Esercizi di Informatica Teorica

Complessità

a cura di
Luca Cabibbo e Walter Didimo

Sommario

- tipologie di problemi e notazioni sulla complessità
- classi di complessità
- appartenenza di problemi a classi di complessità
- problemi NP-completi

notazioni sul livello degli esercizi: (*) facile, (**) non difficile
(***) media complessità, (****) difficile, (*****) quasi impossibile

Tipologie di problemi

data una funzione $f: I \rightarrow \mathbf{P}(O)$ (insieme delle parte di O)
distinguiamo le seguenti tipologie di problemi:

• problema di decisione
dato $x \in I$ ed $y \in O$, è vero che $y \in f(x)$?
equivale ad avere un predicato: $p(x,y,f) \rightarrow \{vero, falso\}$
• problema di ricerca
dato $x \in I$, determinare un $y \in O$ tale che $y \in f(x)$
• problema di enumerazione
dato $x \in I$, determinare $|f(x)|$ (oppure generare l'intero insieme $f(x)$)
• problema di ottimizzazione
dato $x \in I$, determinare $y \in f(x)$ tale che y sia la "migliore soluzione" rispetto ad una misura fissata

Esempio di tipologie di problemi

esempio: dato un grafo $G=(V,E)$, sia $I = \{(u,v) \in V \times V : u \neq v\}$ e sia O l'insieme di tutti i possibili cammini tra due nodi di G; sia inoltre $f(u,v) = \{$tutti i cammini da u a v in G$\}$

• problema di decisione
dato $(u,v) \in I$ ed $y \in O$, è vero che y è un cammino da u a v ?
• problema di ricerca
dato $(u,v) \in I$, determinare un cammino y da u e v
• problema di enumerazione
dato $(u,v) \in I$, determinare il numero di cammini da u e v (cioè $|f(u,v)|$)
• problema di ottimizzazione
dato $(u,v) \in I$, determinare il cammino più corto da u a v

Complessità asintotica di algoritmi

- $t_A(x)$ = tempo speso dall'algoritmo A sull'input x
- $s_A(x)$ = spazio (aggiuntivo) speso dall'algoritmo A sull'input x
- $T_A(n) = \max\{t_A(x) : |x| = n\}$ (tempo speso da A nel caso peggiore su input di dimensione n)
- $S_A(n) = \max\{s_A(x): |x| = n\}$ (spazio speso da A nel caso peggiore su input di dimensione n)
- $T_A(n) = O(g(n))$ se esistono c ed n_0 tali che $T_A(n) \le c|g(n)|\ \forall n \ge n_0$
- $T_A(n) = \Omega(g(n))$ se esistono c ed n_0 tali che $T_A(n) \ge c|g(n)|\ \forall n \ge n_0$
- $S_A(n) = O(g(n))$ se esistono c ed n_0 tali che $S_A(n) \le c|g(n)|\ \forall n \ge n_0$
- $S_A(n) = \Omega(g(n))$ se esistono c ed n_0 tali che $S_A(n) \ge c|g(n)|\ \forall n \ge n_0$

$$c_1|g_1(n)| \le T_A(n) \le c_2|g_2(n)| \quad \Leftrightarrow \quad \Omega(g_1(n)) = T_A(n) = O(g_2(n))$$

$$c_1|g_1(n)| \le S_A(n) \le c_2|g_2(n)| \quad \Leftrightarrow \quad \Omega(g_1(n)) = S_A(n) = O(g_2(n))$$

Complessità asintotica di problemi

- un problema P ha una complessità temporale $O(g(n))$ se esiste un algoritmo A che risolve P tale che $T_A(n) = O(g(n))$; tale complessità si chiama anche un upper bound del problema P;
- un problema P ha una complessità temporale $\Omega(g(n))$ se ogni algoritmo A che risolve P è tale che $T_A(n) = \Omega(g(n))$; tale complessità si chiama anche un lower bound del problema P;

le stesse definizioni valgono nel caso della complessità spaziale

osservazione: l'analisi di complessità di un problema è tanto migliore quanto più piccolo è l'upper bound e quanto più grande è il lower bound; determinare il più grande lower bound significa determinare la complessità intrinseca del problema

Considerazioni utili

• determinare un lower bound alla complessità di un problema è più difficile che determinarne un upper bound, perchè occorre ragionare su ogni possibile algoritmo (cioè ragionare indipendentemente dal procedimento algoritmico adottato per risolvere il problema)

• esistono dei lower bound elementari che si possono ricavare; ad esempio, se un problema richiede necessariamente di leggere o scrivere n celle di nastro per essere risolto, allora $\Omega(n)$ sarà un suo lower bound (non è detto però che sia il più alto)

• se per un problema si trovano un lower bound ed un upper bound coincidenti, allora vuol dire che si è trovata la complessità intrinseca del problema ed anche un algoritmo di risoluzione ottimo

Esercizio sulla complessità asintotica

Esercizio 1(***) si consideri il problema P di decidere se una stringa appartiene o meno al linguaggio $L=\{a^nb^n : n \geq 0\}$:

• trovare un lower bound temporale del problema rispetto al modello di calcolo di Turing (la taglia dell'input è 2n);

• mostrare una macchina di Turing mononastro che risolve P in tempo $O(n^2)$ e specificare quale è la sua complessità spaziale;

• mostrare una macchina di Turing qualunque che risolve P in tempo $O(n)$ e specificare quale è la sua complessità spaziale;

• mostrare una macchina di Turing qualunque che risolve P in spazio $O(\log n)$;

• dire se è possibile trovare una macchina di Turing che risolve il problema in tempo inferiore ad $O(n)$

Classi di complessità

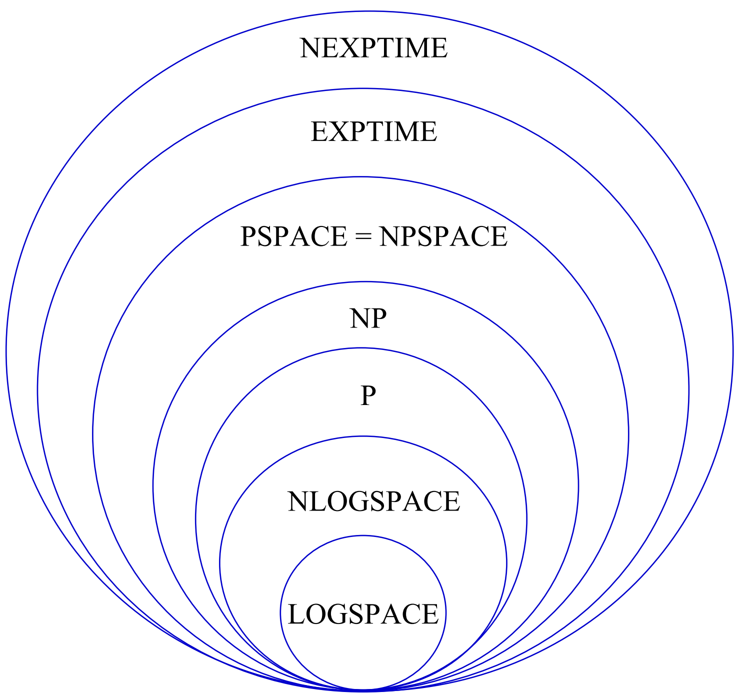
ogni problema di decisione può essere visto come un problema di riconoscimento di linguaggio:
distinguiamo le seguenti classi di complessità:

- DTIME(g(n)) : linguaggi decisi in tempo O(g(n)) da una MT
- DSPACE (g(n)): linguaggi decisi in spazio O(g(n)) da una MT
- NTIME(g(n)): linguaggi accettati in tempo O(g(n)) da una MTND
- NSPACE(g(n)): linguaggi accettati in spazio O(g(n)) da una MTND

classi notevoli:

- LOGSPACE = DSPACE(log n) NLOGSPACE = NSPACE(log n)
- PTIME = P = $\cup_k$ DTIME(n^k) NPTIME = NP = $\cup_k$ NTIME(n^k)
- PSPACE = $\cup_k$ DSPACE(n^k) NPSPACE = $\cup_k$ NSPACE(n^k)
- EXPTIME = $\cup_k$ DTIME(2^{n^k}) NEXPTIME = $\cup_k$ NTIME(2^{n^k})

Relazioni tra classi di complessità



inclusioni strette accertate:

P $\subset$ EXPTIME
NP $\subset$ NEXPTIME
PSPACE $\subset$ EXPSPACE

un problema ancora aperto:

P = NP ?

ci si convince sempre di più che non sia così, ma nessuno lo ha mai dimostrato

Astrarre sul modello di calcolo

• anche se l'appartenenza di problemi alle classi di complessità considerate è stata definita rispetto ad MT, è possibile variare il modello di calcolo, mantenendo valide le relazioni di appartenenza di problemi; ad esempio il modello RAM permette solo di migliorare di un polinomio sui tempi di computazione di una MT (non cambia la classe di complessità)

• per valutare la complessità di un algoritmo descritto come sequenza di passi, si devono analizzare due aspetti:
- il numero di passi richiesti dall'algoritmo rispetto alla lunghezza dell'input;
- la complessità richiesta da ogni singolo passo sempre rispetto alla lunghezza dell'input.

Esercizi sulle classi di complessità

Esercizio 2(***) un grafo G=(V,E) è bipartito se esiste una partizione {A, B} di V tale che due nodi nello stesso insieme (A o B) non sono mai adiacenti (cioè collegati da un arco);
si consideri il seguente problema di decisione (BIPARTITO):
dato un grafo G connesso, è G bipartito?
assumendo che la lunghezza dell'input è il numero di vertici di G:
• dimostrare che il problema BIPARTITO appartiene alla classe NP;
• dimostrare che il problema BIPARTITO appartiene alla classe P.

Soluzione
• dimostriamo che BIPARTITO appartiene alla classe NP:
per fare questo descriviamo un algoritmo non deterministico che in tempo polinomiale decide se G è bipartito o meno

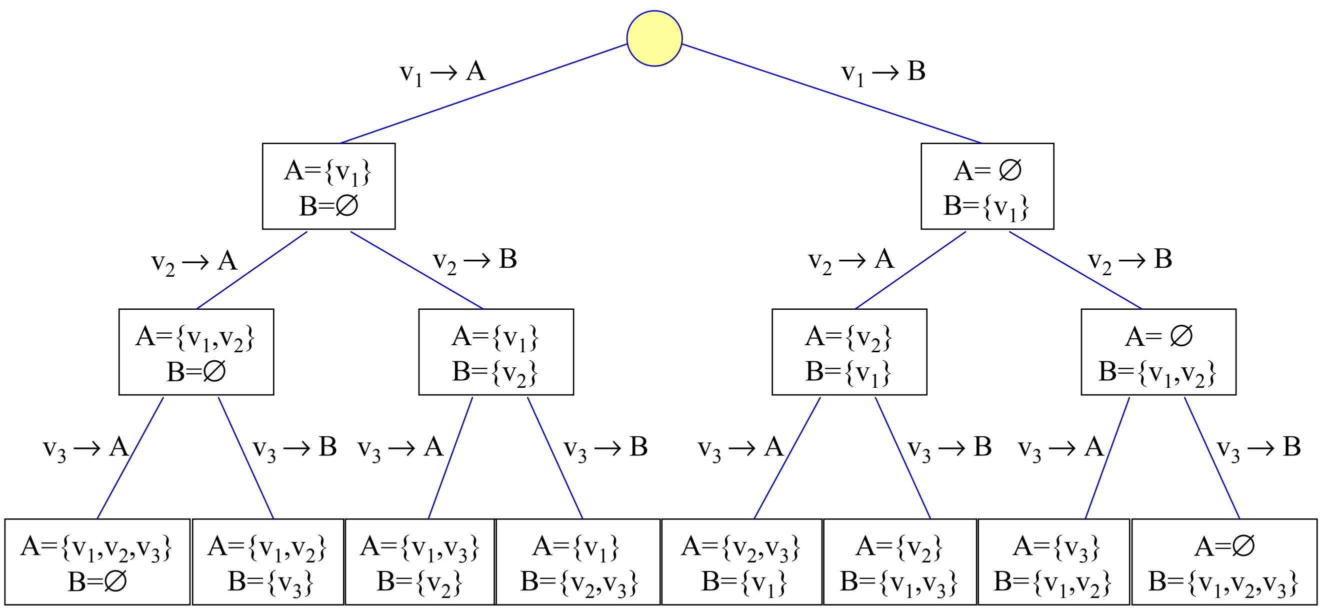
Esercizi sulle classi di complessità

input: G=(V,E) connesso
output: un valore in {vero, falso}

- ordina arbitrariamente i nodi di $V=\{v_1, v_2, ..., v_n\}$ e poni inizialmente $A=\varnothing$ e $B=\varnothing$;
- al primo passo esegui non deterministicamente due operazioni:
  - metti v_1 in A
  - metti v_1 in B

  (la computazione si divide in due rami);
- per ogni i = 2, ...,n (passo i-esimo), su ogni ramo della computazione (i rami sono 2^{i-1}) esegui non deterministicamente due operazioni:
  - metti v_i in A
  - metti v_i in B
- per ogni ramo verifica che in A ed in B non ci siano due nodi adiacenti; se esiste un ramo che verifica la proprietà, ritorna vero, altrimenti ritorna falso

Esercizi sulle classi di complessità



albero di computazione per G con tre vertici

Esercizi sulle classi di complessità

analisi della complessità

- occorre pensare che nell'algoritmo descritto ogni ramo della computazione possa essere eseguito contemporaneamente agli altri;
- su ogni ramo l'algoritmo descritto esegue n passi distinti, in ognuno dei quali effettua l'inserimento di un nodo in uno dei due insiemi A e B (O(1)); all fine degli n passi esegue una verifica sulla correttezza della bipartizione costruita confrontando un numero $O(n^2)$ di coppie di nodi
- dunque il tempo di calcolo è $O(n^2)$

riflessioni

- come si può rendere più "furbo" l'algoritmo non deterministico sopra descritto?
- sapendo che in un grafo planare G risulta m=O(n), è possibile abbassare il tempo di calcolo dell'algoritmo su grafi planari?

Esercizi sulle classi di complessità

• dimostriamo che BIPARTITO appartiene alla classe P:
descriviamo un algoritmo deterministico che in tempo polinomiale decide se G è bipartito o meno

idee di base per la formulazione dell'algoritmo:

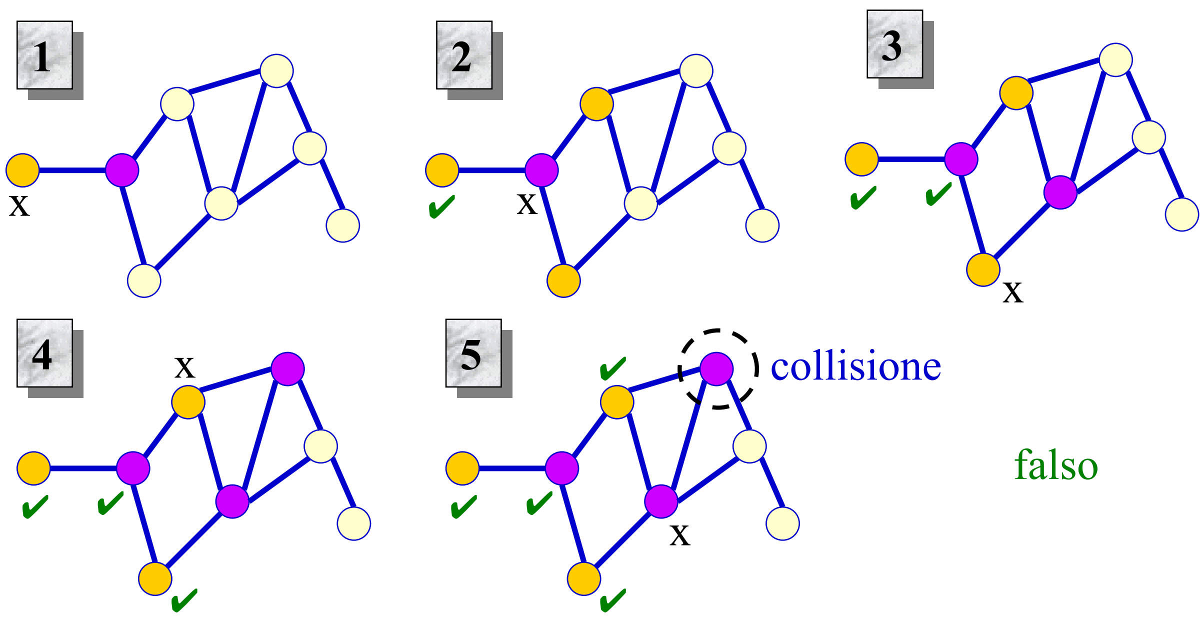
- gli insiemi A e B vengono costruiti progressivamente;
- ad ogni passo si "esamina" un nodo diverso che sta in A o in B e si mettono tutti i suoi adiacenti nell'insieme opposto
- se si ha bisogno di inserire un nodo in un insieme, e se tale nodo era già stato inserito nell'insieme opposto, allora si termina con risposta negativa
- si termina con risposta positiva quando non ci sono più nodi da esaminare

Esercizi sulle classi di complessità

input: G=(V,E) connesso
output: un valore in {vero, falso}
- poni A=∅ e B=∅ e marca tutti i nodi come non esaminati;
- al primo passo scegli un nodo qualunque x e mettilo in A, metti tutti i nodi adiacenti ad x in B, e marca x come esaminato
- al generico passo scegli un nodo x non ancora esaminato tra quelli già inseriti in A o in B:
• se x sta in A, allora controlla tutti i nodi adiacenti ad x; se tra questi nodi ce n'è uno già inserito in A allora termina e ritorna falso, altrimenti inserisci tutti gli adiacenti in B e marca x come esaminato;
• se x sta in B procedi simmetricamente al caso sopra descritto
• se x non esiste (cioè tutti i nodi sono stati esaminati), allora termina e ritorna vero

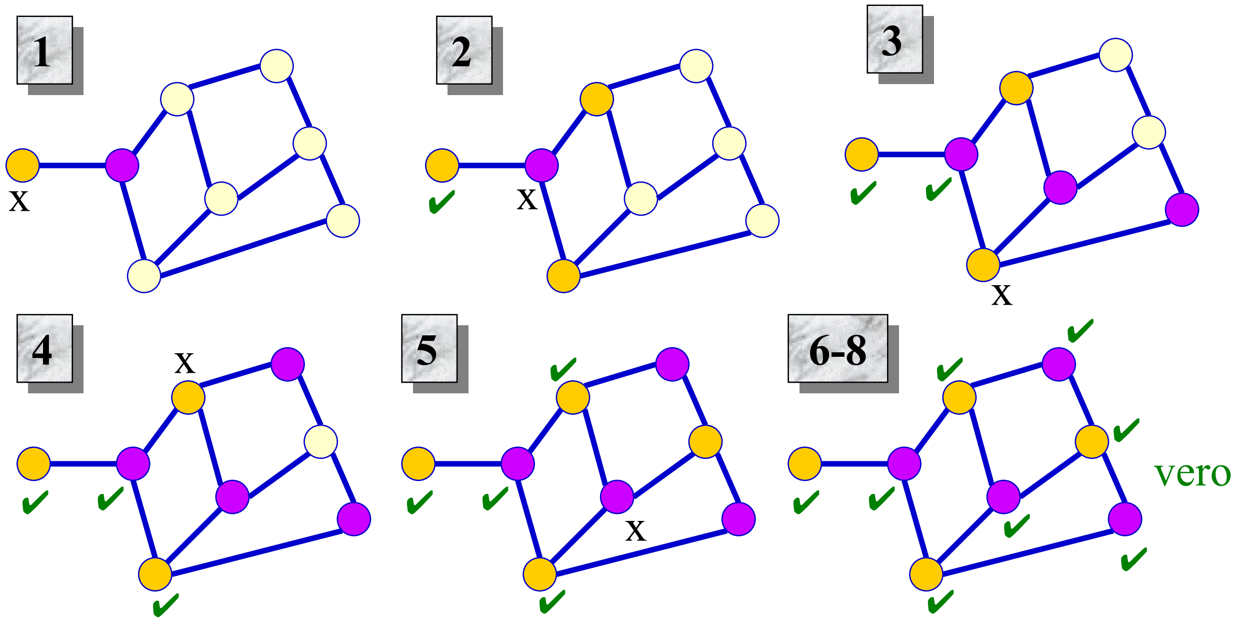
Esercizi sulle classi di complessità

esempio di calcolo su un grafo non bipartito: coloriamo i nodi in due modi diversi per indicare che appartengono ad A o a B



Esercizi sulle classi di complessità

esempio di calcolo su un grafo bipartito: coloriamo i nodi in due modi diversi per indicare che appartengono ad A o a B



Esercizi sulle classi di complessità

analisi della complessità

- l'algoritmo è sicuramente polinomiale, in quanto ad ogni passo esamina un nodo e tutti i suoi adiacenti; poiché i nodi sono n e gli adiacenti di ogni nodo sono al più n, allora l'algoritmo ha costo $O(n^2)$

- l'algoritmo può essere implementato in modo da spendere tempo lineare nel numero m degli archi del grafo, cioè costo $O(m)$ (nota che in un grafo connesso $m \geq n-1$); per far questo è sufficiente:

  - mantenere dei marcatori che indicano se un nodo sta in A o in B e se è già stato esaminato
  - mantenere una lista dei nodi già inseriti in A e in B ma non ancora esaminati (così al generico passo si sceglie x in tempo costante)
  - osservare che per visitare gli adiacenti di un nodo x si spende tempo deg(x) (num. di adiacenti di x) e che risulta $\sum_x \deg(x) = 2m = O(m)$)

Esercizi sulle classi di complessità

Esercizio 3(***) un grafo $G=(V,E)$ è k-colorabile se esiste una partizione $\{A_1, ..., A_k\}$ di V tale che due nodi nello stesso insieme A_i ($i = 1,.., k$) non sono mai adiacenti;
si consideri il seguente problema di decisione (k-COLORABILITA'):
dato un grafo G connesso, è G k-colorabile?
assumendo che la taglia dell'input è il numero di vertici di G:

- dimostrare che il problema k-COLORABILITA' è in EXPTIME
- dimostrare che il problema k-COLORABILITA' è in NP
- dimostrare che il problema 2-COLORABILITA' è in P

Riducibilità

notazioni: dato un problema di decisione P, ed il predicato π ad esso associato, denotiamo con Y_P l'insieme delle istanze x di P per cui $\pi(x)$ = vero (istanze positive), e con N_P l'insieme delle istanze x di P per cui $\pi(x)$ = falso (istanze negative) ;

definizione: un problema di decisione A è Karp-riducibile ad un secondo problema di decisione B se esiste un algoritmo R che trasforma ogni istanza x di A in una particolare istanza R(x) di B, in modo tale che $x \in Y_A \Leftrightarrow R(x) \in Y_B$ (si scrive in tal caso $A \leq_m B$);
se la riduzione R è polinomiale si dice che A è polinomialmente Karp-riducibile a B (si scrive $A \leq_{mp} B$)

implicazione $(A \leq_{mp} B)$ e $(B \in PTIME) \Rightarrow A \in PTIME$

Il problema SAT

il problema SAT:

- istanza:
  - $X = \{x_1, .., x_n\}$ insieme di variabili booleane
  - una formula booleana F su X in forma normale congiuntiva

  esempio: $F = \underbrace{(x_1 \vee x_2 \vee \neg x_3)}_{\text{clausola}} \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee x_3 \vee x_4)$

  (ogni variabile x_i ed ogni variabile negata $\neg x_i$ si chiama letterale)

- quesito: esiste una assegnazione di valori alle variabili booleane in X tale che F è soddisfatta?

(nota che F è soddisfacibile $\Leftrightarrow$ ogni sua clausola è soddisfacibile allo stesso tempo)

Esercizi sulle classi di complessità

Esercizio 4(****) il problema 2SAT è il caso particolare di SAT in cui ogni clausola ha esattamente due letterali; dimostrare che 2SAT è in P

Soluzione

- il problema CAMMINO consiste nel decidere se in un grafo diretto G=(V,E) esiste un cammino tra due nodi fissati; sappiamo che tale problema è risolvibile in tempo polinomiale attraverso una visita in ampiezza del grafo;

- definiamo il problema CAMMINO-L come la seguente variante del problema CAMMINO: dato un insieme di O(|V|) coppie di nodi in G, esiste un cammino tra i nodi di ciascuna coppia?
tale problema è niente altro che un insieme di O(|V|) problemi di tipo CAMMINO, ed è pertanto ancora risolvibile polinomialmente

Esercizi sulle classi di complessità

• mostriamo allora che 2SAT $\leq_{mp}$ CAMMINO-L

le clausole della formula di un problema 2SAT sono formate da due soli letterali, cioè sono della tipo $(\alpha \vee \beta)$, dove α e β sono variabili booleane o variabili booleane negate;

- si osserva che $(\alpha \vee \beta)$ si riscrive in forma di implicazioni logiche al modo: $(\neg\alpha \Rightarrow \beta) \wedge (\neg\beta \Rightarrow \alpha)$
infatti affinché $(\alpha \vee \beta)$ sia vera, α=falsa (cioè $\neg\alpha$=vera) deve implicare β=vera, e β=falsa (cioè $\neg\beta$=vera) deve implicare α=vera;

- possiamo allora riscrivere una formula di 2SAT nella nuova forma logica, cioè come un insieme di implicazioni;

Esercizi sulle classi di complessità

<u>esempio:</u>

$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_3) \wedge (x_2 \vee x_3)$

si riscrive al modo:

$(\neg x_1 \Rightarrow x_2) \wedge (\neg x_2 \Rightarrow x_1) \wedge (x_1 \Rightarrow x_2) \wedge (\neg x_2 \Rightarrow \neg x_1)$
$\wedge (\neg x_1 \Rightarrow \neg x_3) \wedge (x_3 \Rightarrow x_1) \wedge (\neg x_2 \Rightarrow x_3) \wedge (\neg x_3 \Rightarrow x_2)$

- a seguito di questa riscrittura <u>si dimostra</u> che:
una formula del problema 2SAT è <u>non soddisfacibile</u> $\Leftrightarrow$ <u>esiste una variabile x tale che "x implica ¬x" e "¬x implica x"</u>, dove il termine "implica" indica una sequenza di implicazioni logiche "$\Rightarrow$"
(provare a dimostrare per esercizio, usando l'osservazione che se "α implica β" allora è anche vero che "$\neg\beta$ implica $\neg\alpha$", dove α e β sono due letterali)

Esercizi sulle classi di complessità

<u>esempio:</u>

$$F = \overbrace{(\neg x_1 \vee x_2)}^{1} \wedge \overbrace{(\neg x_2 \vee \neg x_3)}^{2} \wedge \overbrace{(\neg x_1 \vee x_3)}^{3} \wedge \overbrace{(x_1 \vee x_2)}^{4} \wedge \overbrace{(x_1 \vee \neg x_2)}^{5}$$

è non soddisfacibile perché (x_1 implica $\neg x_1$) ed ($\neg x_1$ implica x_1);

infatti: $(x_1 \xRightarrow{1} x_2 \xRightarrow{2} \neg x_3 \xRightarrow{3} \neg x_1)$ e $(\neg x_1 \xRightarrow{1} x_2 \xRightarrow{5} x_1)$

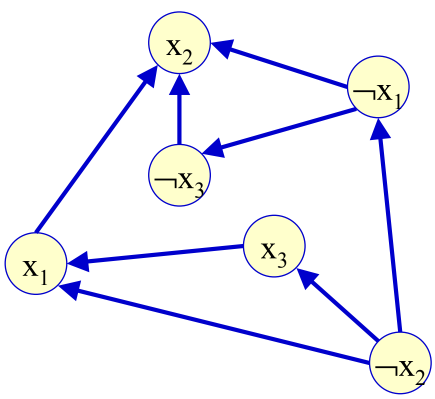
- la <u>riduzione</u> 2SAT $\leq_{mp}$ CAMMINO-L è definita come segue:

- il grafo G del problema CAMMINO-L ha per nodi tutti i letterali del problema 2SAT , cioè per ogni variabile x ci sono i nodi x e $\neg$x
- per ogni clausola $(\alpha \vee \beta)$ inseriamo in G gli archi $(\neg\alpha, \beta)$ e $(\neg\beta, \alpha)$
- le coppie di nodi di G tra cui si deve ricercare un cammino sono tutte quelle del tipo $\{x, \neg x\}$ e $\{\neg x, x\}$, con x variabile booleana

Esercizi sulle classi di complessità

<u>esempio di riduzione:</u>

$$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_3) \wedge (x_2 \vee x_3)$$



ad esempio questa formula è soddisfatta dall'assegnazione seguente:

x_1= vero x_2= vero x_3= falsa

Completezza

definizione: sia C una classe di complessità ed A un problema;
si dice che A è un problema C-completo se:

• A appartiene alla classe C

• ogni problema B appartenente a C è riducibile ad A, secondo una qualche riduzione $\leq_r$

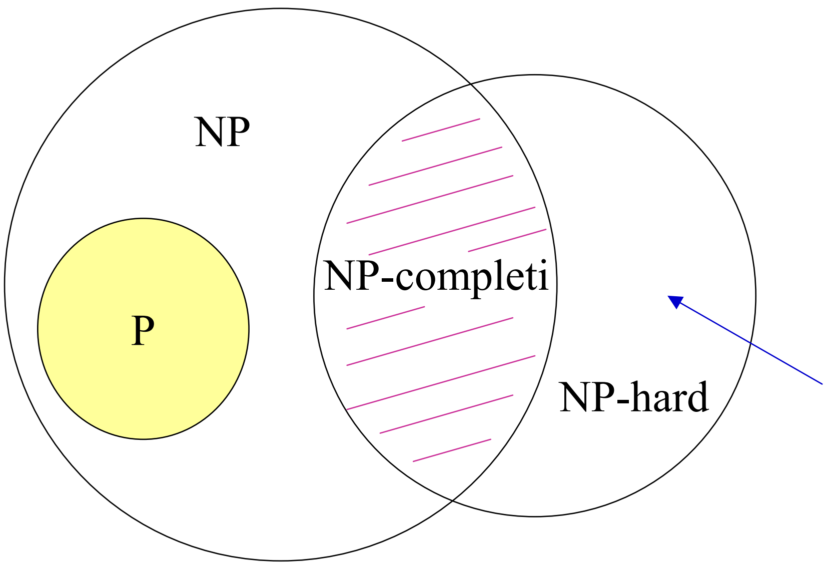
definizione un problema A è NP-completo se:

• A appartiene alla classe NP (membership);

• per ogni problema B di NP riesce: B $\leq_{mp}$ A (hardness)

implicazione se B è un problema NP-completo ed A è un problema tale che B $\leq_{mp}$ A , allora A è NP-completo

Relazione tra P, NP, NP-completi



problemi per i quali si è solo dimostrata la hardness

se si dimostra che un problema NP-completo è in P, allora si dimostra che P = NP (problema aperto)

Esercizi sulle classi di complessità

Esercizio 5(*) il problema SAT con l'ulteriore vincolo che in ogni clausola non ci sono mai ripetizioni di letterali è ancora NP-completo

Soluzione

• il problema è ancora in NP in quanto caso particolare di SAT;

• per mostrare che il problema è NP-completo è dunque sufficiente mostrare una riduzione da SAT: basta osservare che in ogni clausola si possono eliminare le ripetizioni di letterali senza alterare la sua tavola di verità

esempio: $(x_1 \vee x_2 \vee \neg x_3 \vee x_2) \Leftrightarrow (x_1 \vee x_2 \vee \neg x_3)$

Esercizi sulle classi di complessità

Esercizio 6(***) il problema 3SAT è il caso particolare di SAT in cui ogni clausola ha esattamente tre letterali; dimostrare che 3SAT è NP-completo

Soluzione

• il problema è in NP in quanto caso particolare di SAT;

• per mostrare che il problema è NP-completo mostriamo che SAT $\leq_{mp}$ 3SAT; consideriamo una istanza generica <X, F> del problema SAT e costruiamo a partire da essa una istanza <X',F'> del problema 3SAT: poniamo inizialmente X' = X; per ogni clausola C di F distinguiamo tre possibilità:

Esercizi sulle classi di complessità

• C ha esattamente tre letterali $\Rightarrow$ aggiungiamo C ad F';

• C ha meno di tre letterali $\Rightarrow$ sia C' la clausola ottenuta da C ripetendo un qualunque letterale di C tante volte quanto basta affinché C' abbia tre letterali; quindi aggiungiamo C' ad F';

esempio: $C = (x_1 \vee x_2) \Rightarrow C' = (x_1 \vee x_2 \vee x_2)$

• C ha più di tre letterali; in tal caso, se $C = (\alpha_1 \vee \alpha_2 \ldots \vee \alpha_n)$ introduciamo in X' le nuove variabili $\{z_1, \ldots, z_{n-3}\}$ ed aggiungiamo ad F il seguente insieme di clausole

$C' = (\alpha_1 \vee \alpha_2 \vee z_1) \wedge (\neg z_1 \vee \alpha_3 \vee z_2) \wedge \ldots \wedge (\neg z_{n-3} \vee \alpha_{n-1} \vee \alpha_n)$

e verifichiamo che "C' è soddisfatta $\Leftrightarrow$ C è soddisfatta"

Esercizi sulle classi di complessità

- se C è vera $\Rightarrow$ esiste un letterale α_i = vero; allora per fare in modo che C' sia vera basta assegnare valore falso ai nuovi letterali di C' che stanno nella stessa clausola di α_i; poi si propaga l'assegnamento di valori ai nuovi letterali, facendo in modo che in ogni clausola di F' ce ne sia almeno uno uguale a vero

esempio: $C' = (\alpha_1 \vee \alpha_2 \vee z_1) \wedge (\neg z_1 \vee \alpha_3 \vee z_2) \wedge (\neg z_2 \vee \alpha_4 \vee \alpha_5)$

vero falso vero falso vero

- se C è falsa $\Rightarrow$ ogni α_i = falso, ed allora, per far in modo che la prima clausola di C' sia vera occorre assegnare z_1 = vero; questo implica che $\neg z_1$ = falso, e che dunque occorre assegnare z_2 = vero; iterando il ragionamento risulta comunque che $\neg z_{n-3}$ = falso e quindi C' falsa

Esercizi sulle classi di complessità

Esercizio 7(****) il problema CLIQUE è il seguente:
dato un grafo G=(V,E) ed un intero $0 < k \leq |V|$, esiste un sottografo completo di G con k nodi? (un grafo è completo se ogni coppia di nodi ha un arco che li collega);
dimostrare che il problema CLIQUE è NP-completo.

Soluzione

- CLIQUE appartiene ad NP; infatti basta considerare tutti i possibili sottoinsiemi di k nodi in V, e verificare su ciascuno di essi se tutte le coppie di nodi sono collegate con un arco (tale verifica si fa facilmente in tempo $O(k^2)$)

Esercizi sulle classi di complessità

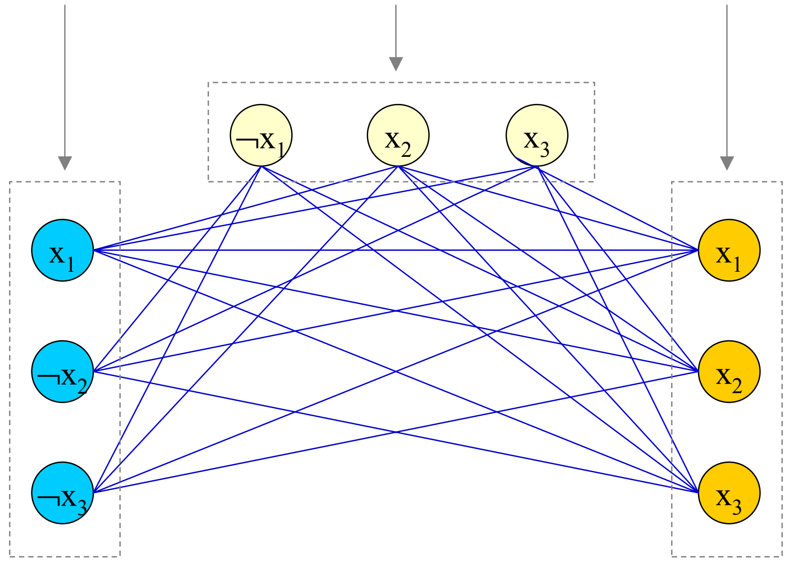
- CLIQUE è NP-hard; per dimostrare ciò cerchiamo una riduzione polinomiale del tipo: 3SAT $\leq_{mp}$ CLIQUE

costruiamo una istanza <G, k> di CLIQUE a partire da una istanza <X, F> di 3SAT;
- i nodi di G sono organizzati in k "gruppi", in cui ogni gruppo è associato ad una diversa clausola di F (stiamo indicando con k il numero di clausole di F);
- ogni gruppo ha un nodo per ogni letterale della clausola a cui è associato;
- due nodi di G sono collegati da un arco se e solo se (i) non appartengono alla stessa clausola e (ii) non si riferiscono a letterali complementari (cioè x e $\neg x$)

Esercizi sulle classi di complessità

<u>esempio di costruzione</u>: costruiamo G associato alla formula

$F = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$



Esercizi sulle classi di complessità

dimostriamo la correttezza della riduzione, cioè facciamo vedere che G ha un sottografo completo con k nodi se e solo se F ha una assegnazione di verità che la soddisfa.

• <u>supponiamo esista un sottografo completo G' di G con k nodi</u>: assegnamo vero a tutti i letterali di F associati ad un nodo di G' e falso ai letterali rimanenti; ogni nodo di G' appartiene ad una clausola distinta (in quanto G' è completo e per costruzione non ci sono archi tra nodi associati a letterali di clausole diverse) , quindi l'assegnamento stabilito soddisfa ciascuna clausola di F; inoltre tale assegnamento è consistente perché non ci sono mai due letterali complementari uniti da un arco;

• <u>supponiamo al contrario che esista una assegnazione che soddisfa F</u>: allora esistono k letterali, uno per clausola e mai complementari, con valore vero; per come G è costruito il sottografo indotto da tali nodi è completo.