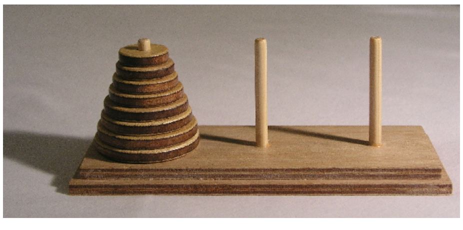
TEORIA DELLA COMPLESSITÀ

Decidibilità e Trattabilità

Ci sono problemi (problema dell'arresto) che non possono essere risolti da nessun calcolatore, indipendentemente dal tempo a disposizione

→ problemi indecidibili

Ci sono problemi decidibili che possono richiedere tempi di risoluzione esponenziali nella dimensione dell'istanza (torri di Hanoi, generazione delle sequenze binarie e delle permutazioni)



→ problemi intrattabili

Decidibilità e Trattabilità

Ci sono problemi che possono essere risolti con algoritmi di costo polinomiale

(ordinamento; ricerca di chiavi in array, liste, alberi; problemi su grafi: OT di DAG, connettività, ricerca di cicli, ricerca di un ciclo euleriano)

→ problemi trattabili («facili»)

Ci sono infine problemi il cui stato non è noto

(clique, cammino hamiltoniano):

- Abbiamo a disposizione solo algoritmi di costo esponenziale
- Nessuna ha dimostrato che che non possano esistere algoritmi di costo polinomiale

→ problemi presumibilmente intrattabili

Algoritmi polinomiali e esponenziali

Studiamo la dimensione dei dati trattabili in funzione dell'incremento della velocità dei calcolatori

Calcolatori: C_1, C_2 (k volte più veloce di C_1)

Tempo di calcolo a disposizione: t

n_1 = dati trattabili nel tempo t su C_1

n_2 = dati trattabili nel tempo t su C_2

Osservazione:

usare C_2 per un tempo t, equivale a usare C_1 per un tempo k * t

Algoritmi polinomiali e esponenziali

Algoritmo polinomiale che risolve il problema in $c\,n^s$ secondi (c, s costanti)

C_1: $c\,n_1^s = t \rightarrow n_1 = (t/c)^{1/s}$

C_2: $c\,n_2^s = k\,t \rightarrow n_2 = (k\,t/c)^{1/s} = k^{1/s}(t/c)^{1/s}$

$$n_2 = k^{1/s}\,n_1$$

Miglioramento di un fattore moltiplicativo $k^{1/s}$

- $k = 10^9$, e $s = 3$ → possiamo moltiplicare per 10^3 i dati trattabili a pari tempo di calcolo
- $k = 10^9$, e $s = 1$ → possiamo moltiplicare per 10^9 i dati trattabili a pari tempo di calcolo

Algoritmi polinomiali e esponenziali

Algoritmo esponenziale che risolve il problema in $c\,2^n$ secondi (c costante)

C_1: $c\,2^{n1} = t \rightarrow 2^{n1} = t/c$

C_2: $c\,2^{n2} = k\,t \rightarrow 2^{n2} = k\,t/c = k\,2^{n1}$

$$n_2 = n_1 + \log_2 k$$

Miglioramento di un fattore additivo $\log_2 k$

- $k = 10^9$ → possiamo solo sommare $\log_2 10^9 \sim 30$ al numero di dati trattabili a pari tempo di calcolo

Problemi

Problema Π

I: insieme delle istanze in ingresso

S: insieme delle soluzioni

Tipologie di problemi

Problemi decisionali

- Richiedono una risposta binaria ($S = \{0,1\}$)
- Es: Un grafo è connesso? Un numero è primo?
- Istanze positive (accettabili): $x \in I$, t.c. $\Pi(x) = 1$
- Istanze negative: $x \in I$, t.c. $\Pi(x) = 0$

Problemi di ricerca

- Data un'istanza x, richiedono di restituire una soluzione s

  Trovare un cammino tra due vertici, trovare il mediano di un insieme di elementi

Tipologie di problemi

Problemi di ottimizzazione

- Data un'istanza x, si vuole trovare la migliore soluzione s tra tutte le soluzioni possibili
- Ricerca della clique di dimensione massima, ricerca del cammino minimo fra due nodi di un grafo

Problemi decisionali

La teoria della complessità computazionale è definita principalmente in termini di problemi di decisione

- Essendo la risposta binaria, non ci si deve preoccupare del tempo richiesto per restituire la soluzione e tutto il tempo è speso esclusivamente per il calcolo

- La difficoltà di un problema è già presente nella sua versione decisionale

Problemi decisionali

- Molti problemi di interesse pratico sono però problemi di ottimizzazione
- È possibile esprimere un problema di ottimizzazione in forma decisionale, chiedendo l'esistenza di una soluzione che soddisfi una certa proprietà.

ESEMPIO:

- MAX-CLIQUE: trovare la CLIQUE più grande in un grafo G
- CLIQUE: Esiste una clique in G di almeno k vertici?
- CLIQUE **non è più difficile** di MAX-CLIQUE:

  Se sappiamo trovare la CLIQUE più grande in G, ne confrontiamo la dimensione con k, e risolviamo anche il problema decisionale

Problemi decisionali

Il problema di ottimizzazione è quindi almeno tanto difficile quanto il corrispondente problema decisionale

Caratterizzare la complessità del problema decisionale permette quindi di dare almeno una limitazione inferiore alla complessità del problema di ottimizzazione

Classi di complessità

Dato un problema decisionale Π ed un algoritmo A, diciamo che A risolve Π se, data un'istanza di input x

$$A(x) = 1 \quad \Leftrightarrow \quad \Pi(x) = 1$$

A risolve Π in tempo t(n) e spazio s(n) se il tempo di esecuzione e l'occupazione di memoria di A sono rispettivamente t(n) e s(n)

Classi Time e Space

Data una qualunque funzione f(n)

Time(f(n))
: insiemi dei problemi decisionali che possono essere risolti in tempo O(f(n))

Space(f(n))
: insiemi dei problemi decisionali che possono essere risolti in spazio O(f(n))

Classe P

Algoritmo polinomiale (tempo)

esistono due costanti $c, n_0 > 0$ t.c. il numero di passi elementari è al più n^c per ogni input di dimensione n e per ogni $n > n_0$

Classe P

è la classe dei problemi risolvibili in tempo polinomiale nella dimensione n dell'istanza di ingresso

Classe PSPACE

Algoritmo polinomiale (spazio)

esistono due costanti $c, n_0 > 0$ t.c. il numero di celle di memoria utilizzate è al più n^c per ogni input di dimensione n e per ogni $n > n_0$

Classe PSPACE

è la classe dei problemi risolvibili in spazio polinomiale nella dimensione n dell'istanza di ingresso

Classe EXP-TIME

La classe Exp Time è la classe dei problemi risolvibili in tempo esponenziale nella dimensione n dell'istanza di ingresso

Relazioni tra le classi

$P \subseteq PSpace$

infatti un algoritmo polinomiale può avere accesso al più ad un numero polinomiale di locazioni di memoria diverse (in ordine di grandezza)

$PSpace \subseteq ExpTime$

Relazioni

- Non è noto (ad oggi) se le inclusioni siano proprie
- L'unico risultato di separazione dimostrato finora riguarda P e ExpTime

  Esiste un problema che può essere risolto in tempo esponenziale, ma per cui tempo polinomiale non è sufficiente

  Torri di Hanoi

Altri problemi interessanti:

Zaino

Clique

Cammino Hamiltoniano

Soddisfacibilità di formule booleane (SAT)

Algoritmo per CLIQUE

- Si considerano tutti i sottoinsiemi di vertici, in ordine di cardinalità decrescente, e si verifica se formano una clique di dimensione almeno k.
- Se n è il numero di vertici, quanti diversi sottoinsiemi esamina l'algoritmo al caso peggiore?

2^n

CLIQUE $\in$ ExpTime

Algoritmo polinomiale non noto!

Algoritmo per Cammino Hamiltoniano

- Si considerano tutte le permutazioni di vertici, e si verifica se i vertici in quell'ordine sono a due a due adiacenti
- Se n è il numero di vertici, quante diverse permutazioni esamina l'algoritmo al caso peggiore?

$n!$

CamminoHamiltoniano $\in$ ExpTime

Algoritmo polinomiale non noto!

SAT

Insieme V di variabili Booleane

- Letterale: variabile o sua negazione
- Clausola: disgiunzione (OR) di letterali

Un'espressione Booleana su V si dice in forma normale congiuntiva (FNC) se è espressa come congiunzione di clausole (AND di OR di letterali)

Esempio

$$V = \{x, y, z, w\}$$

$$FNC : (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee w) \wedge y$$

SAT

Data una espressione in forma normale congiuntiva

verificare se esiste una assegnazione di valori di verità alle variabili che rende l'espressione vera

Esempio

La formula

$$(x \vee \overline{y} \vee z) \wedge (\overline{x} \vee w) \wedge y$$

è soddisfatta dall'assegnazione

$$x = 1 \quad y = 1 \quad z = 0 \quad w = 1$$

Algoritmo per SAT

Si considerano tutti i 2^n assegnamenti di valore alle n variabili, e per ciascuno si verifica se la formula è vera

SAT $\in$ ExpTime

Algoritmo polinomiale non noto!

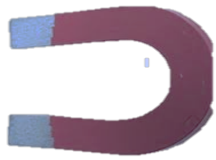
Clique, Cammino Hamiltoniano, SAT

La ricerca esaustiva è necessaria?

Non lo sappiamo

Cercare un ago in un pagliaio

La ricerca esaustiva è necessaria?



No, se si ha a disposizione un magnete…

Problemi decisionali e certificati

In un problema decisionale siamo interessati a verificare se una istanza del problema soddisfa una certa proprietà

esiste una clique di k vertici?

esiste un cammino hamiltoniano?

esiste un assegnamento di valori che rende vera la formula?

Per alcuni problemi, per le istanze accettabili (positive) x è possible fornire un certificato y che possa convincerci del fatto che l'istanza soddisfa la proprietà e dunque è un'istanza accettabile

Certificato

Certificato per CLIQUE

sottoinsieme di k vertici, che forma la clique

Certificato per Cammino Hamiltoniano

permutazione degli n vertici che definisce un cammino semplice

Certificato per SAT

Un'assegnazione di verità alle variabili che renda vera l'espressione

Certificato

Un certificato è un attestato breve di esistenza di una soluzione con determinate proprietà

Si definisce solo per le istanza accettabili

Infatti, in generale, non è facile costruire attestati di non esistenza

Certificato

UNSAT

È vero che nessun assegnamento di valore alle variabili rende vera l'espressione?

Certificato per UNSAT ?

Non è sufficiente esibire un'assegnazione di valori di verità alle variabili…

In questo caso è difficile esprimere anche un certificate che sia 'breve'!

Verifica

IDEA: utilizzare il costo della verifica di un certificato (una soluzione) per un'istanza accettabile (positiva) per caratterizzare la complessità del problema stesso

Un problema Π è verificabile in tempo polinomiale se

1. Ogni istanza accettabile x di Π di lunghezza n ammette un certificato y di lunghezza polinomiale in n
2. Esiste un algoritmo di verifica polinomiale in n e applicabile a ogni coppia <x,y>, che permette di attestare che x è accettabile

Classe NP

NP è la classe dei problemi decisionali verificabili in tempo polinomiale

Cosa vuol dire NP?

P sta per polinomiale, ma N?

N non vuol dire NON…

La classe NP è la classe dei problemi risolvibili in tempo *polinomiale non deterministico*

Osservazioni

Un certificato contiene un'informazione molto prossima alla soluzione, quindi qual è l'interesse di questa definizione?

Dubbio legittimo

- La teoria della verifica è utile per far luce sulle gerarchie di complessità dei problemi, non aggiunge nulla alla possibilità di risolverli efficientemente
- Chi ha una soluzione può verificare in tempo polinomiale che l'istanza è accettabile.
- Chi non ha una soluzione (certificato), può individuarla in tempo esponenziale considerando tutti i casi possibili con una ricerca esaustiva

Le classi P e NP

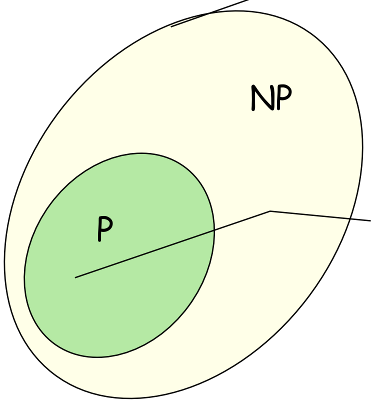
P è incluso in NP oppure no?

Ovviamente sì!

Ogni problema in P ammette un certificato verificabile in tempo polinomiale….come mai?

- Eseguo l'algoritmo che risolve il problema per costruire il certificato!

Le classi P e NP



Problemi facilmente verificabili

Clique, SAT, Zaino
Cammino Hamiltoniano

Problemi facilmente risolvibili

Test di primalità, ciclo euleriano, ricerca

P = NP oppure P ≠ NP ?

Le classi P e NP

- Dobbiamo per forza fare la ricerca esaustiva quando abbiamo un problema come i precedenti?

  Non lo sappiamo

- Si congettura che $P \neq NP$

- È possibile individuare i problemi più difficili all'interno della classe NP, ovvero quelli candidati ad appartenere a NP se $P \neq NP$

Problemi NP-completi

Sono i problemi più difficili all'interno della classe NP

- Se esistesse un algoritmo polinomiale per risolvere uno solo di questi problemi, allora
  - tutti i problemi in NP potrebbero essere risolti in tempo polinomiale, e dunque $P = NP$
- Quindi:

  tutti i problemi NP-completi sono risolvibili in tempo polinomiale oppure nessuno lo è

Riduzioni polinomiali

Π_1 e Π_2 = problemi decisionali

I_1 e I_2 = insiemi delle istanze di input di Π_1 e Π_2

Π_1 si riduce in tempo polinomiale a Π_2

$$\Pi_1 \leq_p \Pi_2$$

se esiste una funzione $f: I_1 \rightarrow I_2$ calcolabile in tempo polinomiale tale che, per ogni istanza x di Π_1

x è un'istanza accettabile di Π_1

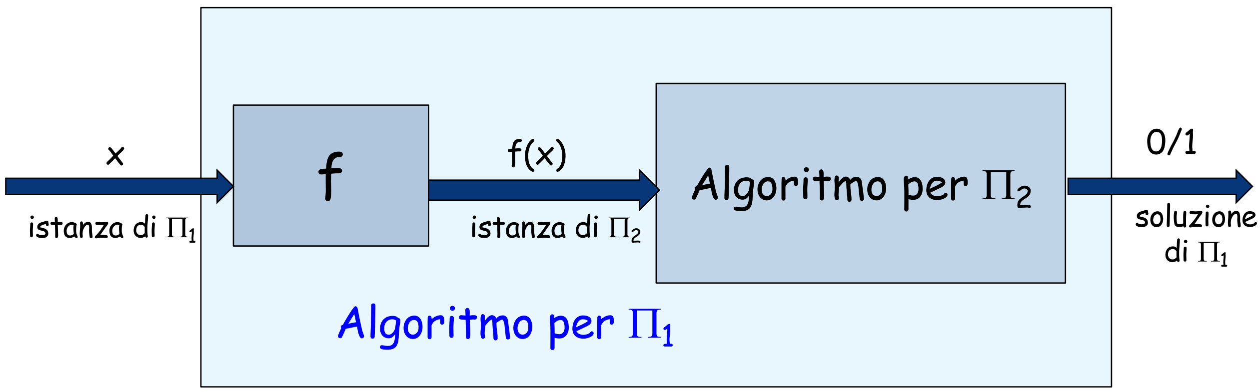
SE E SOLO SE

f(x) è un'istanza accettabile di Π_2

Riduzioni polinomiali

Se esistesse un algoritmo per risolvere Π_2 potremmo utilizzarlo per risolvere Π_1

$$\Pi_1 \leq_p \Pi_2 \quad e \quad \Pi_2 \in P \quad \Rightarrow \quad \Pi_1 \in P$$



Problemi NP ardui

Un problema Π si dice NP-arduo se

per ogni $\Pi' \in NP$, $\Pi' \leq_p \Pi$

Problemi NP completi

Un problema decisionale Π si dice NP-completo se

$\Pi \in NP$

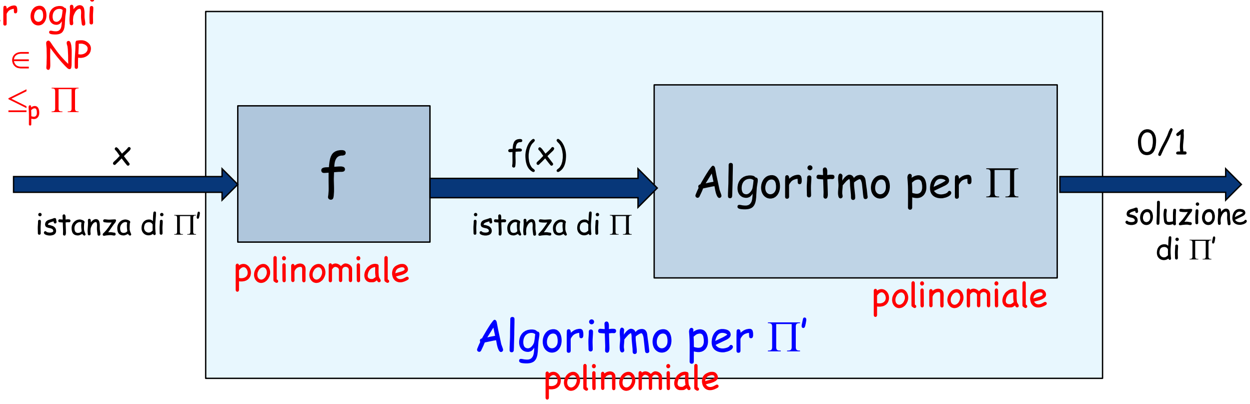
per ogni $\Pi' \in NP$, $\Pi' \leq_p \Pi$

Problemi NP-completi

Sia Π un problema NP-completo

Se $\Pi \in P$, allora P = NP

Per ogni
$\Pi' \in NP$
$\Pi' \leq_p \Pi$



Se un problema NP-completo è facile, allora tutti i problemi in NP sono facili!

Problemi NP completi

Dimostrare che un problema è in NP può essere facile

Basta esibire un certificato polinomiale

Non è altrettanto facile dimostrare che un problema Π è NP-arduo o NP-completo

- Bisogna dimostrare che TUTTI i problemi in NP si riducono polinomialmente a Π
- In realtà la prima dimostrazione di NP-completezza aggira il problema

Teorema di Cook (1971)

TEOREMA

SAT è NP completo

Teorema di Cook (idea)

Cook ha mostrato che

dati un qualunque problema Π in NP
ed una qualunque istanza x per Π

si può costruire una espressione Booleana in forma normale congiuntiva che descrive il calcolo di un algoritmo per risolvere Π su x

l'espressione è vera se e solo se l'algoritmo restituisce 1

Problemi NP completi

Un problema decisionale Π è NP-completo se

$\Pi \in NP$

$SAT \leq_p \Pi$

(o un qualsiasi altro problema NP-completo)

Infatti:

per ogni $\Pi' \in NP$, $\Pi' \leq_p SAT$ e $SAT \leq_p \Pi$

Quindi $\Pi' \leq_p \Pi$

CLIQUE è NP completo

$SAT \leq_p CLIQUE$

data un'espressione booleana F in forma normale congiuntiva con k clausole

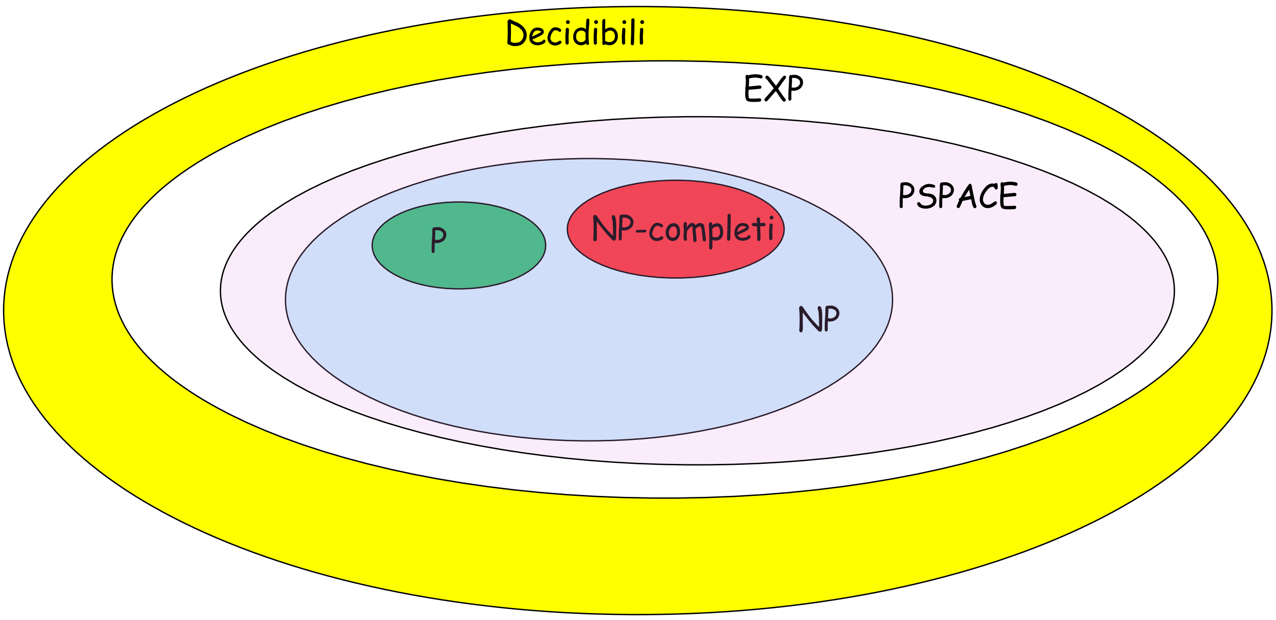
è possibile costruire in tempo polinomiale

un grafo G che contiene una **clique di k vertici se e solo se F è soddisfacibile**.

Problemi NP equivalenti

- SAT $\leq_p$ CLIQUE $\Rightarrow$ CLIQUE è NP completo
- SAT è NP completo $\Rightarrow$ CLIQUE $\leq_p$ SAT
- SAT e CLIQUE sono **NP equivalenti.**
- Tutti i problemi NP completi sono tra loro **NP equivalenti**.
- Sono tutti facili, o tutti difficili

Gerarchia delle classi



Altri famosi problemi NP-completi

Copertura di vertici

- Una copertura di vertici (*vertex cover*) di un grafo $G=(V,E)$ è un insieme di vertici $C \subseteq V$ tale che per ogni $(u,v) \in E$, almeno uno tra u e v appartiene a C
- Dati G e un intero k, verificare se esiste una copertura di vertici di G di dimensione al più k

Altri famosi problemi NP-completi

Commesso viaggiatore

Dati un grafo completo G con pesi w sugli archi ed un intero k, verificare se esiste un ciclo di peso al più k che attraversa ogni vertice una ed una sola volta

Colorazione

Dati un grafo G ed un intero k, verificare se è possibile colorare i vertici di G con al più k colori tali che due vertici adiacenti non siano dello stesso colore

Altri famosi problemi NP-completi

Somme di sottoinsiemi

Dati un insieme S di numeri naturali ed un intero t, verificare se esiste un sottoinsieme di S i cui elementi sommano esattamente a t

Zaino

Dati un intero *k*, uno zaino di capacità c, e *n* oggetti di dimensioni $s_1, \ldots, s_n$ cui sono associati profitti $p_1, \ldots, p_n$, verificare se esiste un sottoinsieme degli oggetti di dimensione **al più** *c* che garantisca profitto **almeno** *k*

Problemi di ottimizzazione NP- hard

Se la soluzione ottima è troppo difficile da ottenere, una soluzione quasi ottima ottenibile facilmente forse è buona abbastanza

- A volte, avere una soluzione esatta non è strettamente necessario
- Ci si accontenta di una soluzione che
  - non si discosti troppo da quella ottima
  - Si possa calcolare in tempo polinomiale

Le classi co-P e co-NP

Profonda differenza tra certificare l'esistenza o la non-esistenza di una soluzione

Es: Problema del Ciclo Hamiltoniano:

- una permutazione di vertici (per certificare esistenza)
- per la non-esistenza è difficile dare un certificato polinomiale che indichi direttamente questa proprietà

Π: problema decisionale

coΠ: problema complementare

(accetta tutte e sole le istanze rifiutate da Π)

La classe co-P

È la classe dei problemi decisionali Π per cui

$$\text{co}\Pi \in P$$

P = co-P

(risolvo il problema, e complemento il risultato)

La classe co-NP

È la classe dei problemi decisionali Π per cui

$$co\Pi \in NP$$

NP ? co-NP

- si congettura che le due classi siano diverse
- se questa congettura è vera, allora $P \neq NP$