

Teoria delle NP - Completezza / 05-06

Classi di problemi:

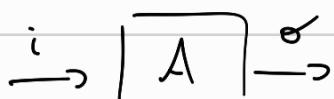
- P: risolvibili polynomialmente $n, n^2, \log n \dots$
(ovvero \exists algoritmo che risolve il problema con complessità polinomiale)
"problemI trattabili"
- Torre di Hanoi: problema intrattabile, \nexists alg che lo risolve polynomialmente
- Problemi spesso: non ha coincidenze fra l'm sup e i'ng
difficile ottenere un sup migliore

La classe P ha:

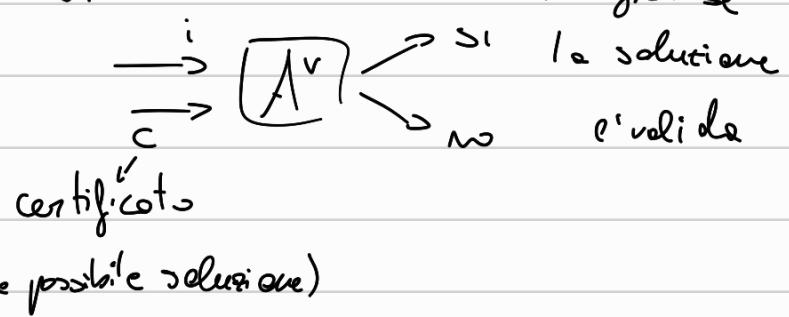
- proprietà di chiusura
- componendo alg polinomiali ho ancora alg polinomiali
- se \exists alg polinomiale spesso si migliora successivamente

Le classi NP riguarda i problemi per i quali \exists alg di verifica polinomiale (non è detto che \exists alg polinomiale che li risolva)

P:



NP:



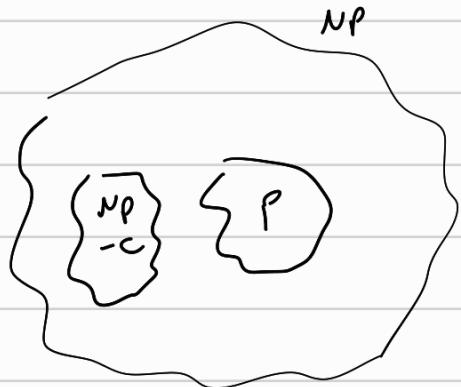
(una possibile soluzione)

es cammino viaggiatore: visitare tutti i nodi di un grafo con un ciclo unico (ciclo hamiltoniano) con costo tot. min
↳ posso produrre tutti i cammini minimi:

Si puo' facilmente (polinomiale) VERIFICARE che sia passato per tutti i nodi

$P \subseteq NP$ Se ho un alg de risolvere un problema ne ho anche uno di verifica

?
 $P = NP$



$NP - c$ sono problemi **NP complessi**, i piu' difficili: se sapessi risolverli risolverei effettuanti tipi di problemi
↳ non conosciamo alg polinomiale

• Classe $NP - c$ sono problemi eserti:

→ sono risolvibili: \exists alg esponenziale omio

→ \nexists alg polinomiale noto

→ non c'e' prove che non posse esistere tale alg

→ complessi = piu' difficili in NP, se sapessi risolverne uno di questi, li saprei risolvere tutti

Formalizziamo

- cos'è un problema
- problemi di decisione
- problema ostacolo e concreto
- classe NP; alg di decisione
- $P \leq Q$ (più difficile)
più facile di Q (se so risolvere Q , so risolvere P , polinomialmente)
- $P = ?$ NP
- esempi di NP

Nozione di problema

$$P \subseteq I \times S \quad \text{ottengo m.a.r.}$$

\downarrow inar \downarrow souz

es grafo non albero ricoprente
orientato
pesato

Un problema di **decisione** sono problemi in cui la risposta è un **BOOLANO**: $P : I \rightarrow \{T, F\}$

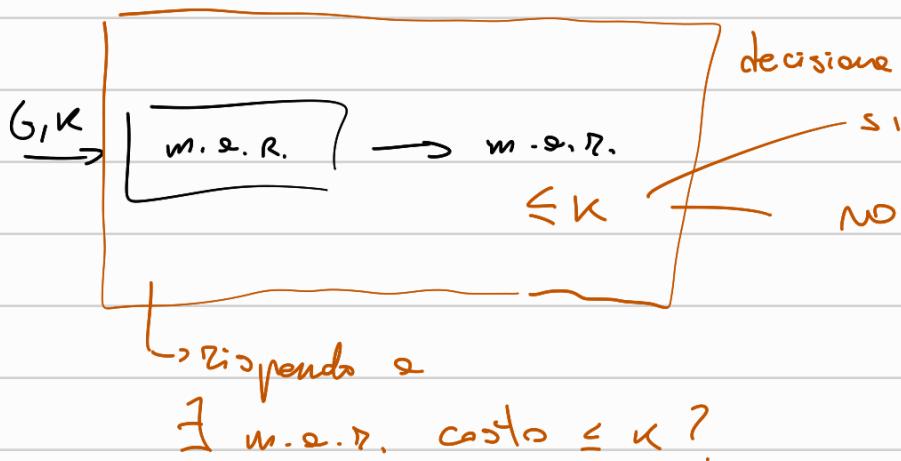
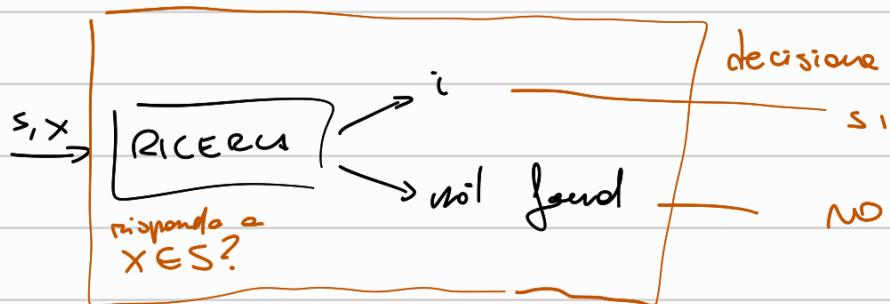
Ho problemi di:

• DECISIONE	risposte SI / NO	\rightarrow facciamo rif. a questi
• RICERCA	ricerca di un valore	
• OTIMIZZAZIONE	cerca un minimo o un max	"sono più semplici"

Questo perche' se so risolvere un problema di ricerca / ottimizzazione, so sempre risolvere quello di decisione "corrispondente"

es. voglio trovare l'indice di un elemento di una seq.

l'output sono: indice trovato / non trovato



} riesco a ridurre da un problema di decisione a uno di ricerca / ottimizzazione

se \exists alg che risolve la decisione
allora \exists alg per ricerca / ottimizzazione

(io' di interesse sono risultati negativi)

(cioe' \exists alg costo polinomiale che risolve il problema di decisione)

11-09

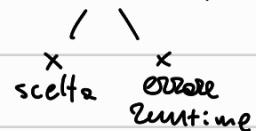
Problemi di decisione concreti

Un algoritmo A

↓

$A: T \rightarrow S$ l'algoritmo è una funzione perché x stesso input ha stesso output (deterministico)

l'algoritmo può non terminare, ha una
funzione possibile



Risolve un problema P se $\forall i \in I \quad \langle i, A(i) \rangle \in P$ (soluzione del problema)

$$\forall i \quad A(i) = ?(i)$$

Si parla di problema trattabile quando \exists alg polinomiale

Un problema **concreto** lo si ha quando l'input è di stringhe binarie

$$P \subseteq \{0,1\}^* \times \{0,1\}^*$$

$$P: \{0,1\}^* \rightarrow \{0,1\}^*$$

INPUT

SOLUZIONI

Decisiones:

$$P: \{0,1\}^* \rightarrow \{0,1\}$$

Si ottiene un codice C che trasforma l'ostacolo in concreto:

Dense input pulsars: vice versa in binary systems

- C iniettive (\times output diversi, ho input diversi), codificate
 - C surgettive NO, possono esistere delle sequenze output che non corrispondono ad alcun input

C coloro che si vogliono condizionare tutti gli input

$$P \longrightarrow C(P)$$

estraido obtengo
P concretos

$$T \xrightarrow{P} \{T, F\}$$

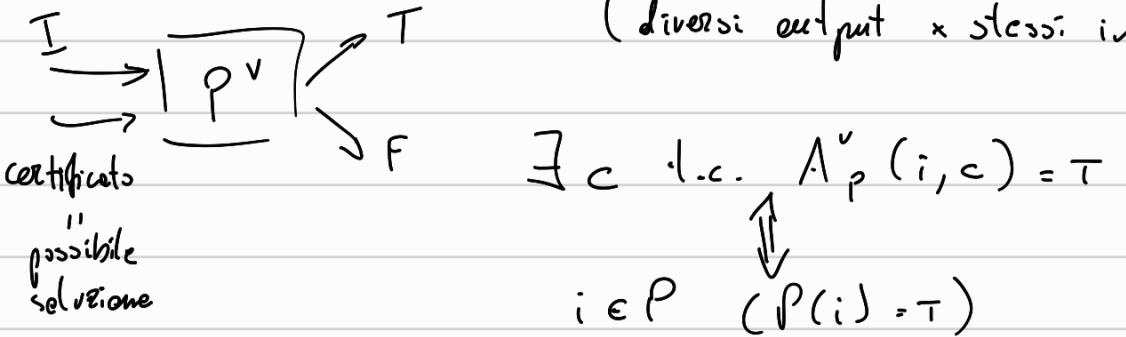
$$C \downarrow \{0,1\}^* \xrightarrow{P_C} \{\top, \text{F}\}$$

$$P_c(x) = T \text{ sse } P(i) = T \quad || \quad F \text{ su } x \text{ che non sono} \\ \in C(i) = x \quad \text{codifiche (dette spazio)}$$

Utilizzo strutture binarie per 'enificiare' gli input
(no strutture complesse & altro)

Claasse NP (non deterministic polynomial time)

Sono i problemi per cui \exists alg di verifica polinomiale
 $\equiv \exists$ alg non deterministica polinomiale
 (diversi output x stessi input)



Prendiamo per esempio consideriamo problemi di soddisfribilità SAT
 e formule in forma normale congiuntiva (CNF)

Td: formule binarie dei letterali: \neg negazioni di essi

Una clausola è un \vee di letterali

Una formula è un \wedge di clausole

$$\text{es. } (x \vee \bar{y}) \wedge x \wedge (\bar{x} \wedge \bar{x} \vee \bar{x}) \wedge (x \vee y)$$

Problema SAT: la formula è soddisfacibile?

(\exists un'assegnazione di valori di verità ai letterali
 che rende l'espressione vera?)

nell'es. per $x = \top$ e $y = \perp$ o $x = \perp$ e $y = \top$

Quindi come risolve SAT?

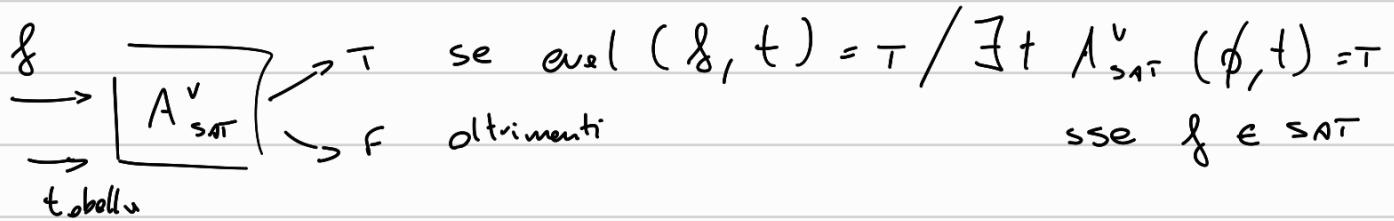
Sì, una brute force, basta prendere 2^n tabelle di verità, con $n = n$ letterali, possibili e valutato la formula con ogni combinazione (costo esponenziale)

\exists alg polinomiale? Non e' ancora noto, problema aperto.

Ma si osserva un facile alg di verifica polinomiale:

prendere in input: valori di verita' assegnati, una formula e
una tabella di verita'

verifica la formula con quei valori di verita'



Pseudocodice

$\text{eval}(f, t) = \text{case } t \text{ of}$ (idea di un pattern matching)

$x :: \text{return } t(x)$

$\bar{x} :: \text{return not } t(\bar{x})$

$l_1 \vee \dots \vee l_n :: \text{eval}(l_1, t) \text{ OR } \dots \text{ OR } \text{eval}(l_n, t)$
 $" \wedge " :: " \text{AND} \dots \text{ AND} "$

$\exists x. f \text{ return eval}(f, t + [x \rightarrow T]) \text{ OR eval}(f, t + [x \rightarrow F])$
 $\forall x. f \text{ " AND }$

Alcuni problemi hanno soluzioni di verifica difficile, ad esempio se invece di CNF aggiungiamo \exists , i quantificatori \exists e \forall posti a inizio formula (immissione)

$$\text{es. } \forall x \exists y (x \vee z) \wedge (\bar{x} \vee y) \wedge \bar{z}$$

è soddisfacibile? Mi interessa solo z vero o falso perché x e y sono vincolate

Per far valere la formula controlla per $x = \text{vero}$ che $\exists y$

Si osserva che il problema delle formule quantificate è più difficile di SAT \uparrow

(Classe NP: problemi per cui \exists soluzioni di verifica polinomiale

$$P \in \text{NP} \iff \exists A^v_p \mid \exists c A^v_p(i, c) = T$$

$$\text{sse } P(\cdot) = T \quad \downarrow$$

piuttosto preciso: la dim del certificato deve essere polinomialmente corretto $\Rightarrow T$	altrimenti non posso verificarlo in tempo polinomiale
--	--

$$P = \{i \mid \exists c \text{ t.c. } A(i, c) = T, |c| = O(|i|^n)\}$$

N.B. Non è NP perché anche l'alg di verifica è esponenziale

$P \subseteq NP$ se \exists alg non deterministico polinomiale

Dunque perche' un alg non det prende un i , sceglie casualmente un certificato e infine esegue $A^V(i)$. L'alg ha successo $\Rightarrow \exists$ almeno un alg che ha successo.
Lo alg diversifica

Non e' il solito "algoritmo", perche' il risultato puo' essere diverso per stessi input

(es. per SAT scelgo una casuale tabella e verifico la formula, ma piu' di una tabella puo' andare bene)

Problemi NP-C (NP-completi)

classe P \exists alg (d. decisione) polinomiale

classe NP \exists alg di verifica polinomiale

$P \subseteq NP$ (P ignora i certificati)

$P \stackrel{?}{=} NP$



NP-C sono i piu' difficili in NP

cioe'

se sapevamo risolvere polynomialmente uno di questi, li sapevamo risolvere tutti \rightarrow sono detti NP-hard

\hookrightarrow se $Q \in NP$ e Q e' un problema NP-hard
allora lo un problema NP-completo

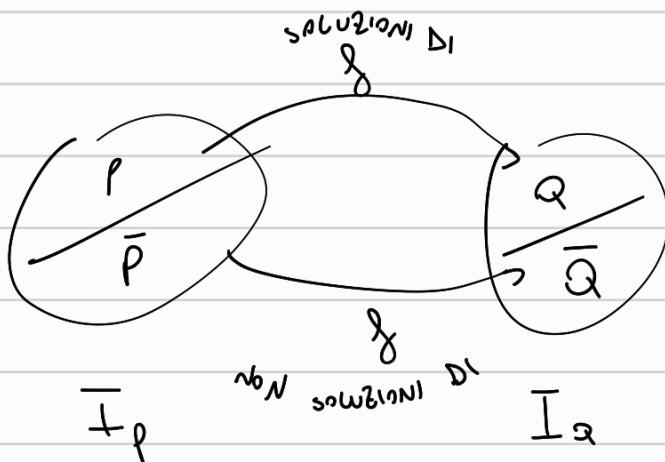
$P \leq_p Q$

detti P e Q problemi, per ogni P , Q è più difficile

$\exists f: I/P \rightarrow I/Q$ GRANDE | calcolabile polinomialmente
 $\{0,1\}^*$ $\rightarrow \{0,1\}^*$ | $x \in f \iff f(x) \in Q$

f di riduzione

$$P(x) = T \rightarrow Q(f(x)) = T \\ = F \quad = \bar{F}$$

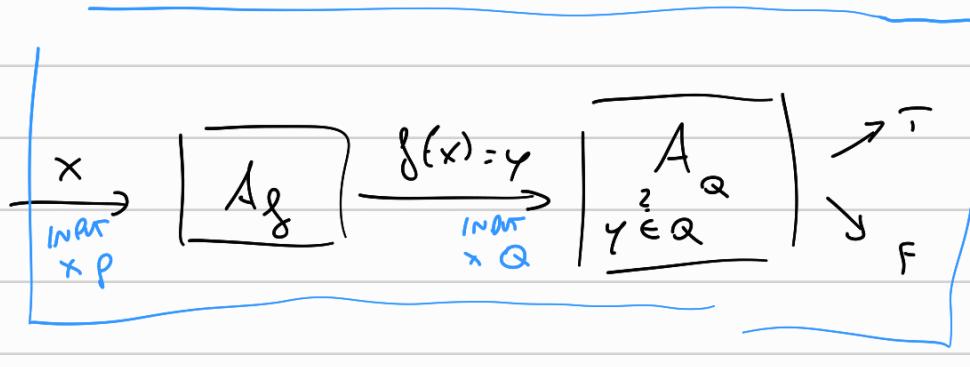


gli input vengono mappati
a seconda del valore T/F
corrispondente

alla base del riutilizzo del software

Se Q è solubile polinomialmente allora anche P lo è

utilizzo algoritmi
per costruire altri
(se tutti polinomiali,
anche il più grande
lo è)



Rispondo ad A_Q
ri-utilizzando il codice
(oppure A_f) il tutto
contenuto in un blocco

A_P

\downarrow

per essere corretto
 $A(x) = P(x)$

Se $Q \in \text{classe } P \rightarrow P \in \text{classe } P$

insolubile polinomialmente

12-04

Equividente in pseudo codice

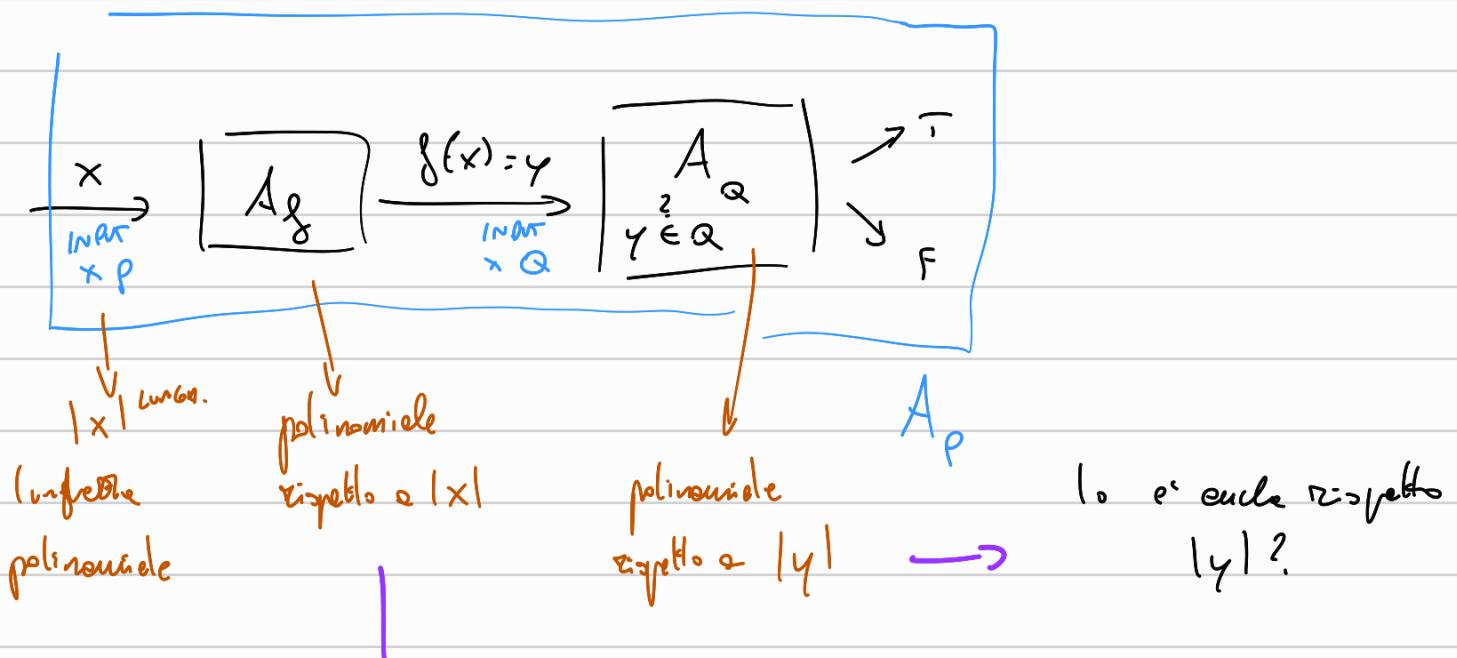
Ap:

input x

$$y = A_f(x)$$

return $A_Q(y)$

Assumendo $x \in \gamma$ stringhe binarie si vuole lunghezza polinomiale
 $\rightarrow \text{su } x \text{ lo e'}$

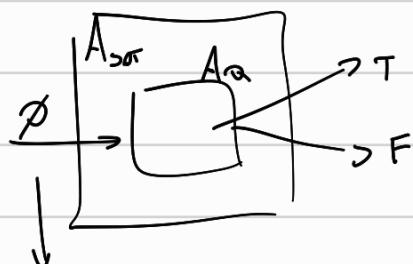


Poiché l'algoritmo è polinomiale, l'output è polinomiale rispetto all'input

quindi |y| polinomiale rispetto a |x|
e dunque A_Q polinom. ordine rispetto a |x|

Esempio

- SAT \leq Q
soddisfogibilità di ϕ in CNF " di ϕ con \forall, \exists



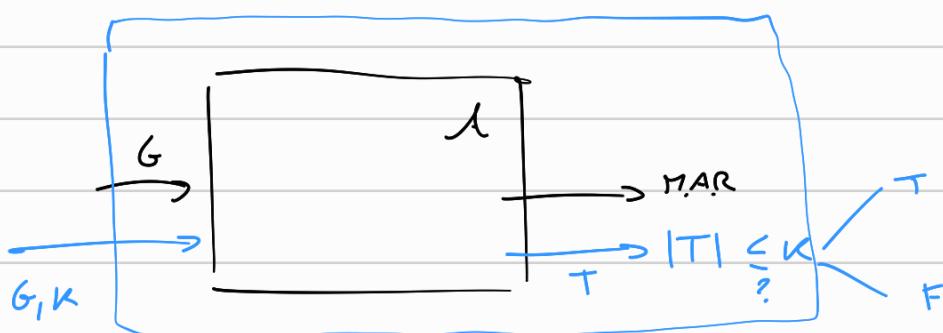
passo la medesima formula di SAT a perche'
a le centine

Q più difficile, se risolvo polinomialmente Q risolvo polinomialmente ϕ di SAT perché sono sottoinsieme



le f di riduzione (qui non trasformano niente) e l'identità (più specificatamente 'immersione')

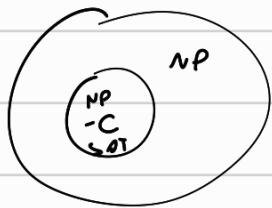
- M.A.R. • \exists m.a.r. con peso $\leq K$?



A restituisce un m.a.r.

Se so fare A so fare anche il problema di decisione che lo include

Problemi NP-Completi > 2



$P \in NP-C \Leftrightarrow$

- 1) $P \in NP \rightarrow$ sta nello stesso classe
- 2) P è NP-hard

$= \forall Q \in NP \quad Q \leq_p P$ } possono
tutti: $Q \Rightarrow$ ricadono su P } non siano NP

Esempi: SAT, Cammino viaggiatore ...

Dunque c'è noto $P \subseteq NP$ ma $P = NP$ per tali problemi? | PROBLEMA
ESISTE OLTRE
APERTO

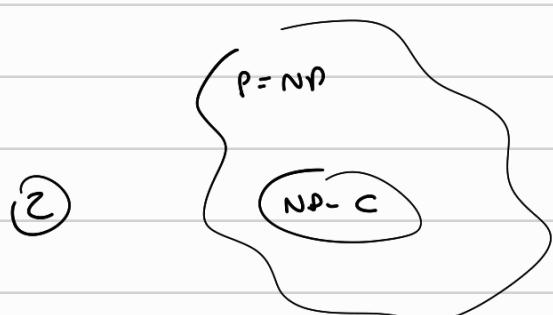
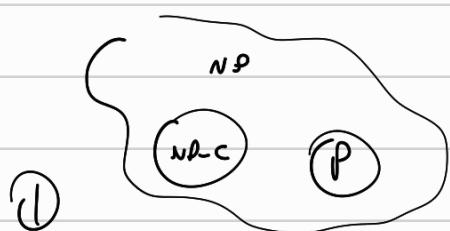
Modi di chiudere il problema

① Trovare alg. polinomiale per problemi in NP-C

In base al fatto che $\forall Q \quad Q \leq_p P$ allora tutti gli altri si ricadono su P risolvibile, quindi risolverei tutti i $Q \in NP$ (in particolare quelli completi $\subseteq NP$)

② Posso dimostrare per almeno un problema NP-C che \exists alg. polinomiale. Dato un altro problema $Q \in NP-C$

se $P \leq_p Q$ allora neanche i Q sono risolvibili polin.



Come si prova che $P \in NP\text{-C}$?

① provare che $P \in NP$ (\exists algoritmo polinomiale)

② provare che $\forall Q \in NP \quad Q \leq_p P$ (cioè P NP-hard)

Se conosciamo un altro problema che sia già essere NP-completo P' (es. SAT), per provare ① basta fornire una riduzione $P' \leq_p P$

\Rightarrow da $\forall Q \quad Q \leq_p P' \leq_p P$ per trasstruttura $Q \leq_p P$

DOT
g o g

La \circ di riduzione complessiva è la composizione delle \circ di riduzione

Ci vuole un primo problema con prove indipendente - autonoma per ②

$$\forall Q \in NP \quad Q \leq_p SAT \quad (\text{non rediemo})$$

Provare che $P \in NP\text{-C}$ è utile? Sì, dimostra che il problema di cui si parla è difficile e dunque non è sensato perdersi a cercare una soluzione polinomiale. Mi permette di cercare altre soluzioni:

{ decidere di approssimazione (es $Z^3 \rightarrow Z^{27}$)
considero problemi facili nei termini:
uso un algoritmo che da soluzione con una certa probabilità,
ma magari non corretta

Esemp: dr. riduzione

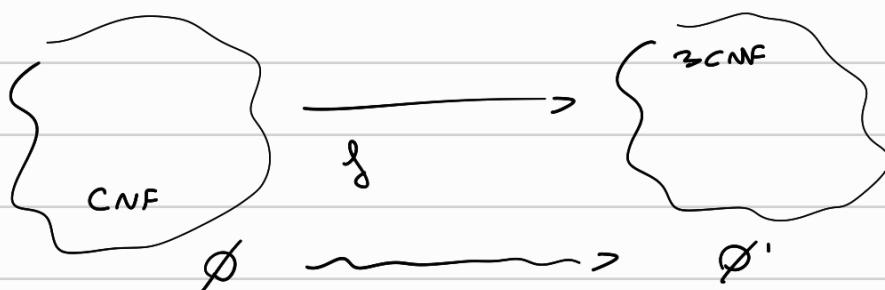
1. $SAT \leq 3SAT$ 3SAT codifica ulteriori formule
 2. $3SAT \leq CLIQUE$ 3SAT si rivolge alle formule
CLIQUE si rivolge ai grafici
- dopo aver dimostrato
che 3SAT è NP-C

1) $\ell ::= x \mid \bar{x}$ 3CNF esattamente 3 literal per OR
 $c ::= \ell_1 \vee \dots \vee \ell_n \quad |_{CNF} \rightarrow c ::= \ell_1 \vee \ell_2 \vee \ell_3$
 $\emptyset ::= c_1 \wedge \dots \wedge c_n$

$$SAT \leq 3SAT \quad (\text{SAT con 3CNF})$$

→ 3SAT è più semplice

→ se prendessi 2SAT (2 literal CNF) → da 2SAT ∈ P (deg polinomiali)



g trasforma una formula in una equivalente (soddisf. preservata)

$$\ell_1 \vee \dots \vee \ell_n \xrightarrow{\text{codifica}} \ell'_1 \vee \ell'_2 \vee \ell'_3$$

$n \neq 3$

Scelgo un modo

- Scegli 3 variabili: y_1, y_2, y_3 che dovranno essere TRUE
- Aggiungi tutte le combinazioni $y_1 \vee y_2 \vee y_3, \bar{y}_1 \vee y_2 \vee y_3, \dots$ come $\bar{y}_1 \vee \bar{y}_2 \vee \bar{y}_3$ (almeno una falsa) \rightarrow e quindi non volendo questo $y_1 = y_2 = y_3 = \text{TRUE}$

Così:

dove essere true

$$\neg l \rightarrow l \vee \bar{y}_1 \vee \bar{y}_2$$

$$\neg l_1 \vee l_2 \rightarrow l_1 \vee l_2 \vee \bar{y}_1$$

$$\neg l_1 \vee l_2 \vee l_3 \rightarrow l_1 \vee l_2 \vee l_3 \quad \text{rimane interdetto}$$

$$\neg l_1 \vee l_2 \vee c'$$

per d: 1

sostituisco

$$= y \vee c, l_1 \vee l_2 \vee \bar{y}, \bar{l}_1 \vee y \vee \bar{y}_1, \bar{l}_2 \vee y \vee \bar{y}_1$$

scalandolo di 1 il nodo literal

se $y=F, \bar{y}=F$ puoi decidere y_1 è sempre vero.

quindi: \bar{l}_1 deve essere falso

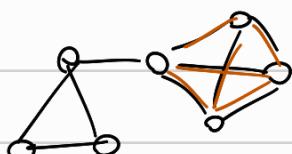
(l_1 vero per rendere $l_1 \vee l_2 \vee c'$ vero)

finché non ne ho 3

Abbiamo dimostrato che $\text{CNF} \leq 3\text{CNF}$ (CNF riducibile a 3CNF)

quindi: $\text{SAT} \leq 3\text{SAT}$

2) Problema CLIQUE: preso un grafo non orientato, prendo un sottoinsieme di nodi dove ho un arco per ogni coppia di nodi. Si vuole trovare quelle di dim max.



CLIQUE
di max dim

Il problema di decisione corrispondente è: \exists clique di dim $\geq n$?
è un problema NP-C

\downarrow

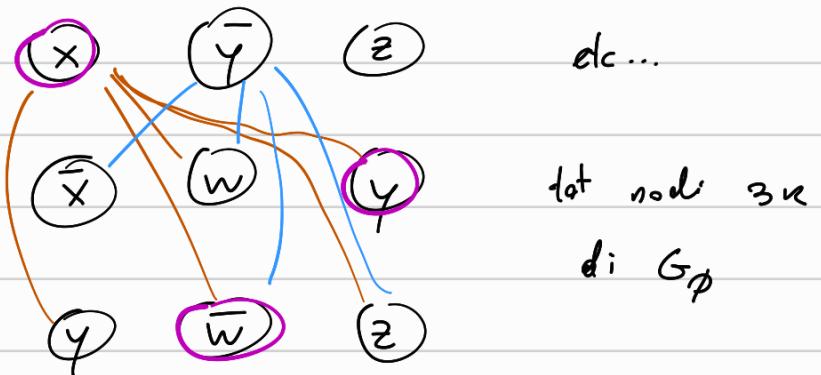
$$3\text{CNF} \leq \text{CLIQUE}$$

Preso ad es ϕ : $(x \vee \bar{y} \vee z) \wedge (\bar{x} \vee w \vee y) \wedge (y \vee \bar{w} \vee \bar{z})$

$\phi \rightsquigarrow G_\phi$

Tendo un literale x \Rightarrow n° nodi:

metto uno se sto in
clausole diverse e non
sono opposti



ϕ è soddisfacibile $\Leftrightarrow G_\phi$ ha una clique di dim n
dove $n = n^{\circ}$ clausole

soddisf. per

$x=T, y=T, w=F$ e $z=T \rightarrow$ trovo una clique • di dim 3

$x=T, y=T, w=F$ \leftarrow \exists clique di dim n , non avendo uno fra
literali di stesse clausole ha un nodo della clique
sicuramente rende vero ϕ
questo perché ha una
verità per clausola di OR
fra AND

\downarrow

faccio in modo che
il literale valga T

Quindi risulta vero $3\text{CNF} \leq \text{CLIQUE} \rightsquigarrow$ è NP-Completo

\downarrow

è NP sta in NP
(può valere di verità e
falso sostituisci)

\downarrow

è NP prendendo come certificato
un insieme di n nodi

Altri esempi di NP-C

→ se ho tutte le coppie

- insieme indipendente (in un G non orientato, sottoins. che non hanno archi fra loro)
→ problema: cercare il più grande

CLIQUE \subseteq INS. INDIP.

\forall clique in G \Leftrightarrow \forall ins. indip in G

- cammino viaggiatore (in un G un cammino (minimo se G pesato) che copre tutti i nodi)
- zaino $\xrightarrow{\text{oggetti}} \xleftarrow{\text{peso}} \text{rendimento}$ (si vuole maximizzare il rendimento col vincolo del peso)
- copertura vertici (in un G trovare sottoins. di archi che copri tutti i vertici)
- colorazione (in un G voglio colorare i nodi t.c. non ha coppie di nodi di stesso colore vicino c'è un interesse il no di colori minimo)