Classificazione dei problemi

Problema vs Algoritmi

- **Complessità di un algoritmo:** misura del numero di passi che si devono eseguire per risolvere il problema
- **Complessità di un problema:** complessità del migliore algoritmo che lo risolve

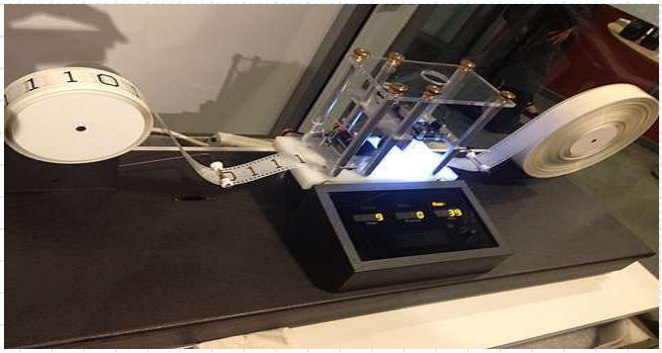
Classi di complessità

- Classi di problemi risolubili utilizzando una certa quantità di risorse (per esempio di tempo)
- **Problemi decidibili** = hanno una soluzione algoritmica
- **Problemi indecidibili** = **non** hanno una soluzione algoritmica

3

Problemi indecidibili

- la **tesi di Church-Turing** è un'ipotesi che afferma: la classe delle funzioni calcolabili coincide con quella delle funzioni calcolabili da una macchina di Turing



4

Problemi indecidibili

- Turing ha dimostrato che riuscire a dimostrare se un programma arbitrario si arresta e termina la sua esecuzione non è solo un'impresa ardua, ma in generale è IMPOSSIBILE!
- Non può esistere un algoritmo che decida in tempo finito se un algoritmo arbitrario A termina la sua computazione su dati arbitrari D.

  Ciò non significa che non si possa decidere in tempo finito la terminazione di algoritmi particolari

TEOREMA: Il problema dell'arresto è INDECIDIBILE.

5

Problema dell'arresto (esempio)

La **congettura di Goldbach** è uno dei più vecchi problemi irrisolti nella teoria dei numeri. Essa afferma che ogni numero pari maggiore di 2 può essere scritto come somma di due numeri primi (che possono essere anche uguali).

Per esempio,
4 = 2 + 2
6 = 3 + 3
8 = 3 + 5
10 = 3 + 7 = 5 + 5
12 = 5 + 7
14 = 3 + 11 = 7 + 7

6

Problema dell'arresto (esempio)

```
Goldbach()
n = 2;
do {
   n = n + 2;
   controesempio = true;
   for (p = 2; p ≤ n - 2; p++) {
    q = n – p;
    if (Primo(p) && Primo(q)) {
                    controesempio = false;
                    }
   } }
while (!controesempio);
return n;.
```

Cerco tutte le coppie per cui p+q=n

7

Problema dell'arresto (esempio)

Congettura falsa: Se
il programma Goldbach() si arresta

Congettura vera: Se
il programma Goldbach() NON si arresta

L'algoritmo dell'ARRESTO costituirebbe dunque uno strumento estremamente potente ma è INDECIDIBLE

Servirebbe un algoritmo che si applica ad un altro algoritmo e sulla base di algoritmo e input, dice "Sì, si ferma" oppure "No, non si ferma".

8

PROBLEMI INDECIDIBILI: Non ne possiamo analizzare la complessità

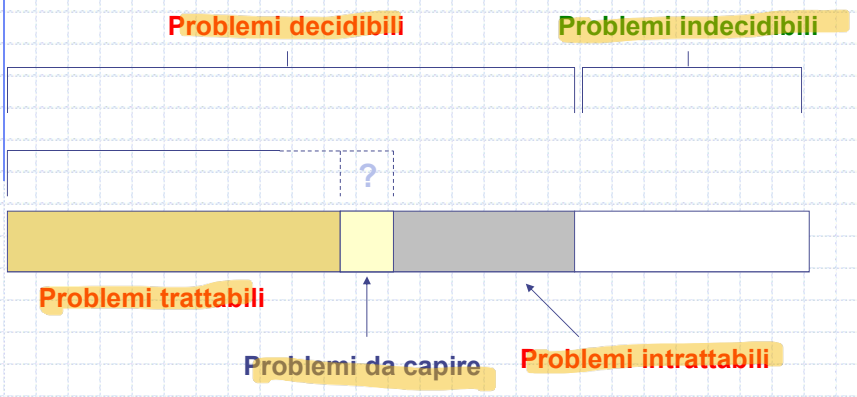
Classificazione di problemi

(decidibili)

- Possiamo classificare i problemi in base alla quantità di risorse necessarie per ottenere la soluzione
- Per certi gruppi di problemi, le difficoltà incontrate per trovare un algoritmo efficiente sono sostanzialmente le stesse
- Possiamo raggruppare i problemi in tre categorie:
  - I problemi che ammettono algoritmi di soluzione efficienti: **trattabili**;
  - I problemi che per loro natura non possono essere risolti mediante algoritmi efficienti e che quindi sono **intrattabili**;
  - I problemi per i quali algoritmi efficienti non sono stati trovati ma per i quali nessuno ha finora provato che tali algoritmi non esistano

9

Classificazione dei problemi



10

Classe di complessità P

(Prob. trattabili)

Problemi polinomiali =

problemi per i quali esistono soluzioni praticabili, cioè di complessità in Θ(f(n)) dove f(n) è un **polinomio** oppure è limitato superiormente da un polinomio

- Esempi: ordinamento di una lista, ricerca in una lista
- Sono problemi **trattabili** = ammettono un algoritmo di soluzione efficiente

11

Problemi intrattabili

- Problemi che ***non*** possono essere risolti in un tempo polinomiale (quindi non appartengono alla classe P)
- Per questi problemi si può provare che ogni algoritmo risolutivo richiede, nel caso peggiore, un tempo di calcolo **esponenziale** o comunque asintoticamente superiore ad ogni polinomio
- Quindi pur essendo risolubili per via automatica, richiedono un tempo di calcolo molto elevato, tale da rendere ogni algoritmo di fatto inutilizzabile anche per dimensioni piccole dell'input

12

Problemi intrattabili

Dimensione del problema

Complessità algoritmo risolutivo		10	20	30	40	50	60
Algoritmi polinomiali	n	$10x10^{-6}$	$20x10^{-6}$	$30x10^{-6}$	$40x10^{-6}$	$50x10^{-6}$	$60x10^{-6}$
	n^2	$1x10^{-4}$	2^2x10^{-4}	3^2x10^{-4}	4^2x10^{-4}	5^2x10^{-4}	6^2x10^{-4}
	n^3	$1x10^{-3}$	2^3x10^{-3}	3^3x10^{-3}	4^3x10^{-3}	5^3x10^{-3}	6^3x10^{-3}
	n^5	$1x10^{-1}$	2^5x10^{-1} =3,2 s	3^5x10^{-1} = 24,3 s	4^3x10^{-3} = 1,7 m	5^3x10^{-3} = 5,2 m	6^3x10^{-3} = 13,0 m
Algoritmi esponenziali	2^n	10^{-3} s	1,0 s	17,9 min	12,7 giorni	35,7 anni	366 secoli
	3^n	0,059 s	58 min	6,5 anni	3855 secoli	$2,0x10^8$ secoli	$1,3X10^{13}$ secoli

Tabella di Garey and Johnson (1979)

Si assume che la singola operazione viene eseguita in 10^{-6} secondi

Classe di complessità NP

- Un problema è detto di classe **NP** se, data una possibile soluzione, esiste un algoritmo polinomiale nelle dimensioni *n* del problema in grado di verificare se questa è effettivamente soluzione del problema.
- Ovviamente la classe **NP** è più generale della classe **P** e la contiene.
- **Problemi polinomiali non deterministici** = problemi risolvibili in tempo **polinomiale** da un **algoritmo non deterministico**, ma per i quali non è ancora stata trovata una soluzione deterministica in tempo polinomiale
- Algoritmo non deterministico = si basa sulla creatività del meccanismo che esegue il programma

Classe di complessità NP

Problemi decisionali

La teoria della complessità computazionale o della "NP completezza" (NP-Completeness) punta ad un'ulteriore classificazione dei problemi facendo riferimento ai cosiddetti problemi decisionali.

Un problema decisionale è un problema formulato in modo da ammettere due sole possibili risposte: SI o NO.

Esempio:
Dato un numero intero K, esiste una coppia di numeri interi m ed n (con m,n>1) tali che K=mn ?

15

Problema del commesso viaggiatore: TSP (classe NP)

Un commesso viaggiatore deve visitare alcuni suoi clienti in città diverse senza superare il budget per le spese di viaggio: il suo problema è trovare un percorso (che parta dalla sua abitazione, arrivi nelle varie città da visitare e poi lo riconduca a casa) la cui lunghezza totale non superi i chilometri consentiti

16

Problema del commesso viaggiatore: TSP (classe NP)

- Soluzione tipica:
  - Si considerano i potenziali percorsi in modo sistematico confrontando la lunghezza di ogni percorso con il limite chilometrico finché si trova un percorso accettabile oppure sono state considerate tutte le possibilità
- **Non** è una soluzione in tempo polinomiale
  - Il numero dei tragitti da considerare aumenta più velocemente di qualsiasi polinomio al crescere del numero delle città **(n!)**

17

Problema del commesso viaggiatore: TSP (classe NP)

- **Algoritmo non deterministico**
  - Se esiste un percorso accettabile e lo selezioniamo per primo, l'algoritmo termina velocemente

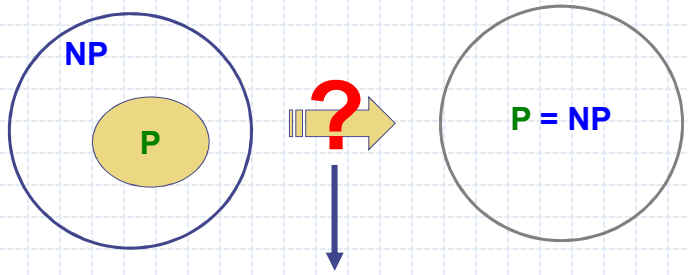
→ Trovo un algoritmo che non mi garantisce di arrivare alla soluz. in tempo polinomiale.
(Non c'è una procedura da seguire: genero soluz. e verifico)

18

Algoritmo non deterministico: per lo stesso input può esibire comportamenti diversi su corse diverse.

Relazione fra P e NP

- Tutti i problemi in P sono anche in NP



- Tutti i problemi in NP sono anche in P ?

19

Tempo di trasformaz. = polinomiale

Problemi NP-completi

- Dati ora due problemi R e Q, si dice che R si riduce a Q (e si indica con RμQ) se esiste un algoritmo polinomiale che associa a ogni istanza di R un'istanza di Q in modo tale che la soluzione dell'istanza di Q fornisce la soluzione della corrispondente istanza di R.
- Un problema $Q \in NP$ si dice NP-completo (o appartenente alla classe NP-completa) se RμQ per ogni R∈NP.
- In questo senso i problemi NP-completi sono i problemi più difficili della classe NP.

20

Se riesco a risolvere un problema della classe NP-completa risolvo tutti i prob. NP con un costo polinomiale.

Problemi NP-completi

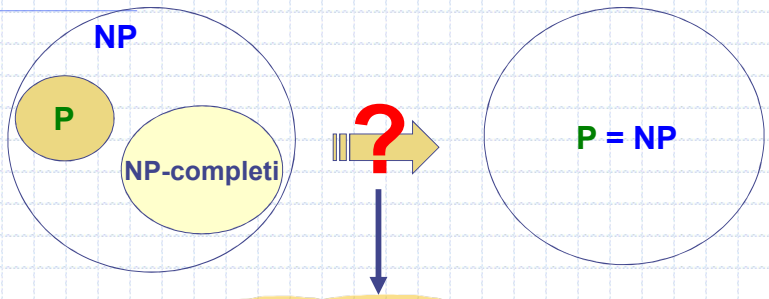
- In altri termini i Problemi NP-completi sono i problemi in NP con la seguente proprietà:

  una soluzione deterministica polinomiale in termini di tempo per ognuno di essi fornirebbe anche una soluzione polinomiale per tutti gli altri problemi in NP

  Esempi: commesso viaggiatore, problema dello zaino, etc.

21

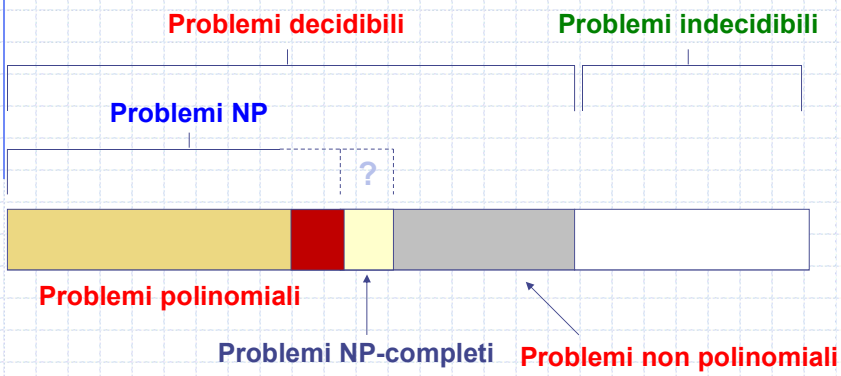
Relazione fra P e NP



- Se si dimostra che anche **solo** per **uno** dei problemi NP-completi esiste una soluzione deterministica polinomiale allora **P e NP sono equivalenti**

22

Classificazione dei problemi



23

Problemi di ottimizzazione

Ogni problema di ottimizzazione può essere formulato come problema decisionale.

Problema **decisionale** del TSP

Date n città, esiste un circuito che un commesso viaggiatore può effettuare per visitare ciascuna città una sola volta che risulti di lunghezza non superiore ad un valore B (≤B)?

Problema di **ottimizzazione** del TSP

Date n città, si individui il circuito di lunghezza minima che un commesso viaggiatore deve effettuare per visitare ciascuna città una sola volta.

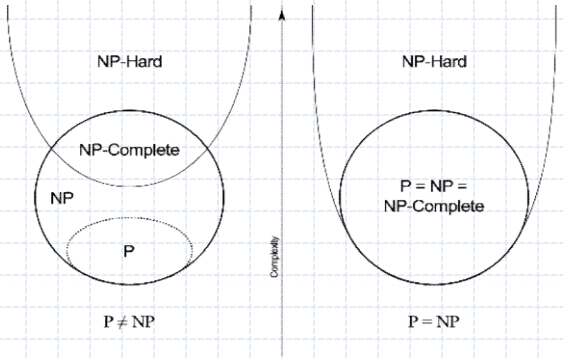
Poiché per risolvere un problema di ottimizzazione si devono risolvere più problemi di decisione ad esso associati, si può concludere che **la complessità di un problema di ottimizzazione sarà almeno pari a quello del problema di decisione ad esso associato**

problema di ottimizzazione la cui versione decisionale appartiene alla classe dei problemi NP-completi è detto NP-hard (NP-difficile).

24

Problemi di ottimizzazione

Un problema di ottimizzazione la cui versione decisionale appartiene alla classe dei problemi NP-completi è detto **NP-hard.**



25

Problemi di ottimizzazione
Conclusioni

La teoria della NP-completezza consente di classificare nella classe dei problemi NP e NP-completo, problemi che non risultano né trattabili, né intrattabili.

Anche se non è stato dimostrato formalmente si adotta la congettura (l'ipotesi) P≠NP, ovvero si ritiene che i problemi NP siano intrattabili.

Ne consegue che i problemi NP vanno trattati, in pratica, come se fossero intrattabili. Pertanto poiché i tempi di calcolo di algoritmi risolutivi esatti applicati a questi problemi risultano esponenziali, nelle applicazioni pratiche bisogna ricorrere ad algoritmi di risoluzione euristici.

26

Ringraziamenti

Questi lucidi sono un adattamento del materiale preparato dal prof.ssa M. Federico per il corso di Algoritmi e Strutture dati

27