teoria della complessità: la risorsa tempo

teoria della complessità

- ➔ obiettivo: classificare i problemi dal punto di vista delle risorse di calcolo che richiedono
- ➔ identificazione di classi di complessità
- ➔ risorsa: tempo impiegato per risolvere il problema
- ➔ modello di calcolo: macchina di Turing; ipotesi di notevole generalità

$t_A(n)$ tempo di esecuzione dell'algoritmo A sull'input x con $|x|=n$ nel caso peggiore

teoria della complessità - tipi di problemi

classificazione abbastanza generale:

- decisione
- ricerca
- enumerazione
- ottimizzazione

problemi di decisione

definiamo il problema di decisione P_D

insieme di istanze I_{PD}

predicato π: $I_{PD} \rightarrow \{T,F\}$ che determina la partizione $Y_{PD} \cup N_{PD}$ con $Y_{PD} \cap N_{PD} = \emptyset$

Y_{PD} istanze positive tali che $x \in Y_{PD} \Leftrightarrow \pi(x)=T$

N_{PD} istanze negative tali che $x \in N_{PD} \Leftrightarrow \pi(x)=F$

i problemi studiati di riconoscimento di un linguaggio L sono problemi di decisione con $I_{PD}=\Sigma^*$ e $Y_{PD}=L$; inoltre ad ogni P_D può essere associato un problema di riconoscimento di linguaggi equivalente

esempio: problema di decisione cricca o clique

istanza: grafo $G=(V,E)$, intero $K>0$

predicato: esiste $V' \subseteq V$ con $|V'| \geq K$ tale che per ogni $u,v \in V'$ $(u,v) \in E$?

problemi di ricerca

mentre per un problema di decisione si chiede se esiste una soluzione per l'istanza, in un problema di ricerca si chiede di determinare la soluzione;
definiamo un problema di ricerca P_R

- insieme di istanze I_{PR}
- insieme di soluzioni S_{PR}
- relazione di ammissibilità $R \subseteq I_{PR} \times S_{PR}$
- $Sol(x)=\{y \in S_{PR} | <x,y> \in R\}$ è l'insieme delle soluzioni ammissibili di x

ogni problema di ricerca P_R ha associato un problema di decisione P_D con $I_{PD}=I_{PR}$ e predicato π "esiste $y \in S_{PR}$ tale che $<x,y> \in R$?"

problemi di ricerca

esempio: problema di ricerca cricca o clique

istanza: grafo $G=(V,E)$, intero $K>0$

soluzione: $V' \subseteq V$

ammissibilità: $|V'| \geq K$ tale che per ogni $u,v \in V'$ $(u,v) \in E$

l'ammissibilità e' il predicato di decisione di cricca

problemi di enumerazione

i problemi di enumerazione sono collegati ai problemi di ricerca; definiamo un problema di enumerazione P_E

insieme di istanze I_{PE}

insieme di soluzioni S_{PE}

relazione di ammissibilità $R \subseteq I_{PE} \times S_{PE}$

un algoritmo per un problema di enumerazione determina il numero di soluzioni ammissibili per una data istanza

problemi di ottimizzazione

definiamo un problema di ottimizzazione P_O

- insieme di istanze I_{PO}
- insieme di soluzioni S_{PO}
- relazione di ammissibilità $R \subseteq I_{PO} \times S_{PO}$
- funzione $\mu: I_{PO} \times S_{PO} \rightarrow N$ di misura tale che per ogni $x \in I_{PO}$ $y \in S_{PO}$ $\mu(x,y)$ e' definita se e solo se $y \in Sol(x)$
- criterio di scelta: MIN o MAX

esempio: problema di ottimizzazione cricca o clique

istanza: grafo $G=(V,E)$
soluzione: $V' \subseteq V$
ammissibilità: $|V'| \geq K$ tale che per ogni $u,v \in V'$ $(u,v) \in E$
misura: $|V'|$
criterio: MAX

complessità e problemi di decisione su linguaggi

- dato un problema di decisione P_D possiamo codificare le istanze di I_{PD} in stringhe
- P_D può quindi essere interpretato come problema di discriminazione tra l' insieme di stringhe che codificano Y_{PD} e l'insieme di stringhe che codificano N_{PD}
- osservazione: possiamo affrontare un problema di decisione come problema di riconoscimento di un linguaggio

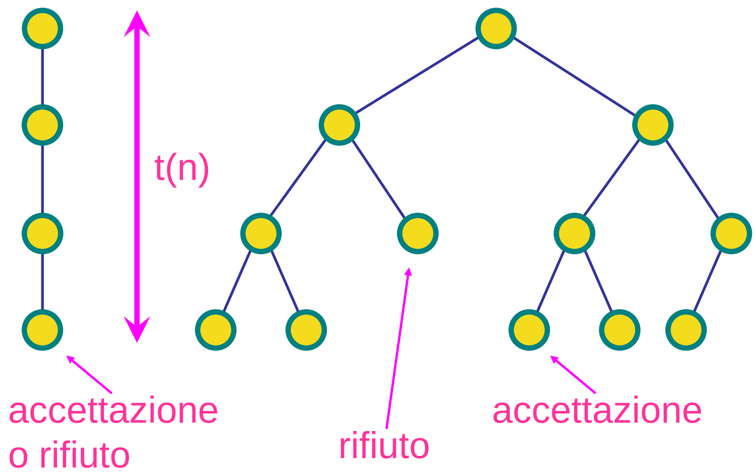
complessità e problemi di decisione su linguaggi

data una funzione $t:N\rightarrow N$ possiamo definire le seguenti classi di linguaggi con riferimento a MT ad un nastro

DTIME(t(n)) insieme dei linguaggi decisi da una MT deterministica in tempo O(t(n))

NTIME(t(n)) insieme dei linguaggi decisi da una MT non deterministica in tempo O(t(n))

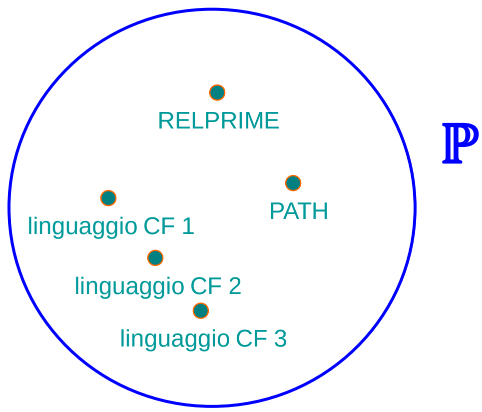
tempo deterministico e tempo non deterministico



la classe $\mathbb{P}$

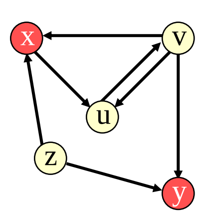
$$\mathbb{P} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(n^k)$$

esempi di problemi in $\mathbb{P}$



PATH

- istanza: un grafo G orientato e due suoi vertici x ed y
- predicato: esiste un cammino orientato in G da x ad y?



PATH

- osservazione
  - l'approccio brute-force non funziona
- algoritmo (MT)
  - visita in profondità con marcatura
- analisi
  - ogni arco è visitato un numero costante di volte
- domanda
  - per dimostrare l'appartenenza a $\mathbb{P}$ possiamo usare il rapporto con il problema dell'appartenenza ad un linguaggio CF?

RELPRIME

- istanza: X, Y interi positivi
- predicato: X e Y non hanno divisori interi comuni diversi da 1?

- esempio
  - 10, 21 SI
  - 10, 22 NO

RELPRIME

- osservazione
  - l'approccio brute-force non funziona
  - nella rappresentazione binaria un numero ha valore esponenziale nel numero di cifre usate per rappresentarlo
- algoritmo:
  - variante dell'algoritmo per l'MCD di Euclide

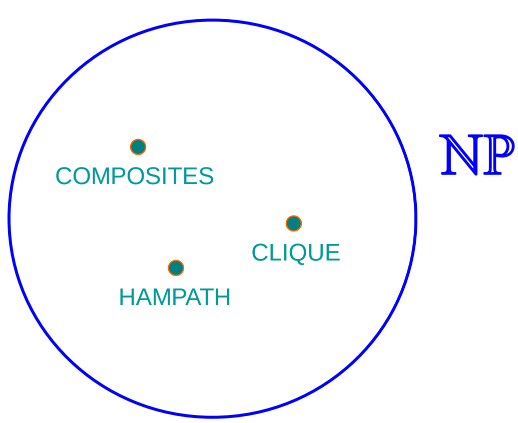
RELPRIME

- analisi
  - ogni passo dimezza il valore (almeno)

la classe $\mathbb{NP}$

$$\mathbb{NP} = \bigcup_{k=0}^{\infty} \mathrm{NTIME}(n^k)$$

esempi di problemi in $\mathbb{NP}$



CLIQUE

teorema: CLIQUE $\in \mathbb{NP}$

dimostrazione: istanza: grafo G=(V,E), intero K>0

possiamo costruire una MTND M che risolve CLIQUE in tempo polinomiale:

- facciamo generare ad M tutte i possibili sottoinsiemi V' di V
- ogni nodo dell'albero di computazione di M è associato alla scelta di appartenenza o non appartenenza di un arco a V'
- ogni cammino radice-foglia è di lunghezza polinomiale
- su ogni foglia verifichiamo in tempo polinomiale se il sottoinsieme V' prodotto è una clique

HAMPATH e verificabilità polinomiale

problema del cammino Hamiltoniano

- istanza: grafo orientato G=(V,E), vertici s e t
- predicato: G ha un cammino Hamiltoniano (passa una ed una sola volta su ogni vertice) da s a t?

teorema: HAMPATH $\in \mathbb{NP}$

dimostrazione: basta generare non deterministicamente tutti i sottoinsiemi di archi e per ciascuno di essi verificare se è un cammino Hamiltoniano

HAMPATH e verificabilità polinomiale

HAMPATH una caratteristica interessante che si chiama verificabilità polinomiale

se ci viene presentata una soluzione ad HAMPATH possiamo verificare in tempo polinomiale se la soluzione è corretta oppure no

COMPOSITES e verificabilità polinomiale

problema di stabilire se un numero è primo

- istanza: un numero intero positivo x
- predicato: esistono due numeri interi p,q>1 tali che x=pq?

- anche COMPOSITES ha la proprietà di verificabilità polinomiale
- effettivamente se qualcuno ci propone p e q come soluzione possiamo verificare facilmente se la soluzione è corretta

verificatori e certificati

- un verificatore per un linguaggio A è una MT V tale che A={w | V accetta <w,c> per qualche stringa c}
- c è il certificato o la prova dell'appartenenza di w ad A
- misuriamo il tempo impiegato da un verificatore per l'accettazione solo in funzione di w
- un verificatore impiega tempo polinomiale quando impiega tempo polinomiale in |w|

verificatori e certificati

- un linguaggio è verificabile in tempo polinomiale se ha un verificatore che impiega tempo polinomiale
- osservazione: se un linguaggio è verificabile in tempo polinomiale i certificati delle sue stringhe w hanno lunghezza polinomiale in |w|

certificati per HAMPATH e COMPOSITES

- per HAMPATH un certificato è un cammino Hamiltoniano da s a t
- per COMPOSITES è uno dei divisori di x

equivalenza tra verificabilità polinomiale ed appartenenza a NP

- teorema: un linguaggio appartiene a NP se e solo se ha un verificatore che impiega tempo polinomiale
- dimostrazione: sia V un verificatore che impiega tempo n^k per il linguaggio A

  quando riceviamo in input una stringa w di lunghezza n generiamo non deterministicamente tutte le stringhe c di lunghezza n^k e invochiamo V su <w,c>

  se V accetta accettiamo, altrimenti rifiutiamo

equivalenza tra verificabilità polinomiale ed appartenenza a NP

- dimostrazione: sia N una MTND che decide il linguaggio A in tempo polinomiale

  per ogni coppia di stringhe <w,c>:

  simuliamo il comportamento di N su w, usando c come selettore della scelta da fare ad ogni step (vedi teorema di corrispondenza tra MTND e MT)

  se una branca della computazione accetta accettiamo, altrimenti rifiutiamo

una definizione alternativa per NP

- a questo punto possiamo definire NP anche come la classe dei linguaggi che hanno un verificatore polinomiale

la classe co-NP

- classe dei linguaggi complemento dei linguaggi in NP
- non è chiaro quanto sia facile trovare un verificatore polinomiale per HAMPATH

riducibilità

ridurre un problema P1 ad un problema P2 riconduce il problema di risolvere P1 al problema della ricerca di un algoritmo per risolvere P2

se riusciamo quindi a risolvere P2 riusciamo, come effetto collaterale, a risolvere anche P1

pro-memoria: un problema di decisione P1 e' riducibile a un problema di decisione P2 (P1≤P2) se esiste un algoritmo (MT) R che trasforma ogni istanza $x \in I_{P1}$ di P1 in una istanza $y \in I_{P2}$ di P2 in modo tale che $x \in Y_{P1}$ se e solo se $y \in Y_{P2}$; R e' una riduzione da P1 a P2

osservazione: se esiste R da P1 a P2 ed esiste un algoritmo A2 per P2, allora possiamo costruire un algoritmo A1 per P1:

1. data $x \in I_{P1}$ applico R a x e ottengo $y \in I_{P2}$
2. applico A2 a y, se A2 restituisce Y allora restituisco Y altrimenti restituisco N

riducibilità

osservazione: la complessità computazionale di A1, costruito con A2 ed R, dipende dalla complessità di A2 e dalla complessità di R

P1 e' polinomialmente riducibile a P2

(P1 $\leq_p$ P2) se la riduzione R è un algoritmo con complessità polinomiale

osservazione: se P1 $\leq_p$ P2 e so risolvere in tempo polinomiale P2 allora so risolvere in tempo polinomiale P1

$$P1 \leq_p P2 \wedge P2 \in \mathbb{P} \Rightarrow P1 \in \mathbb{P}$$

completezza

la completezza è uno strumento per indagare le relazioni di inclusione fra classi di complessità

la usiamo per indagare il rapporto tra $\mathbb{P}$ ed $\mathbb{NP}$

un problema di decisione P e' $\mathbb{NP}$-completo se $P \in \mathbb{NP}$ e per ogni $P1 \in \mathbb{NP}$ $P1 \leq_p P$

completezza

richiamo: un problema di decisione P e' $\mathbb{NP}$-completo se P $\in \mathbb{NP}$ e per ogni P1 $\in \mathbb{NP}$ P1 $\leq_p \mathbb{P}$

osservazione: se riesco a dimostrare che un problema $\mathbb{NP}$-completo P appartiene a $\mathbb{P}$, cioe' e' risolvibile in tempo polinomiale da un algoritmo deterministico, allora ogni altro problema di $\mathbb{NP}$ e' risolvibile in tempo polinomiale da un algoritmo deterministico

nota: nessuno e' mai riuscito a dare per un problema $\mathbb{NP}$-completo un algoritmo deterministico polinomiale

nota: nessuno e' mai riuscito a dimostrare un lower bound non polinomiale per un problema di $\mathbb{NP}$

nota: non si sa se $\mathbb{P}=\mathbb{NP}$ o se $\mathbb{P}\neq\mathbb{NP}$

ancora sulla classe NP

classe dei problemi risolvibili in tempo polinomiale da un algoritmo (MT) non deterministico

esempio: problema della soddisfacibilità (SAT)

$X=\{x_1,\ldots,x_n\}$ insieme di variabili booleane

$T(X)=\{x_1,\ldots,x_n,\sim x_1,\ldots,\sim x_n\}$ insieme di termini

assegnamento f di verità su X f: $X\rightarrow\{true,false\}$ soddisfa x_i se $f(x_i)=true$ e $\sim x_i$ se $f(x_i)=false$

clausola disgiuntiva: insieme di termini $c=\{t_1,\ldots,t_k\}\subseteq T(X)$

c e' soddisfatta da f se esiste un $t_i \in c$ soddisfatto da f

esempio (SAT)

formula in forma normale congiuntiva (CNF): insieme di clausole $F=\{c_1,...,c_m\}$

F e' soddisfatta da f se ogni $c_i \in F$ e' soddisfatta da f

problema di decisione soddisfacibilità (SAT)

istanza: formula F in CNF su un insieme X di variabili booleane

predicato: esiste un assegnamento

$f:X \rightarrow \{true,false\}$ che soddisfa F?

la classe NP

teorema: SAT $\in \mathbb{NP}$

dimostrazione: possiamo costruire una MTND M che risolve SAT in tempo polinomiale:

facciamo generare ad M tutte i possibili assegnamenti sul X; questi sono in numero esponenziale

associamo ad ogni assegnamento un cammino radice-foglia dell'albero di computazione di M

ogni nodo dell'albero e' associato alla scelta vero/falso per una variabile

ogni cammino radice-foglia e' di lunghezza polinomiale

su ogni foglia verifichiamo in tempo polinomiale se l'assegnamento prodotto soddisfa F

la classe $\mathbb{NP}$

esempio: problema di decisione della programmazione lineare {0,1} (PL{0,1})

istanza: insieme di variabili $Z=\{z_1,\ldots,z_n\}$ con dominio in {0,1}; insieme I di disequazioni lineari su Z

predicato: esiste un assegnamento di valori alle variabili di Z che verifica tutte le disequazioni di I?

teorema: PL{0,1}$\in\mathbb{NP}$

dimostrazione: come per SAT

la classe $\mathbb{NP}$

teorema: $SAT \leq_p PL\{0,1\}$

dimostrazione: ogni istanza x_{SAT}=<X,F> di SAT può essere ridotta ad una istanza $x_{PL\{0,1\}}$=<Z,I> di PL{0,1}

se $\{t_{i1},\ldots,t_{ik}\}$ e' la i-esima clausola di F definiamo la i-esima disequazione di I $\tau_{i1}+\ldots+\tau_{ik}>0$

$\tau_{ij}=z_{ij}$ se $t_{ij}=x_{ij}$

$\tau_{ij}=1-z_{ij}$ se $t_{ij}=\sim x_{ij}$

ogni assegnamento di verità $f:X\rightarrow\{true,false\}$ soddisfa F se e solo se tutte le disequazioni di I sono verificate da un assegnamento $f':Z\rightarrow\{0,1\}$ tale che

$f'(z_i)=1$ se e solo se $f(x_i)$=vero

la riduzione e' calcolabile in tempo polinomiale

la classe $\mathbb{NP}$

esempio: problema di decisione insieme indipendente (INDSET)

istanza: grafo G=(V,E), intero K>0

predicato: esiste un sottoinsieme U di V ($U \subseteq V$) con $|U| \geq K$ tale che per ogni $u,v \in U$ $(u,v) \notin E$?

teorema: INDSET $\in$ NP

dimostrazione: generiamo con una MTND tutti i sottoinsiemi di V, uno per ogni cammino radice-foglia dell'albero di computazione

per ogni sottoinsieme U di V verifichiamo in tempo polinomiale: (1) che U e' un insieme indipendente; (2) che $|U| \geq k$

$\mathbb{NP}$-completezza

richiamo: un problema di decisione P e' $\mathbb{NP}$-completo se $P \in \mathbb{NP}$ e per ogni $P1 \in \mathbb{NP}$ $P1 \leq_p \mathbb{P}$

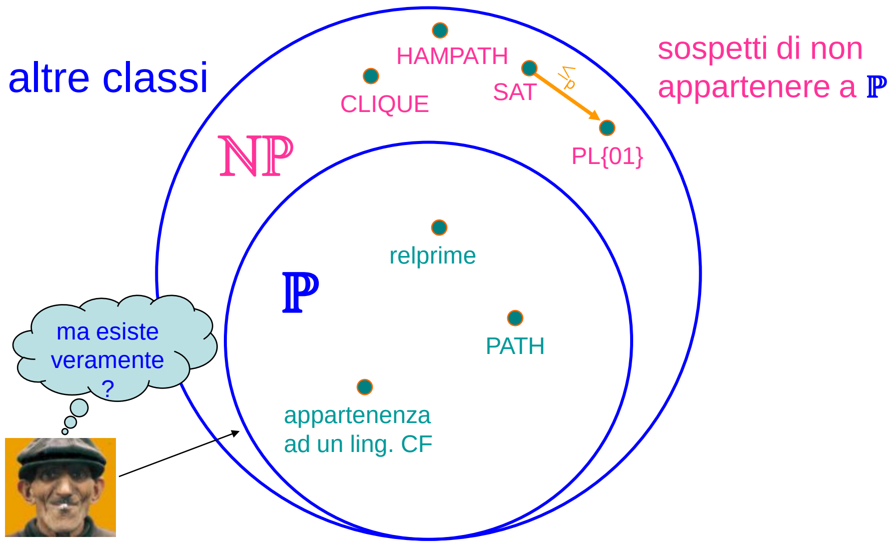
osservazione: per dimostrare che un problema P e' $\mathbb{NP}$-completo ci sono due possibili metodi

1. dimostrare che ogni problema P1 in $\mathbb{NP}$ è riducibile a P, oppure

2. trovare un problema P2 già noto per essere $\mathbb{NP}$-completo e dimostrare che $P2 \leq_p P$; infatti,
se $P2 \leq_p P$ e per ogni $P1 \in \mathbb{NP}$ $P1 \leq_p P2$ e $P2 \leq_p P$, allora per ogni $P1 \in \mathbb{NP}$ $P1 \leq_p P$

perche' il metodo 2 sia applicabile c'e' bisogno di almeno un problema $\mathbb{NP}$-completo di base

il punto della situazione



ℕℙ-completezza

teorema (di Cook): SAT e' NP-completo

dimostrazione: abbiamo già dimostrato che SAT∈NP, ora dimostriamo che per ogni P∈NP $P \leq_p SAT$

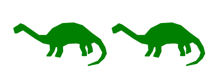
facciamo riferimento al linguaggio L associato a P e al linguaggio SAT associato a SAT

se P∈NP allora esiste una MTND che accetta L in tempo polinomiale

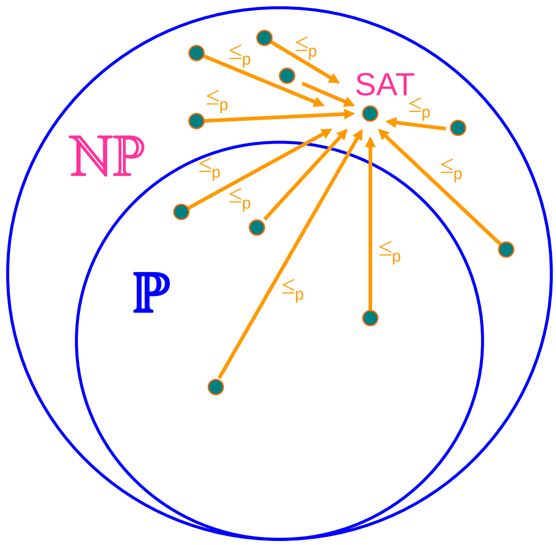
sia M una MTND che accetta ogni stringa x di L in tempo p(|x|) con p polinomio

dati M ed x costruiamo una formula w in CNF tale che w e' soddisfacibile se e solo se M accetta x, cioe' se e solo se x∈L

ipotesi semplificative: M con nastro semi-infinito con x nelle prime celle; esiste uno stato q_Y di accettazione; la computazione ha esattamente p(|x|) passi



teorema di cook



dimostrazione (SAT è ℕℙ-completo)

w è congiunzione di quattro formule

$$w = w_M \wedge w_I \wedge w_A \wedge w_T$$

w_M specifica le proprietà generali delle MT

w_I specifica la posizione di x sul nastro

w_A specifica la funzione di q_Y

w_T specifica la funzione di transizione di M

dimostrazione (SAT è $\mathbb{NP}$-completo)

variabili di w

Q(t,k), Q(t,k) e' true se e solo se M si trova nello stato q_k all'istante t

H(t,i), H(t,i) e' true se e solo se la testina di M si trova sulla cella i-esima del nastro all'istante t

C(t,i,h), C(t,i,h) e' true se e solo se la cella i-esima del nastro contiene all'istante t il simbolo σ_h

dimostrazione(SAT e' $\mathbb{NP}$-completo)

w_M è congiunzione di tre formule $w_M = w_{MS} \wedge w_{MH} \wedge w_{MC}$

w_{MS} specifica che ad ogni istante M si trova in uno stato

$$w_{MS} = \bigwedge_t \Big(\big(\bigvee_k Q(t,k) \big) \wedge \ldots \wedge \bigwedge_{k1 \neq k2} \big({\sim}Q(t,k1) \vee {\sim}Q(t,k2) \big) \Big)$$

w_{MH} specifica che ad ogni istante la testina si deve trovare su una cella del nastro

$$w_{MH} = \bigwedge_t \Big(\big(\bigvee_i H(t,i) \big) \wedge \ldots \wedge \bigwedge_{i1 \neq i2} \big({\sim}H(t,i1) \vee {\sim}H(t,i2) \big) \Big)$$

w_{MC} specifica che ogni cella contiene un carattere

considerando tutte le possibili combinazioni $|w_M| = O(p^3(|x|))$

dimostrazione(SAT e' $\mathbb{NP}$-completo)

esempio per w_M

se M ha tre stati q0, q1 e q2 e la computazione dura due mosse

$((Q(0,0) \vee Q(0,1) \vee Q(0,2)) \wedge$
$\wedge(\sim Q(0,0) \vee \sim Q(0,1)) \wedge$
$\wedge(\sim Q(0,0) \vee \sim Q(0,2)) \wedge$
$\wedge(\sim Q(0,1) \vee \sim Q(0,2)))$ t=0

$\wedge$

$\wedge$ t=1 $\wedge$ t=2

dimostrazione(SAT e' $\mathbb{NP}$-completo)

se $x=\sigma_{h0},\sigma_{h1},....,\sigma_{h(n-1)}$, e q_0 e' lo stato iniziale

$w_I = Q(0,0) \wedge H(0,0) \wedge$
$C(0,0,h_0) \wedge C(0,1,h_1) \wedge .... \wedge C(0,n-1,h_{n-1}) \wedge$
$C(0,n,0) \wedge C(0,n+1,0) \wedge .... \wedge C(0,p(|x|),0)$

$w_A = Q(p(|x|),k)$ con $q_Y=q_k$

w_T è congiunzione di due formule

$$w_T = w_{T1} \wedge w_{T2}$$

le formule che costruiamo non sono in CNF, ma possono essere trasformate in CNF con aumento lineare della lunghezza

dimostrazione (SAT e' $\mathbb{NP}$-completo)

w_{T1} specifica che per ogni t i contenuti del nastro agli istanti t e t+1 sono gli stessi, eccetto, eventualmente, per cio' che riguarda la cella su cui la testina e' posizionata all'istante t

w_{T2} codifica la funzione di transizione di M sotto forma di un insieme di regole e consta della congiunzione di $p^2(|x|)\ |Q|\ (|\Sigma|+1)$ formule

la formula w ha lunghezza $O(p^3(|x|))$ e puo' essere costruita in tempo proporzionale alla sua lunghezza

w e' soddisfacibile se e solo se M accetta la striga x in tempo p(|x|)

esempi di problemi $\mathbb{NP}$-completi

esempio: 3-soddisfacibilità (3-SAT)

istanza: formula F in CNF su un insieme X di variabili booleane con ogni clausola composta esattamente da 3 termini

predicato: esiste un assegnamento

f: $X\rightarrow\{true,false\}$ che soddisfa F?

riduzione: $SAT\leq_p 3\text{-}SAT$

esempio: insieme copertura (VERTEXCOVER)

istanza: grafo G=(V,E), intero K>0

predicato: esiste un sottoinsieme U di V ($U\subseteq V$) con $|U|\leq K$ tale che per ogni $(u,v)\in E$ almeno uno tra u e v $\in U$?

riduzione: $3\text{-}SAT\leq_p VERTEXCOVER$

Karp 1972

esempi di problemi $\mathbb{NP}$-completi

esempio: insieme dominante (DOMINATINGSET)

istanza: grafo G=(V,E), intero K>0

predicato: esiste un sottoinsieme U di V ($U \subseteq V$) con $|U| \leq K$ tale che per ogni $u \in V-U$ esiste un $v \in U$ tale che $(u,v) \in E$?

riduzione: VERTEXCOVER$\leq_p$DOMINATINGSET

Garey,Johnson 1979

esempio: k-colorabilità (CHROMATIC#)

istanza: grafo G=(V,E), intero K>0

predicato: esiste una funzione $f:V \rightarrow \{1,2,....,k\}$ tale che $f(u) \neq f(v)$ se $(u,v) \in E$?

riduzione: 3-SAT$\leq_p$CHROMATIC#

Karp 1972

nota: il problema e' polinomiale per k=2

esempi di problemi $\mathbb{NP}$-completi

esempio: circuito Hamiltoniano (HCIRCUIT)

istanza: grafo G=(V,E)

predicato: G contiene un circuito Hamiltoniano?

riduzione: VERTEXCOVER$\leq_p$HCIRCUIT

Karp 1972

nota: il problema del circuito Euleriano e' invece polinomiale

il punto della situazione