

Lezione 31 – NP-completezza

Lezione del 22/05/2024

Teorema di Cook-Levin e la struttura di NP

- La domanda era: **fra i problemi in NP che non si riesce a collocare in P, ce ne sono alcuni più “difficili” di altri?**
- Il **Teorema di Cook-Levin** ci dice che NP contiene un problema NP-completo
 - il problema SAT
- E, poiché sappiamo, che i problemi completi per una classe sono i problemi più “difficili” fra i problemi in quella classe
- Il **Teorema di Cook-Levin** ci dice che SAT è uno dei problemi più difficili in NP
 - perché sappiamo che se SAT appartenesse a P
 - allora ogni altro problema in NP apparterrebbe a P
 - perché, ricordiamo, P è chiusa rispetto alla riducibilità polinomiale
- Ma il **Teorema di Cook-Levin** ci dice molto di più!
- Facciamo un passo indietro

Il Teorema e la congettura

- Sappiamo della congettura $P \neq NP$
 - e del milione di dollari sulla sua risoluzione
 - in positivo, o in negativo!
- Bene. Arriva qualcuno e dimostra che $P = NP$ – *e lo fa descrivendo un algoritmo deterministico che decide SAT in tempo polinomiale*
- E allora? Nel senso: a cosa mi serve sapere che $P = NP$?
- Beh, se so che $P = NP$, sono certo che, comunque prendo un problema in NP, **esiste** un algoritmo (deterministico) che lo decide in tempo polinomiale
- Bello, per carità! Ma che ci faccio con l'**esistenza**?!
 - ... se io ho un problema importantissimo da decidere
 - e sono anni e anni che non riesco a progettare un algoritmo deterministico per deciderlo
 - però, riesco a dimostrare che quel problema è in NP
- *A che mi serve sapere che, siccome è in NP e $P = NP$, un algoritmo deterministico polinomiale che lo decide **esiste**, se io un tale algoritmo non riesco a progettarlo?!*

Se sapessi che $P = NP$

- A che mi serve sapere che, siccome il mio problema è in NP e $P = NP$, un algoritmo deterministico polinomiale che lo decide **esiste**, **se io un tale algoritmo non riesco a progettarglielo?!**
- In realtà, il **teorema di Cook-Levin** fa molto di più che dimostrare che SAT è NP-completo
- La dimostrazione del teorema di Cook-Levin è *la descrizione di un algoritmo deterministico che trasforma le istanze di un qualunque problema in NP in istanze di SAT*
- Cerchiamo di capire:
- **se abbiamo un algoritmo (deterministico) polinomiale che decide SAT allora la dimostrazione del teorema di Cook-Levin ci mostra come costruire un algoritmo polinomiale che decide qualunque problema in NP**
- Vediamo come

Se sapessi che $P = NP$

- **Supponiamo**, di avere un algoritmo (deterministico) polinomiale che decide SAT
 - chiamiamolo T_{SAT} , e diciamo che, per ogni $y \in \{0,1\}^*$, $dtime(T_{SAT}, y) \leq |y|^k$ (per qualche costante k)
- Ho un problema decisionale Γ e dimostro che $\Gamma \in NP$
 - cioè, progetto una macchina non deterministica NT_Γ che lo decide in tempo polinomiale
- Allora considero il seguente algoritmo: con input $x \in \{0,1\}^*$ (codifica di un'istanza di Γ)
 - FASE 1. Costruisce $E(x)$ – come nella dimostrazione del teorema di Cook-Levin
 - FASE 2. Esegue $T_{SAT}(E(x))$: se termina in q_A allora accetta, altrimenti rigetta
- In virtù della dimostrazione del teorema di Cook-Levin, tale algoritmo decide L_Γ
- Inoltre, esso richiede tempo polinomiale in $|x|$, infatti:
 - La FASE 1, come sappiamo, richiede tempo polinomiale in $|x|$
 - La FASE 2 richiede tempo $|E(x)|^k$ – e $E(x)$ ha lunghezza polinomiale in $|x|$
- Allora, abbiamo costruito un algoritmo (deterministico) polinomiale che decide Γ !
 - **Nell'ipotesi di avere un algoritmo (deterministico) polinomiale che decide SAT**

Il teorema e la congettura

- Quindi, **se si dimostrasse che $P = NP$**
 - **e se si trovasse un algoritmo (deterministico) polinomiale che decide SAT**
- il teorema di Cook-Levin ci permetterebbe di **costruire** un algoritmo deterministico polinomiale per decidere qualunque problema in NP
- **Ma se, invece, si dimostrasse che $P \neq NP$?**
- Allora, sapremmo che $SAT \notin P$
- E, ogni volta che riuscissimo a dimostrare che un problema è NP-completo
- sapremmo che quel problema non è in P!
- Ossia, **i problemi NP-completi sono i problemi separatori fra P e NP, nell'ipotesi $P \neq NP$**
- Certo che, se per dimostrare che un problema è NP-completo, dovessimo, ogni volta, ripetere una dimostrazione come quella di Cook-Levin...
- Per fortuna, abbiamo uno strumento che ci aiuta

Da problema a problema

- ▶ Per fortuna, abbiamo uno strumento che ci aiuta – il teorema 9.3
- ▶ Prima di vedere questo strumento, però, una precisazione è d'uopo: nella dispensa 6 abbiamo parlato di riducibilità fra linguaggi
 - ▶ Ora, però, stiamo studiando la classe NP relativamente a problemi decisionali
- ▶ Dati due problemi decisionali Γ e Λ quando è che $\Gamma \leq \Lambda$?
- ▶ Facile, quando $L_\Gamma \leq L_\Lambda$
 - ▶ dove L_Γ e L_Λ sono i linguaggi associati alle codifiche ragionevoli delle istanze sì dei due problemi
- ▶ Allora $\Gamma \leq \Lambda$ se esiste una funzione $f: \mathfrak{S}_\Gamma \rightarrow \mathfrak{S}_\Lambda$ tale che
 - ▶ $f \in \text{FP}$
 - ▶ x è una istanza sì Γ di se e soltanto se $f(x)$ è una istanza sì di Λ
- ▶ Per semplicità, d'ora in poi scriveremo $x \in \Gamma$ per intendere “ x è una istanza sì Γ ”

Da problema a problema

- **Teorema 9.3:** Sia Γ un problema in NP.
Se esiste un problema NP-completo riducibile a Γ allora Γ è NP-completo.
- Sia Λ un problema NP-completo tale che $\Lambda \leq \Gamma$.
- Poiché $\Lambda \leq \Gamma$,
 - esiste una funzione $f : \Sigma_{\Lambda} \rightarrow \Sigma_{\Gamma}$ tale che $f \in \text{FP}$ e
 - per ogni $y \in \Sigma_{\Lambda}$, $y \in \Lambda$ se e soltanto se $f(y) \in \Gamma$.
- Poiché Λ è NP-completo, per ogni problema $\Delta \in \text{NP}$, si ha che $\Delta \leq \Lambda$:
 - esiste una funzione $g : \Sigma_{\Delta} \rightarrow \Sigma_{\Lambda}$ tale che $g \in \text{FP}$ e,
 - per ogni $x \in \Sigma_{\Delta}$, $x \in \Delta$ se e soltanto se $g(x) \in \Lambda$.
- La composizione delle due funzioni g e f è una riduzione polinomiale da Δ a Γ :
 - sia $x \in \Sigma_{\Delta}$: allora, $x \in \Delta$ se e soltanto se $g(x) \in \Lambda$ e,
 - inoltre, $g(x) \in \Lambda$ se e soltanto se $f(g(x)) \in \Gamma$
 - allora, se chiamiamo h la composizione delle funzioni g e f , questo dimostra che h è una riduzione da Δ a Γ .

Da problema a problema

- ▶ Ma quanto costa calcolare h ?
- ▶ $g \in \mathbf{FP}$: allora esistono un trasduttore T_g e una costante $k \in \mathbb{N}$ tali che, per ogni $x \in \mathfrak{S}_\Delta$, $T_g(x)$ calcola $g(x)$ e $\text{dtime}(T_g, x) \leq |x|^k$
 - ▶ poiché $T_g(x)$ deve anche scrivere il risultato $g(x)$ sul nastro di output, allora $|g(x)| \leq |x|^k$
- ▶ $f \in \mathbf{FP}$: allora esistono un trasduttore T_f e una costante $c \in \mathbb{N}$ tali che, per ogni $y \in \mathfrak{S}_\Delta$, $T_f(y)$ calcola $f(y)$ e $\text{dtime}(T_f, y) \leq |y|^c$
- ▶ Definiamo il trasduttore T_h , a tre nastri, che calcola h : con $x \in \mathfrak{S}_\Delta$ scritto sul primo nastro, T_h
 - ▶ 1) esegue la computazione $T_g(x)$ scrivendo il suo output $y = g(x)$ sul secondo nastro;
 - ▶ 2) esegue la computazione $T_f(y)$ scrivendo il suo output $f(y)$ sul nastro di output.
- ▶ Per ogni $x \in \mathfrak{S}_\Delta$ $\text{dtime}(T_h, x) \leq |x|^k + |g(x)|^c \leq |x|^k + |x|^{kc} \leq 2|x|^{kc}$ e ciò dimostra che $h \in \mathbf{FP}$.
- ▶ Quindi, abbiamo dimostrato che $\Delta \preceq \Gamma$,
- ▶ poiché Δ è un qualunque problema in NP, questo prova che ogni problema in NP è riducibile polinomialmente a Γ .
- ▶ Dall'appartenenza di Γ a NP segue che Γ è NP-completo.

Da problema a problema

- Alla luce del **teorema 9.3**, per dimostrare che un problema Γ è **NP-completo** è sufficiente
 - mostrare che Γ è contenuto in NP
 - scegliere un problema NP-completo noto Λ e dimostrare che $\Lambda \leq \Gamma$
- E, in effetti, in seguito al **teorema di Cook-Levin**
 - e utilizzando il **teorema 9.3**
- è stata dimostrata la NP-completezza di numerosissimi problemi
- E noi di queste dimostrazioni ne vedremo diverse
 - a partire da oggi

Il problema 3SAT

- Abbiamo già incontrato il problema 3SAT, che qui ricordiamo:
- dati un insieme X di variabili booleane ed un predicato f , definito sulle variabili in X e contenente i soli operatori $\wedge$, $\vee$ e $\neg$, decidere se esiste una assegnazione a di valori in $\{\text{vero}, \text{falso}\}$ alle variabili in X tale che $f(a(X)) = \text{vero}$
- Consideriamo soltanto predicati f in forma 3-congiuntiva normale (3CNF), ossia,
 - f è la congiunzione di un certo numero di clausole: $f = c_1 \wedge c_2 \dots \wedge c_m$
 - e ciascuna c_j è la disgiunzione ($\vee$) di tre letterali, ad esempio $x_1 \vee \neg x_2 \vee x_3$
 - (un letterale è una variabile o una variabile negata)
- Questo problema prende il nome di 3SAT, ed è così formalizzato:
 - $\mathfrak{I}_{3\text{SAT}} = \{ \langle X, f \rangle : X \text{ è un insieme di variabili booleane } \wedge f \text{ è un predicato su } X \text{ in 3CNF} \}$
 - $\mathbf{S}_{3\text{SAT}}(X, f) = \{ a : X \rightarrow \{\text{vero}, \text{falso}\} \}$ (è l'insieme delle assegnazioni di verità alle variabili in X)
 - $\pi_{3\text{SAT}}(X, f, \mathbf{S}_{3\text{SAT}}(X, f)) = \exists a \in \mathbf{S}_{3\text{SAT}}(X, f) : f(a(X)) = \text{vero}$
- Non può non balzare all'occhio la sua somiglianza con SAT...
 - sono quasi uguali!

Il problema 3SAT

- Non può non balzare all'occhio la somiglianza di 3SAT con SAT
- Formalmente:
 - $\mathfrak{I}_{3SAT} \subseteq \mathfrak{I}_{SAT}$
 - $\mathbf{S}_{3SAT}(X, f) = \mathbf{S}_{SAT}(X, f)$
 - $\pi_{3SAT}(X, f, \mathbf{S}_{3SAT}(X, f)) = \pi_{3AT}(X, f, \mathbf{S}_{SAT}(X, f))$
- Per questa ragione diciamo che 3SAT è una **restrizione** di SAT
- Sappiamo già che $3SAT \in NP$
- Ma non sappiamo se 3SAT è NP-completo
 - la restrizione che impone che tutte le clausole contengano 3 letterali potrebbe rendere il problema più semplice rispetto alla versione in cui le clausole contengono quanti letterali gli pare...
 - Magari, proprio il fatto di avere clausole con 3 letterali può essere la chiave per trovare un algoritmo (deterministico) polinomiale che decide 3SAT
 - come accade per 2SAT (si veda dispensa 8)...

3SAT è NP-completo

- Siamo al paragrafo 9.5.1
- Dimostriamo che 3SAT è NP-completo
 - già sappiamo che $3SAT \in NP$
 - per dimostrarne la completezza per NP, utilizziamo il teorema 9.3
 - ci basta scegliere un altro problema, che già sappiamo essere NP-completo, e **ridurlo a** 3SAT
- Visto che, al momento, conosciamo un solo problema NP-completo, la scelta non è difficile...
- Riduciamo, dunque, SAT a 3SAT:
- sia $\langle X, f \rangle$ un'istanza di SAT, con $X = \{x_1, x_2, \dots, x_n\}$ e $f = c_1 \wedge c_2 \wedge \dots \wedge c_m$
 - dove ciascuna clausola c_i è la disgiunzione di un certo numero di letterali
 - ossia, tante variabili, eventualmente negate, collegate da $\vee$
- **dobbiamo trasformare $\langle X, f \rangle$ in un'istanza di 3SAT $\langle X', f' \rangle$ in modo tale che**

f è soddisfacibile se e soltanto se f' è soddisfacibile

3SAT è NP-completo

- sia $\langle X, f \rangle$ un'istanza di SAT, con $X = \{x_1, x_2, \dots, x_n\}$ e $f = c_1 \wedge c_2 \wedge \dots \wedge c_m$
 - dove ciascuna clausola c_i è la disgiunzione di un certo numero di letterali
 - ossia, tante variabili, eventualmente negate, collegate da $\vee$
- dobbiamo trasformare $\langle X, f \rangle$ in un'istanza di 3SAT $\langle X', f' \rangle$ in modo tale che **f è soddisfacibile se e soltanto se f' è soddisfacibile**
- Procediamo in questo modo:
 - consideriamo una clausola c_j di f alla volta
 - prendiamo c_j e la trasformiamo nella congiunzione D_j di un insieme di clausole di f'
 - eventualmente, aggiungendo nuove variabili (non contenute in X) che faranno parte di un insieme Y
 - dove la struttura di D_j dipende dal numero di letterali di c_j
 - e dimostriamo che **se esiste una assegnazione di verità per X che soddisfa c_j allora è possibile assegnare un valore di verità alle variabili in Y in modo tale che tutte le clausole in D_j sono soddisfatte**
 - mentre **se nessuna assegnazione di verità per X soddisfa c_j allora non è possibile assegnare un valore di verità alle variabili in $Y \cup X$ in modo tale che tutte le clausole in D_j siano soddisfatte**

3SAT è NP-completo

- Caso 1: c_j contiene 1 letterale, ossia $c_j = \ell$, con $\ell = x_i$ o $\ell = \neg x_i$
- allora $D_j = (\ell \vee \mathbf{y_{j1}} \vee \mathbf{y_{j2}}) \wedge (\ell \vee \neg \mathbf{y_{j1}} \vee \mathbf{y_{j2}}) \wedge (\ell \vee \mathbf{y_{j1}} \vee \neg \mathbf{y_{j2}}) \wedge (\ell \vee \neg \mathbf{y_{j1}} \vee \neg \mathbf{y_{j2}})$
 - dove $\mathbf{y_{j1}}$ e $\mathbf{y_{j2}}$ sono due nuove variabili – ossia, $\mathbf{y_{j1}}, \mathbf{y_{j2}} \notin X$
 - se a ℓ viene assegnato valore **vero**, allora D_j assume valore **vero** *qualunque valore di verità si assegna a y_{j1} e y_{j2}*
 - invece, se a ℓ viene assegnato valore **falso**, allora D_j assume valore **falso** *qualunque valore di verità si assegna a y_{j1} e y_{j2}*
 - qualunque assegnazione di verità a y_{j1} e y_{j2} rende falsa una delle clausole in D_j
- Caso 2: c_j contiene 2 letterali, ossia $c_j = \ell_1 \vee \ell_2$,
- allora $D_j = (\ell_1 \vee \ell_2 \vee \mathbf{y_j}) \wedge (\neg \mathbf{y_j} \vee \ell_1 \vee \ell_2)$
 - dove $\mathbf{y_j}$ è una nuova variabile – ossia, $\mathbf{y_j} \notin X$
 - se viene assegnato valore **vero** a ℓ_1 oppure a ℓ_2 allora D_j assume valore **vero** *qualunque valore di verità si assegna a y_j*
 - se viene assegnato valore **falso** sia a ℓ_1 che a ℓ_2 allora D_j assume valore **falso** *qualunque valore di verità si assegna a y_j*

3SAT è NP-completo

- Caso 3: c_j contiene 3 letterali, ossia $c_j = \ell_1 \vee \ell_2 \vee \ell_3$,
- allora $D_j = c_j = \ell_1 \vee \ell_2 \vee \ell_3$,
 - è il caso più facile!
- Caso 4: c_j contiene 4 letterali, ossia $c_j = \ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4$,
- allora $D_j = (\ell_1 \vee \ell_2 \vee y_j) \wedge (\neg y_j \vee \ell_3 \vee \ell_4)$
 - dove y_j è una nuova variabile – ossia, $y_j \notin X$
 - se viene assegnato valore **vero** a ℓ_1 , oppure a ℓ_2 , oppure a ℓ_3 , oppure a ℓ_4 allora **esiste una assegnazione di verità a y_j che fa assumere a D_j valore vero**
 - se viene assegnato valore **falso** sia a ℓ_1 che a ℓ_2 che a ℓ_3 che a ℓ_4 allora D_j assume valore **falso qualunque valore di verità si assegni a y_j**
- Caso 5: c_j contiene 5 letterali, ossia $c_j = \ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4 \vee \ell_5$,
- allora $D_j = (\ell_1 \vee \ell_2 \vee y_{j1}) \wedge (\neg y_{j1} \vee \ell_3 \vee y_{j2}) \wedge (\neg y_{j2} \vee \ell_4 \vee \ell_5)$

3SAT è NP-completo

- Caso 6: c_j contiene 6 letterali, ossia $c_j = \ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4 \vee \ell_5 \vee \ell_6$,
- allora $D_j = (\ell_1 \vee \ell_2 \vee \mathbf{y_{j1}}) \wedge (\neg \mathbf{y_{j1}} \vee \ell_3 \vee \mathbf{y_{j2}}) \wedge (\neg \mathbf{y_{j2}} \vee \ell_4 \vee \mathbf{y_{j3}}) \wedge (\neg \mathbf{y_{j3}} \vee \ell_5 \vee \ell_6)$
- Caso ... capito il gioco?
- Caso generico: c_j contiene h letterali, ossia $c_j = \ell_1 \vee \ell_2 \vee \dots \vee \ell_h$,
- allora $D_j = (\ell_1 \vee \ell_2 \vee y_{j1}) \wedge (\neg y_{j1} \vee \ell_3 \vee y_{j2}) \wedge (\neg y_{j2} \vee \ell_4 \vee y_{j3}) \wedge (\neg y_{j3} \vee \ell_5 \vee y_{j4}) \wedge \dots \wedge (\neg y_{jh-4} \vee \ell_{h-2} \vee y_{jh-3}) \wedge (\neg y_{jh-3} \vee \ell_{h-1} \vee \ell_h)$
 - dove $y_{j1}, y_{j2}, \dots, y_{jh-3}$ sono nuove variabile – ossia, non appartengono a X
 - se viene assegnato valore **vero** a ℓ_1 , oppure a ℓ_2 , ..., oppure a ℓ_h allora **esiste una assegnazione di verità alle variabili $y_{j1}, y_{j2}, \dots, y_{jh-3}$** che fa assumere a D_j valore **vero**
 - se viene assegnato valore **falso** sia a ℓ_1 che a ℓ_2 ... che a ℓ_h allora D_j assume valore **falso qualunque valore di verità si assegni a $y_{j1}, y_{j2}, \dots, y_{jh-3}$**
 - infatti: per soddisfare $(\ell_1 \vee \ell_2 \vee y_{j1})$ occorre assegnare a y_{j1} il valore **vero**, da cui segue che occorre assegnare a y_{j2} il valore **vero**, ..., da cui segue che occorre assegnare a y_{jh-3} il valore **vero**, ma, a questo punto $(\neg y_{jh-3} \vee \ell_{h-1} \vee \ell_h)$ assume il valore **falso**!

3SAT è NP-completo

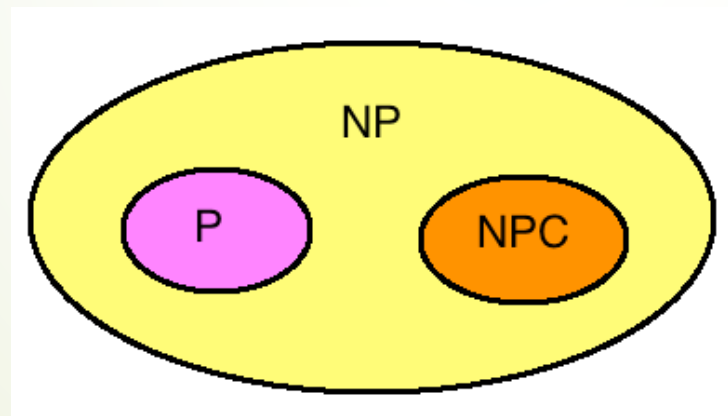
- Ricapitolando: abbiamo costruito un'istanza $\langle X', f' \rangle$ di 3SAT a partire da un'istanza $\langle X, f \rangle$ di SAT
 - se $f = c_1 \wedge c_2 \wedge \dots \wedge c_m$ allora $f' = D_1 \wedge D_2 \wedge \dots \wedge D_m$, e $X' = X \cup Y$
- e (passo passo) abbiamo dimostrato che f è soddisfacibile se e soltanto se f' è soddisfacibile
 - perché abbiamo dimostrato che **la soddisfacibilità di D_j non dipende dai valori di verità che assegniamo alle variabili in Y**
- e, poiché occorre tempo polinomiale in $|\langle X, f \rangle|$ per costruire $\langle X', f' \rangle$
 - perché per costruire ciascuna D_j occorrono $O(|X|)$ passi
 - e devono essere costruite $O(|f|)$ congiunzioni di clausole D_j
- abbiamo dimostrato che 3SAT è NP-completo

La struttura di NP

- A partire dal teorema di Cook-Levin
 - che ha individuato in SAT il capostipite dei problemi NP-completi
- ed utilizzando il teorema 9.3
- uno dopo l'altro sono stati individuati tanti, tantissimi, problemi NP-completi
 - una miriade di problemi NP-completi!
 - e ne vedremo un po' nel corso delle prossime lezioni
- A questo punto, la domanda sorge spontanea: e chi ce lo dice che tutti i problemi in NP non sono altrettanto "difficili" di SAT?
- Ossia: non sarà, magari, che tutti i problemi in NP sono NP-completi?
- Beh, insomma, questa questione va almeno posta diversamente, perché, ricordiamo, $P \subseteq NP$
- e, quindi, **se crediamo che sia $P \neq NP$, almeno i problemi in P**
che appartengono anch'essi a NP
non possono essere NP completi!
 - Altrimenti, per il corollario 6.4 sarebbe $P = NP$

La struttura di NP

- Allora, dobbiamo riformulare la nostra domanda: **se crediamo che sia $P \neq NP$, non sarà, magari, che tutti i problemi in NP-P sono NP-completi?**
- La risposta è no, e a dimostrarlo è il seguente
- **Teorema di Ladner: se $P \neq NP$ allora esiste un problema in NP-P che non è NP-completo**
 - la (bellissima) dimostrazione del teorema di Ladner non la studiamo
- Allora, **nell'ipotesi $P \neq NP$** , alla luce del teorema di Cook-Levin e del teorema di Ladner, la struttura della classe NP, è quella illustrata nella seguente figura



NPC contiene i problemi NP-completi

ATTENZIONE!!!!

- I problemi in NP-P che non sono NP-completi si dicono **NP-intermedi**
- Come si fa a dimostrare che un problema è NP-intermedio?
- Risposta: non si fa!
- Perché, se si dimostrasse che un problema è NP-intermedio questo vorrebbe dire che si sarebbe dimostrato che quel teorema è in NP-P
- ossia, si sarebbe dimostrato che **$P \neq NP$**
 - e vinto il milione di dollari
- Chiaro il punto?
- **Perciò: se avete un problema che dimostrate che appartiene a NP**
 - ma non riuscite a deciderlo mediante un algoritmo (deterministico) polinomiale
 - e non riuscite nemmeno a dimostrare che è NP-completo
- **non vi venga in mente di concludere che quel problema è NP-intermedio!**

La struttura di NP e coNP

- La seconda congettura della teoria della complessità afferma che $NP \neq coNP$
- Ricordando il **teorema 6.25** che ci dice che:

un linguaggio L è NP-completo se e soltanto se
il suo complemento L^c è coNP-completo

- e che, quindi, esistono sia problemi coNP-completi che coNP-intermedi
- e ricordando che, poiché $P = coP$, allora $P \subseteq coNP$
- possiamo riassumere la struttura delle classi P, NP e coNP nella seguente figura

