $\mathcal{P} \in NP$ è NP-completo se ogni $\mathcal{P}_1 \in NP$ è Karp-riducibile polinomialmente a $\mathcal{P}$.

La NP-completezza di un problema di decisione $\mathcal{P} \in NP$ è mostrata individuando una riduzione polinomiale da qualche altro problema NP-completo $\mathcal{P}_1$.

Infatti, si ha:
$\mathcal{P}_2 \leq_m^p \mathcal{P}_1$, per ogni $\mathcal{P}_2$, essendo $\mathcal{P}_1$ NP-completo;
$\mathcal{P}_1 \leq_m^p \mathcal{P}$ (lo dimostriamo); allora, per transitivitá,
$\mathcal{P}_2 \leq_m^p \mathcal{P}$, per ogni $\mathcal{P}_2$, e quindi $\mathcal{P}$ è NP-completo;

Teorema 6 *(di Cook) Il problema SODDISFACIBILITÁ è NP-completo rispetto ad una riduzione polinomiale.*