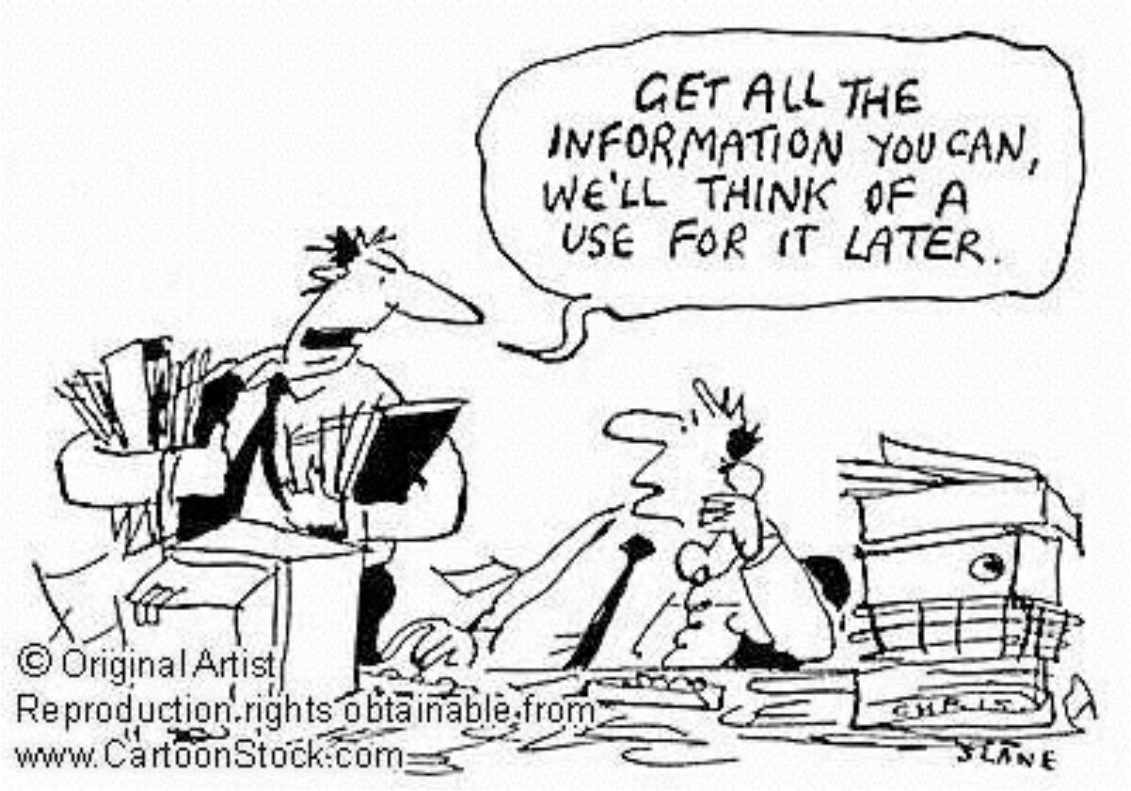
Automi, Complessità e Calcolabilità
Prof. E. Fachini

1. introduzione al corso
    1. il programma in sintesi
2. aspetti organizzativi
    1. il libro di testo
    2. i momenti di verifica

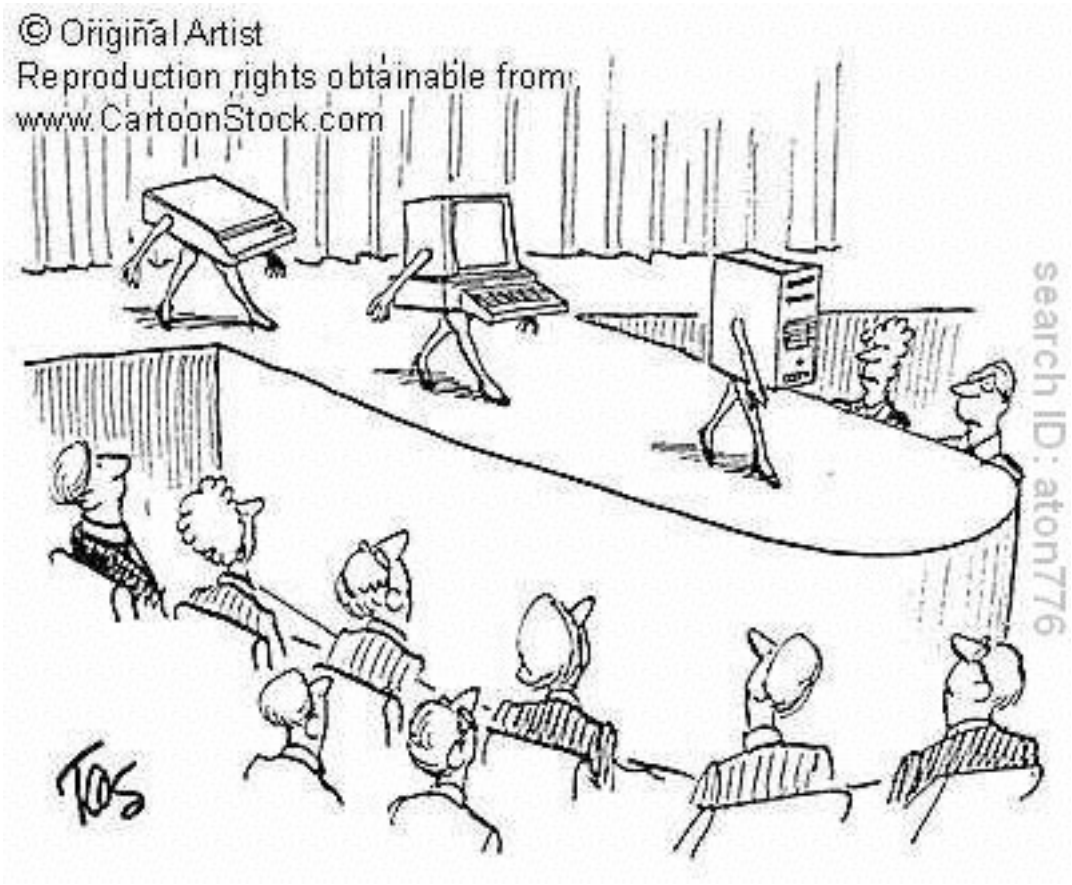


Introduzione al corso: cosa si studierà

1. Linguaggi formali e modelli di calcolo
2. Teoria della calcolabilità
3. Teoria della complessità

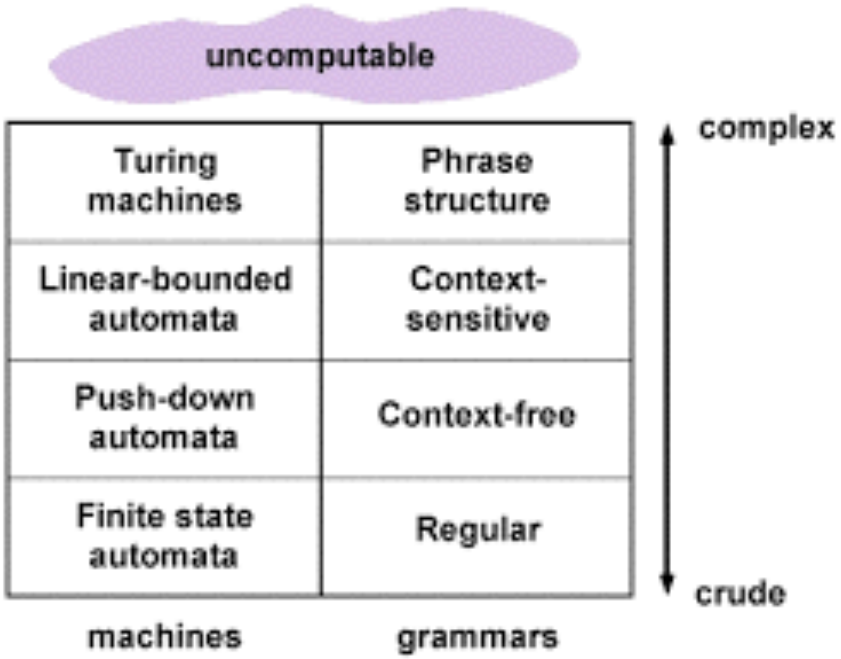
1. Linguaggi formali e modelli di calcolo

Introdurremo alcuni dei modelli matematici di computazione della gerarchia di Chomsky e ne studieremo le principali proprietà e risultati.



La gerarchia di Chomsky

modelli matematici di calcolo: riconoscono linguaggi



Grammatiche: generano linguaggi

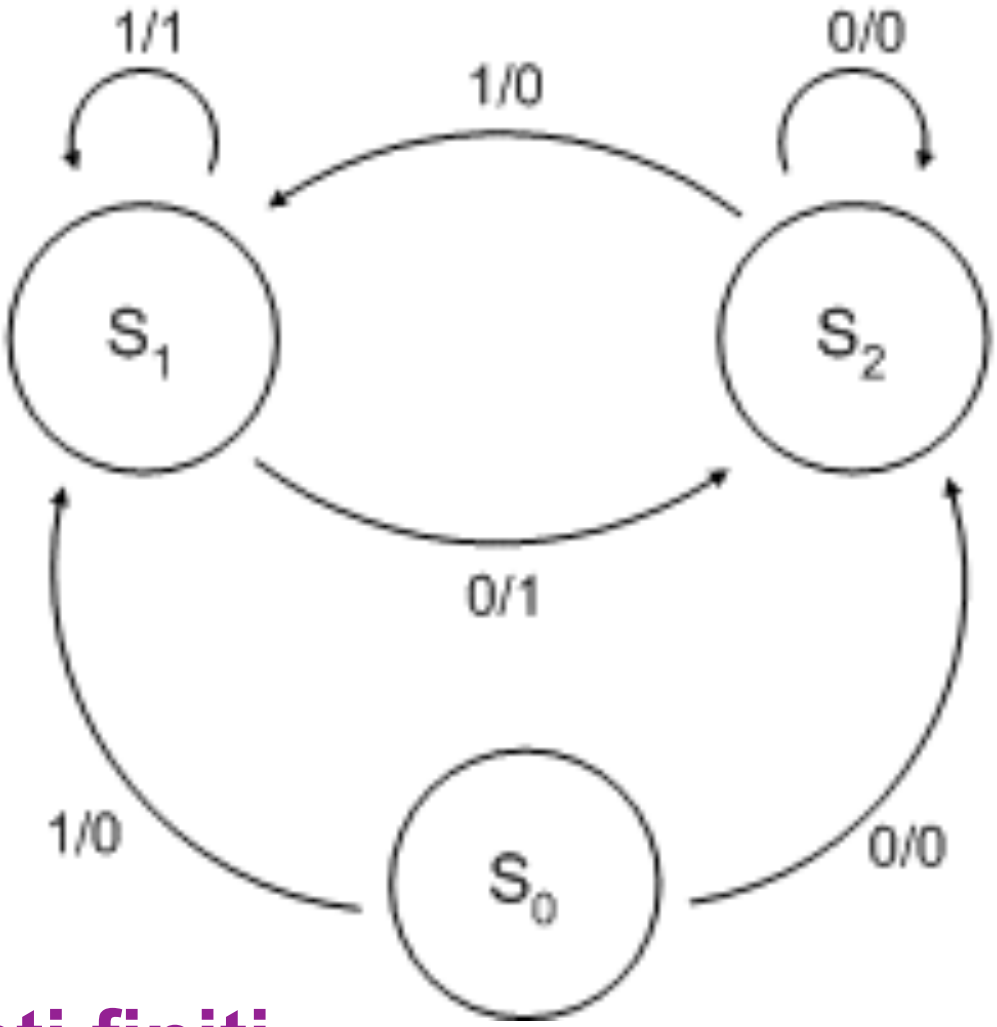
Esempio di grammatica (da Wikipedia)

<indirizzo postale> ::= <destinatario> <indirizzo> <località>
<destinatario> ::= [<titolo>] [<nome>|<iniziale>] <cognome> <a capo>
<indirizzo> ::= <via> <numero civico> <a capo>
<località> ::= [<CAP>] <comune> <provincia>

...

<CAP> ::= <cifra><cifra><cifra><cifra><cifra>
<cifra> ::= 0|1|2|3|4|5|6|7|8|9

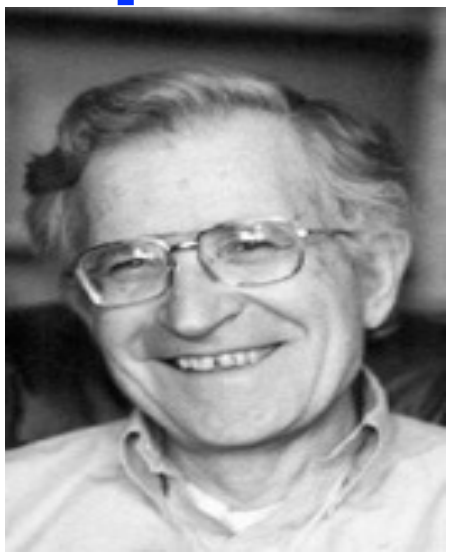
Esempio di macchina di Miley da Wikipedia



A stati finiti

Contemporaneamente o quasi

Noam Chomsky (1928 - professore al MIT) introduce nel 1956 la gerarchia di linguaggi formali che prende il suo nome. Insieme a Miller, nel 1958 dimostra l'equivalenza tra le grammatiche regolari e gli automi a stati finiti.



Stephen Cole Kleene (1909 - 1994) introduce nel 1956 le espressioni regolari e ne dimostra l'equivalenza con gli automi a stati finiti

Equivalenze. perché?

Il risultato di equivalenza tra le grammatiche regolari e gli automi a stati finiti, essendo costruttivo, consente di produrre l'automa dalla grammatica.
Un analizzatore lessicale è essenzialmente un automa a stati finiti.
L'analisi lessicale è la prima fase di un compilatore: si individuano le parole del linguaggio.

l'equivalenza tra le espressioni regolari e gli automi a stati finiti consente di produrre questo automa a partire dalle espressioni regolari

generatore di analizzatore lessicale, p.e. LEX

E anche

A. G. Oettinger introduce gli automi a pila in "Automatic Syntactic Analysis and the pushdown store" in Proc. Symp. Applied Math., 1961 e

M.P. Schutzenberger, in "Context free languages and pushdown automata" Information and Control, 1963, dimostra l'equivalenza tra gli automi a pila e i linguaggi context-free

M.P. Schutzenberger (1920-1996)

L'analisi sintattica è la seconda fase di un compilatore: verifica la correttezza rispetto alla sintassi. Il teorema fornisce le basi per creare un generatore di parser, p.e. YACC

Linguaggi formali

Introdurremo gli automi a stati finiti, gli automi a pila e le macchine di Turing e ne studieremo le proprietà fondamentali.

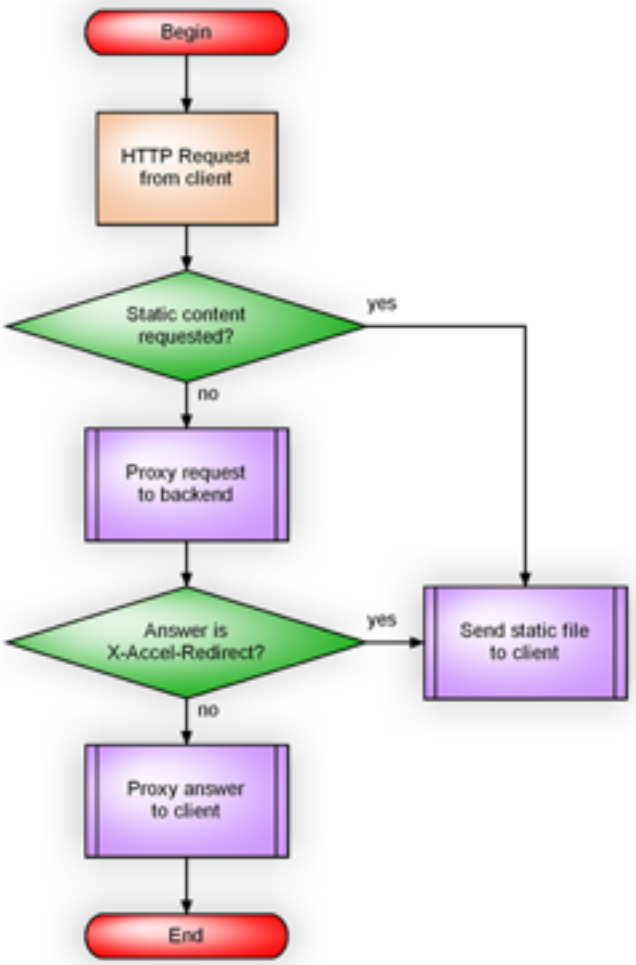
Teoria della calcolabilità

È la teoria che studia quali problemi possono essere risolti con un calcolatore.

Trovare una soluzione algoritmica per un problema può essere arduo, in molti casi è impossibile!

Algoritmo

Quando ci troviamo di fronte ad un algoritmo che risolve un problema siamo in grado di riconoscerlo come tale.



NON esistenza di un algoritmo

Volendo provare che per un particolare problema non esiste un algoritmo è necessario disporre di una definizione precisa di algoritmo.

Church e Turing indipendentemente nel 1936 diedero due risposte diverse, subito provate equivalenti.

Per quale problema hanno dimostrato la non esistenza di un algoritmo?

David Hilbert

1862 - 1943

Un forte impulso verso la definizione precisa di cio' che e' calcolabile viene da Hilbert. Hilbert pensava che tutta la matematica potesse essere assiomatizzata.

Una volta ottenuta questa assiomatizzazione si sarebbe potuta concepire una procedura effettiva che avrebbe deciso in un numero finito di passi se una affermazione matematica era deducibile dagli assiomi o no, quindi se era vera o falsa, tenuto conto che l'assiomatizzazione doveva essere completa e consistente.

Entscheidungsproblem (problema della decisione)

L' Entscheidungsproblem è uno dei problemi posto da David Hilbert in 1928.

Il problema consiste nel chiedere di esibire una procedura, eseguibile del tutto meccanicamente, in grado di stabilire, per ogni formula espressa nel linguaggio formale della logica del primo ordine, se tale formula è o meno un teorema della logica del primo ordine: in altri termini, se tale enunciato è o meno deducibile all'interno del sistema formale.

Entscheidungsproblem (problema della decisione)

Un algoritmo come quello cercato da Hilbert avrebbe potuto risolvere problemi aperti della matematica come

la

Goldbach's conjecture: ogni intero pari maggiore di 2 è esprimibile come la somma di due interi primi.

La congettura vale fino a 4x1018, ma non è dimostrata in generale.

Il problema della decisione di Hilbert (1862-1943)

Risolto negativamente da Church e

da Turing

nel 1936

Alonso Church

4/6/1903-11/8/1995

Identifica il termine "effettivamente calcolabile" con λ-definibile.

Una funzione di interi positivi è detta effettivamente calcolabile se è λ-definibile oppure se è ricorsiva.

La nozione di λ-definibile è dovuta a Church e a Kleene (1933) e quella di ricorsività è dovuta a Gödel, la prova della loro equivalenza è dovuta a Kleene.

Il λ-calcolo ha avuto una grande influenza sulla costruzione dei linguaggi di programmazione, in particolare i funzionali (pure Lisp, Haskell, Miranda and ML).

E' il metalinguaggio della semantica denotazionale

Alan Turing

1912- 1954

Da wikipedia:
Nel 1934 si laureò con il massimo dei voti presso l'Università di Cambridge. Nello stesso anno si trasferì alla Princeton University dove studiò per due anni, sotto la direzione di Church ottenendo infine un Ph.D.

In quegli anni pubblicò l'articolo "*On computable Numbers, with an application to the Entscheidungsproblem*" nel quale descriveva, per la prima volta, quella che sarebbe poi stata chiamata la *macchina di Turing* e nel quale risolve negativamente il problema della decisione di Hilbert. Qui egli dimostra che il problema della fermata per macchine di Turing non è algoritmicamente risolvibile.

Turing e Church

- **Si dimostra che una funzione calcolabile con una macchina di Turing è λ- definibile e viceversa**

- **L'insieme delle funzioni calcolabili è indipendente dal modello di calcolo (purchè generale)**

Ancora sui “Mathematical Problems” di Hilbert

Hilbert presenta 10 (su 23 da lui individuati) problemi al Secondo Congresso Internationale di Matematica in Parigi, nel 1900. Tra questi uno è chiaramente un problema algoritmico :

C’e’ un algoritmo che, ricevuto in input un polinomio a coefficienti interi, stabilisce se la relativa equazione ha soluzione intera?

Hilbert: “…to devise a process according to which it can be determined by a finite number of operations whether the (diophantine) equation is solvable in rational integers.”

Nota. Diophantine: da Diophantus, matematico greco del 3° secolo a.c.

Il 10° problema di Hilbert (1862-1943)

Risolto negativamente da Matijasevic (1947) nel 1970,

grazie ai risultati precedenti di Julia Robinson (1019-1985) e altri.

Calcolabilità

- **Dimostreremo l'indecidibilità del problema della fermata per le macchine di Turing.**
- **Introdurremo il concetto di riduzione tra problemi come strumento per dimostrare decidibilità o indecidibilità dei problemi.**

Teoria della complessità

Ordinare n numeri è facile: O(n lg n).

Mentre determinare se in un grafo diretto con n vertici c'è un cammino hamiltoniano, da un vertice s a uno t, cioè un cammino in cui tutti i vertici sono presenti una e una sola volta, è più difficile: O(n!)

Raggruppiamo in una classe tutti i problemi che possono essere risolti con algoritmi della stessa complessità (di tempo o di spazio) nel caso peggiore.

Esempi di classi di complessità

P = la classe dei problemi che possono essere risolti con un algoritmo con complessità di tempo polinomiale nel caso peggiore.
[cioè in $O(n^k)$, per $k \geq 1$]

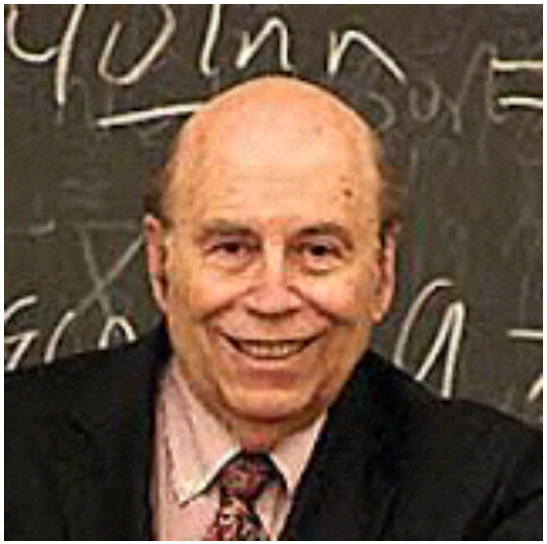
Es. ordinamento è in P

EXP = la classe dei problemi che possono essere risolti con un algoritmo con complessità di tempo esponenziale [cioè in $O(2^{n^k})$, per $k \geq 1$] nel caso peggiore.
Es. il problema del cammino hamiltoniano

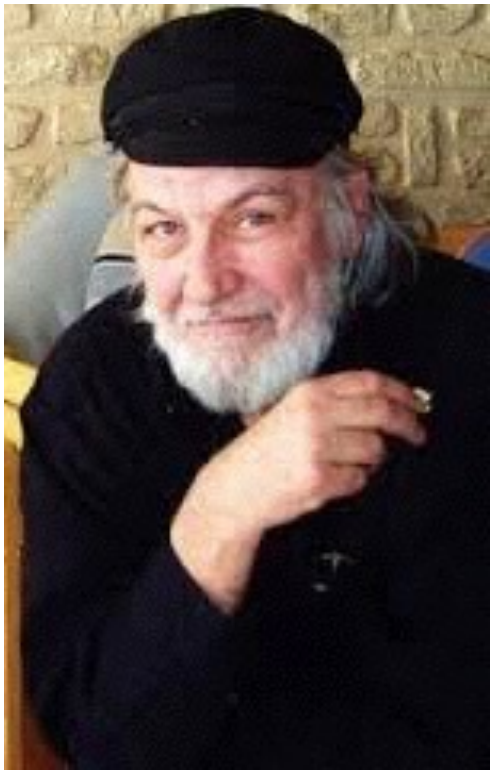
I problemi trattabili

Cobham [64], Edmonds [65] e Rabin [66] hanno contribuito alla identificazione di P come la classe dei problemi trattabili:

Tesi di Cobham o anche Cobham-Rabin



Michael Rabin



Jack Edmonds

I problemi intrattabili, forse

Cook [71] e Levin [73] hanno introdotto il concetto di NP-completo. La classe NP è quella dei problemi verificabili in tempo polinomiale

Stephen Cook

Se si trova un algoritmo polinomiale per un problema NP-completo si dimostra che P = NP

Leonid Levin

Complessità

- **Studieremo le classi di complessità più note**
- **Introdurremo il concetto di riduzione polinomiale tra problemi come strumento per dimostrare l'appartenenza o meno a una classe di complessità.**
- **affronteremo il più antico problema aperto dell'informatica teorica: P vs NP.**

Modelli di calcolo

Durante il corso dovremo confrontare il potere computazionale di diversi modelli di calcolo.

Per poter confrontare tra loro modelli di calcolo diversi bisogna decidere cosa intendiamo per computazione.

Prima considerazione: trattiamo solo problemi decisionali

Un problema decisionale è un problema che ha solo risposta sì o no.

Esempio di problema decisionale: Dato un grafo diretto G=(V,E) e due suoi vertici s e t, esiste un cammino hamiltoniano nel grafo da s a t, cioè un cammino che passa per ogni vertice una sola volta?

Un istanza di un problema è un particolare input per quel problema.

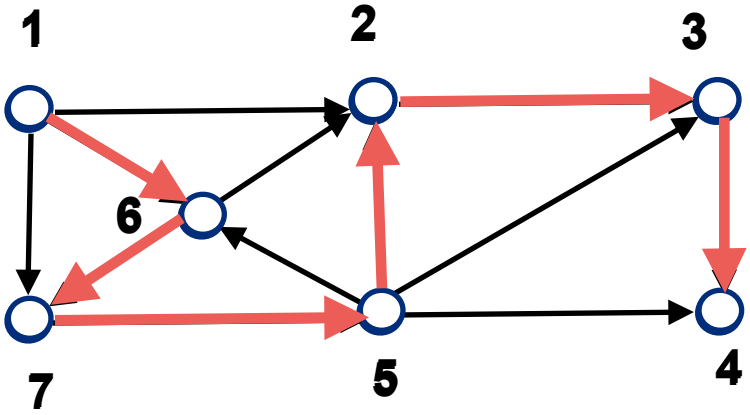
Istanze sì

Esempio di istanza: Un particolare grafo diretto, e due suoi vertici costituiscono un'istanza del problema del cammino hamiltoniano.

Chiamiamo istanze sì quelle che hanno soluzione sì e istanze no quelle che hanno soluzione no.

input G, 1,4

G=

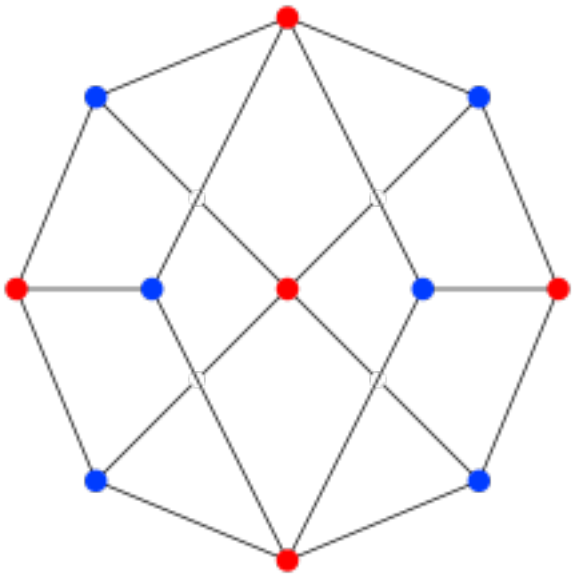


istanza sì

e istanze no

Esempio di istanza: Un particolare grafo diretto, e due suoi vertici costituiscono un'istanza del problema del cammino hamiltoniano.

Chiamiamo istanze sì quelle che hanno soluzione sì e istanze no quelle che hanno soluzione no.



istanza no

Problemi decisionali e ricerca di una soluzione

In generale in realtà siamo interessati a trovare una soluzione piuttosto che a sapere se c'è una soluzione.

Chiamiamo problemi di ricerca, in contrapposizione ai problemi di decisione, quelli per i quali cerchiamo una soluzione.

Per esempio il problema della ricerca di un cammino hamiltoniano consiste nel fornire un cammino hamiltoniano in un grafo diretto G da un suo vertice s a un suo vertice t

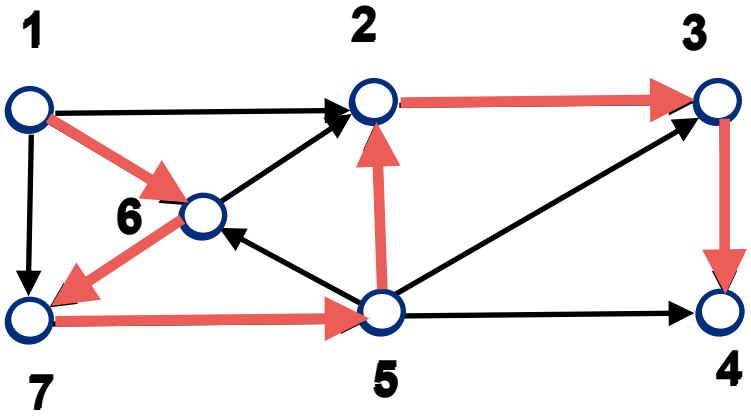
Ma possiamo usare un algoritmo per il problema decisionale per costruire un algoritmo per il problema della ricerca.

Esempio: Per il problema della ricerca di un cammino hamiltoniano rimuoviamo un arco alla volta se il grafo ottenuto dopo la rimozione ha ancora un cammino hamiltoniano. Al termine del processo resta giusto il cammino hamiltoniano.

Esempio Hamilton

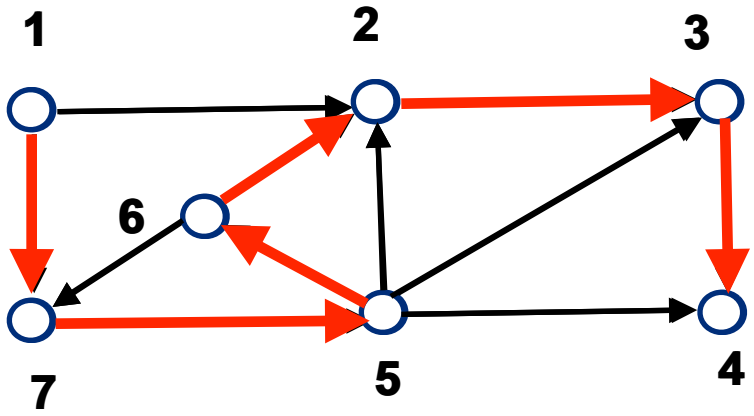
input G, 1,4

G=

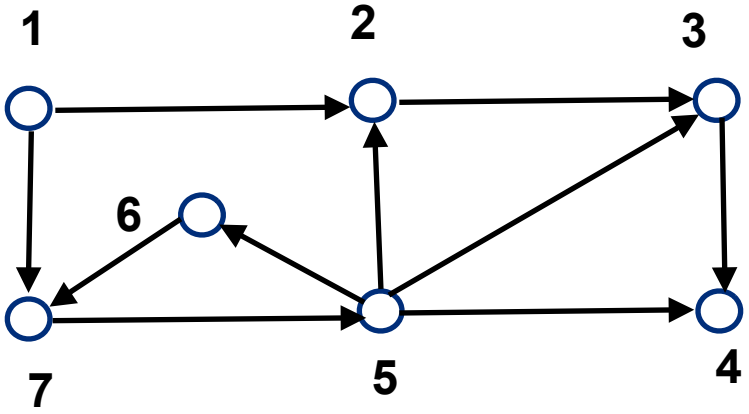


Considera (1,6)

G=



G=



L'eliminazione di (6,2) produrrebbe un grafo che non ha un hampath 1-4

Complessità problemi decisionali e di ricerca

L'algoritmo per determinare un cammino hamiltoniano ha una complessità che dipende polinomialmente da quello per la ricerca.

Un risultato analogo si può ottenere per molti problemi, in particolare per i problemi NP-completi.

Lo studio delle classi di complessità dei problemi è semplificato dal ridursi a considerare problemi decisionali ma i risultati sono spesso applicabili ai problemi di ricerca associati.

Questa osservazione giustifica la concentrazione su problemi di decisione.

Linguaggio: definizione

Una parola è una sequenza di simboli presi da un insieme finito, l'alfabeto.

Un linguaggio è un insieme di parole.

Problemi decisionali e linguaggi

Un problema decisionale può essere facilmente "visto" come un linguaggio

Esempio: Dato un grafo diretto G=(V,E), e due suoi vertici s e t esiste un cammino hamiltoniano da s a t nel grafo?

Il linguaggio associato al problema è quello delle istanze sì del problema.

Esempio: L'alfabeto è quello della codifica delle istanze del problema del cammino hamiltoniano, (per esempio l'alfabeto binario) e il linguaggio che ci interessa è quello delle parole (binarie) che codificano grafi diretti e due vertici del grafo s e t che hanno un cammino hamiltoniano da s a t, cioè le istanze sì.

Un dispositivo che riconosce, tra le parole che codificano le istanze, quelle che codificano le istanze sì è un algoritmo che risolve il problema decisionale del cammino hamiltoniano in un grafo.

Problemi decisionali e linguaggi

Una computazione è identificata con il riconoscimento di una parola e

un algoritmo con il riconoscimento un linguaggio.

Prime definizioni e convenzioni

Σ = {*a,b,c*} è un esempio di alfabeto, che è semplicemente un insieme finito.

x = *abbaacba* è un esempio di parola, che è semplicemente una sequenza di lettere dell'alfabeto

ε (o λ) è la parola vuota, senza lettere

Σ^* = {ε,*a,b,c,aa,ab,ac,ba,bb,bc,…*} è l'insieme di tutte le parole che si possono costruire sull'alfabeto {*a,b,c*}.

Un linguaggio L su Σ è un sottoinsieme di Σ^*

Macchine e problemi

- **parola corrisponde a un' istanza**
- **Problema = insieme delle sue istanze**
- **Dato un problema decisionale l'insieme delle sue istanze sì è il linguaggio associato al problema**
- **Modelli di calcolo sono riconoscitori di linguaggi**
- **Una "macchina" del modello di calcolo che riconosce un linguaggio corrisponde a un algoritmo per il problema**

Aspetti organizzativi

- **Il testo di riferimento è**

  Autore: Michael Sipser

  Titolo: Introduction to the Theory of Computation

  Editore: Course Technology (Thomson)

- **La pagina del corso, su TWIKI, è raggiungibile dalla pagina degli insegnamenti**

- **Durante la settimana di interruzione delle lezioni ci sarà un esonero.**