Automi, Complessità e Calcolabilità
Prof. E. Fachini

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

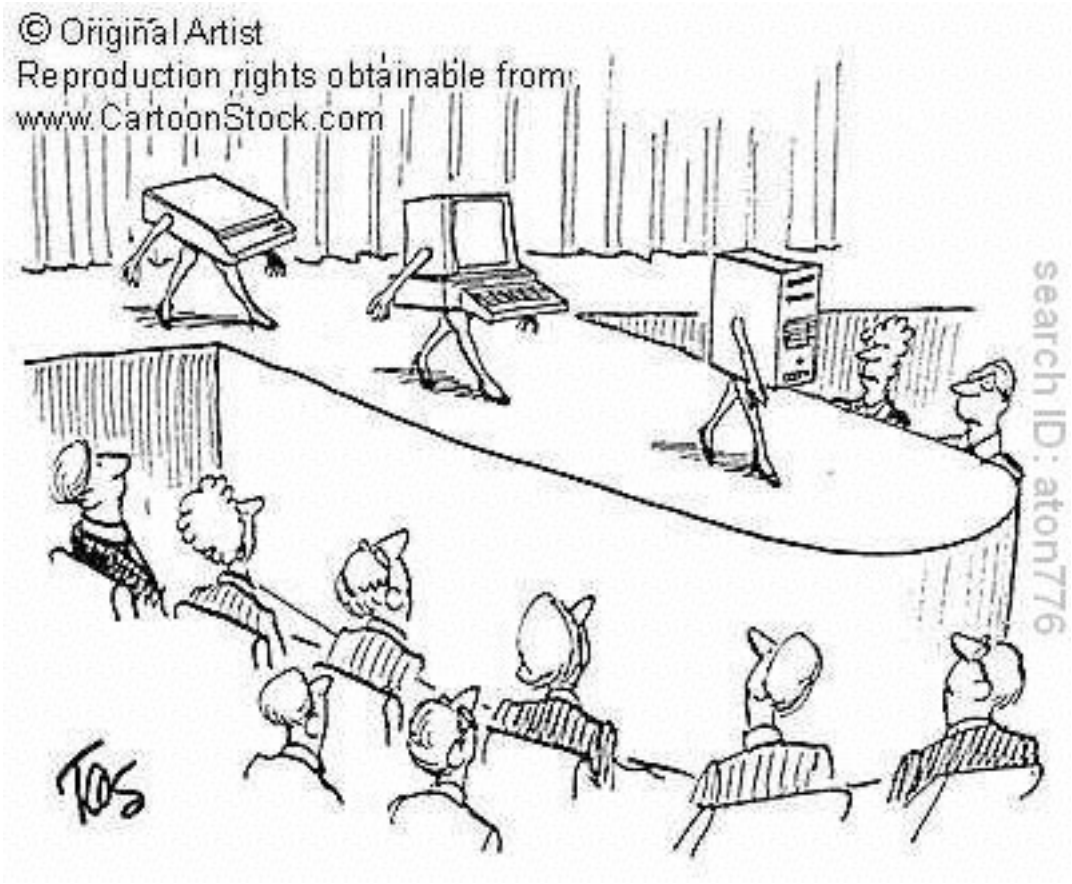
Introduzione al corso: cosa si studierà

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

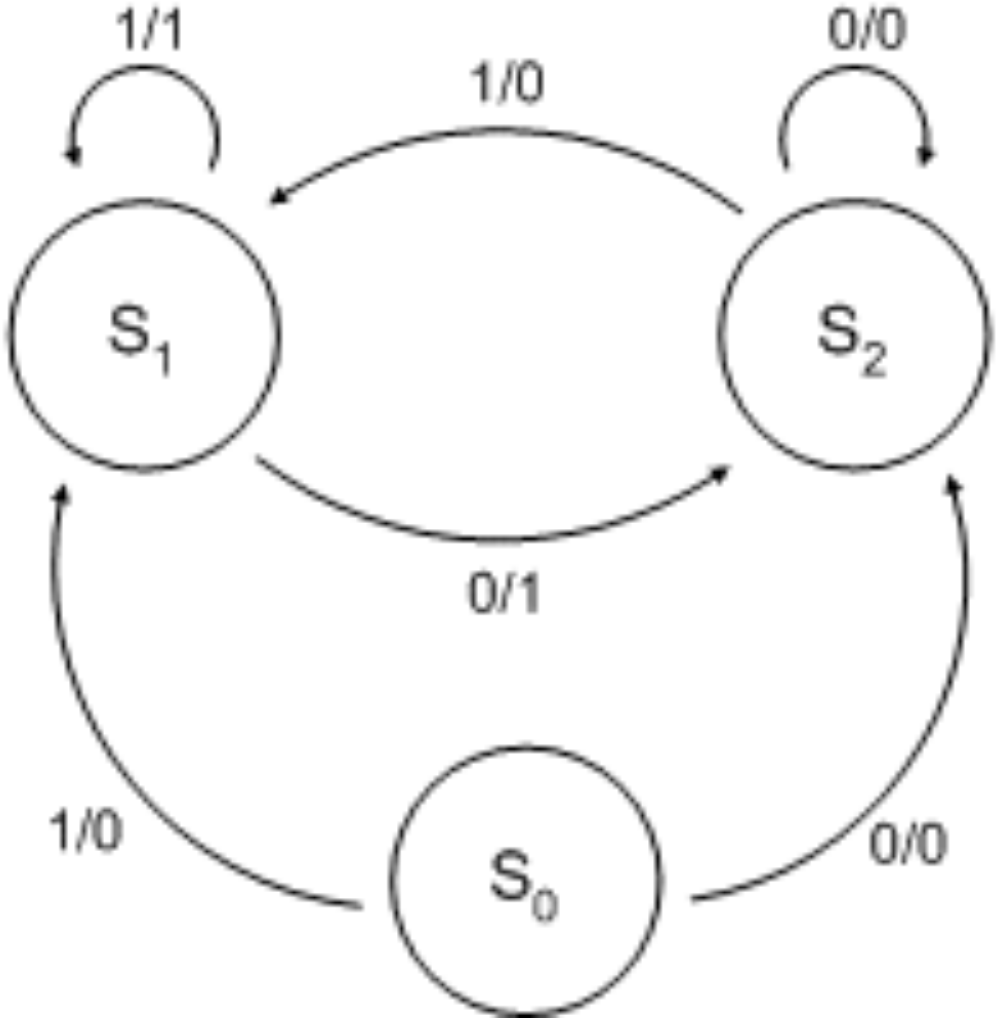
Linguaggi formali e modelli di calcolo

Introdurremo vari modelli matematici di computazione di potere computazionale via via crescente.

Dovremo quindi avere un metodo uniforme di confronto del potere computazionale di diversi modelli.



Esempio di macchina di Miley da Wikipedia



Il numero degli stati è finito

Linguaggi formali e modelli di calcolo

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

Questi modelli di calcolo sono considerati dei riconoscitori di linguaggi, piuttosto che come macchine che dato un input producono un output, per rendere più semplice il confronto in termini di potere computazionale.

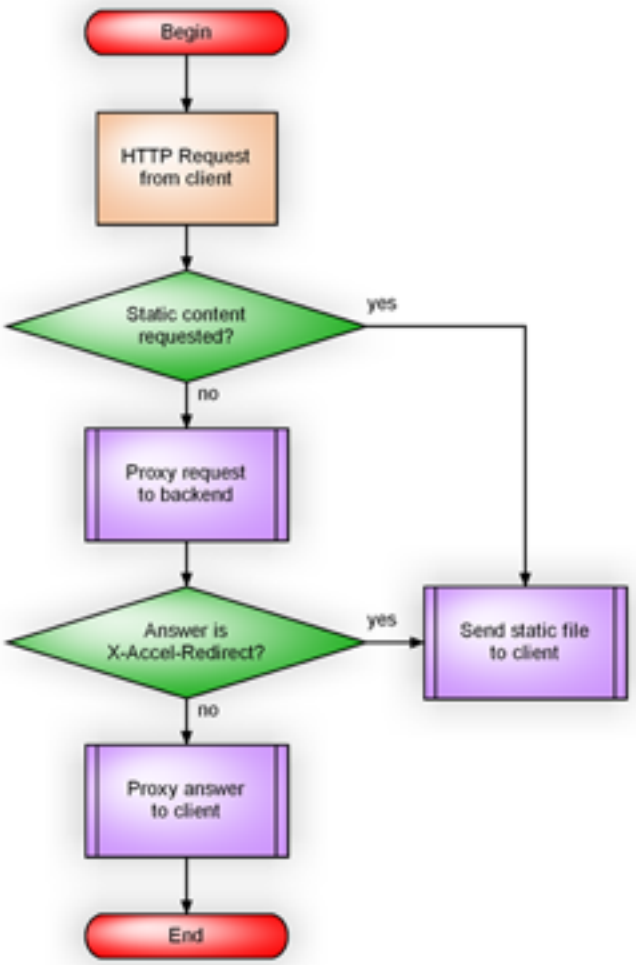
2. Teoria della calcolabilità

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 formale 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)

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).**

Una funzione f è calcolabile se esiste un algoritmo che

- **dà in output il valore della funzione sull'input x, se f è definita su x**
- **e non si ferma altrimenti.**

“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.

10° problema di Hilbert:
c'è un algoritmo che riceve in input un polinomio a coefficienti interi e dà in output "sì" se l'equazione corrispondente ha soluzioni intere e "no" altrimenti?

Risolto negativamente da Matijasevic (1947) nel 1970, grazie ai risultati precedenti di Julia Robinson (1019-1985) e altri.

Teoria della calcolabilità

E' una vasta area di studio che comprende

- **la definizione, lo studio e il confronto di vari modelli di calcolo,**
- **metodi di prova per dimostrare la non esistenza di un algoritmo per risolvere un problema**

Noi introdurremo un unico modello di calcolo generale, la Macchina di Turing, e alcune sue varianti,

- **un unico metodo di prova, la funzione di riduzione,**
- **e dimostreremo l'indecidibilità di alcuni problemi, tra cui quello della fermata per le macchine di Turing e altri la cui indecidibilità si ottiene con il metodo della riduzione.**

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. L'algoritmo basato sulla ricerca esaustiva è in O(n!)

Teoria della complessità

Ma la complessità esponenziale è proibitiva, per dimensioni di istanze anche ragionevolmente piccole, tanto da rendere i problemi di complessità esponenziale e intrattabili, come se l'algoritmo che li risolve non ci fosse!
La teoria cerca di capire le ragioni di questa complessità e come dominarla, se possibile.

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

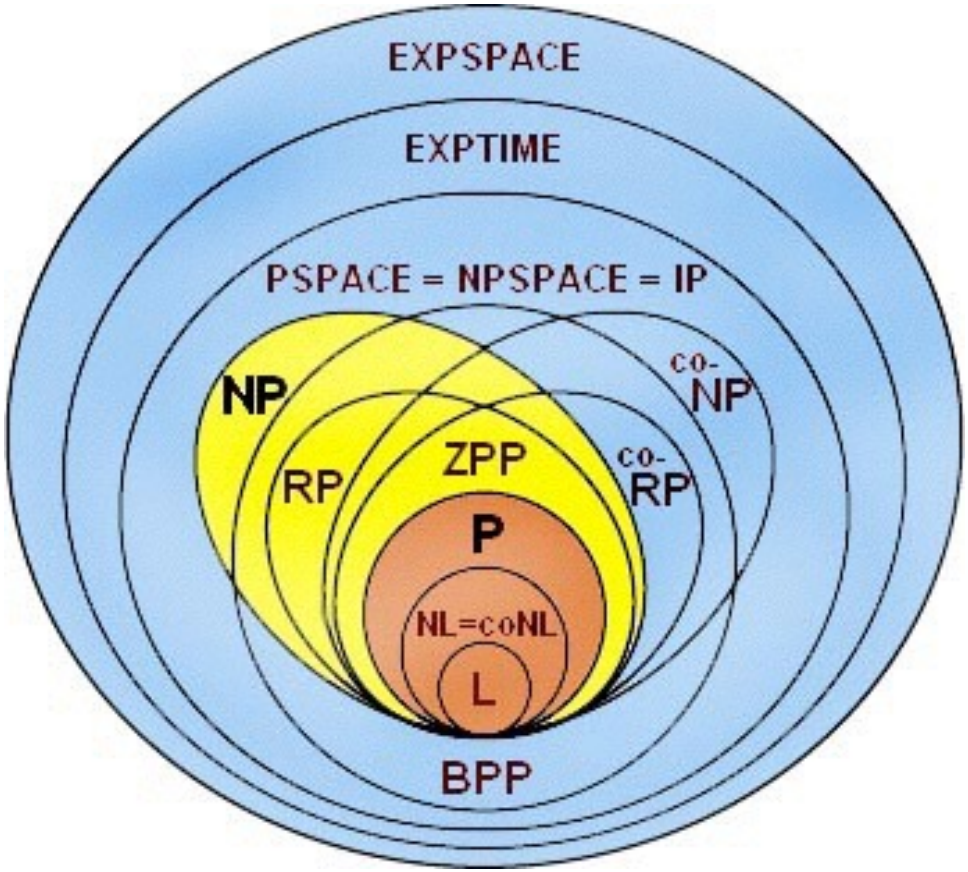
Esempi di classi di complessità

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 $k \geq 0$]

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 $k \geq 0$] nel caso peggiore.
Es. il problema del cammino hamiltoniano, che può essere risolto da un algoritmo il cui tempo di esecuzione nel caso peggiore è in $O(n^2 2^n)$.

Teoria della complessità



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.**
- **esporremo il più importante problema aperto dell'informatica teorica: P vs NP.**

Aspetti organizzativi

- **Il testo di riferimento è**

  Autore: Michael Sipser

  Titolo: Introduction to the Theory of Computation

Titolo italiano: Introduzione alla teoria della computazione

a cura di Clelia de Felice, Luisa Gargano e Paolo D'Arco

Editore: Course Technology (Thomson)

per la traduzione: Apogeo

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

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.

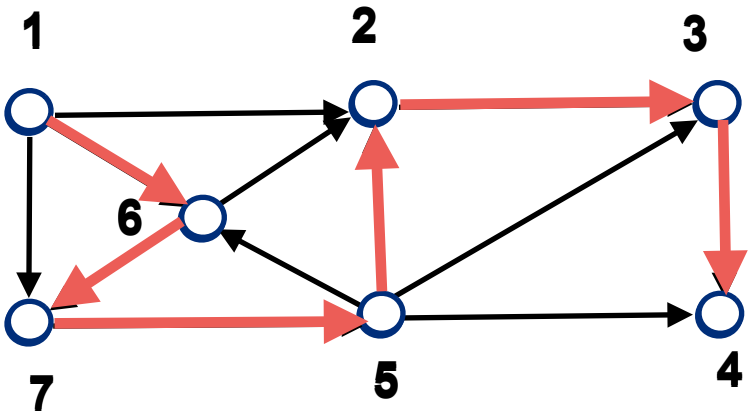
Istanze sì

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=

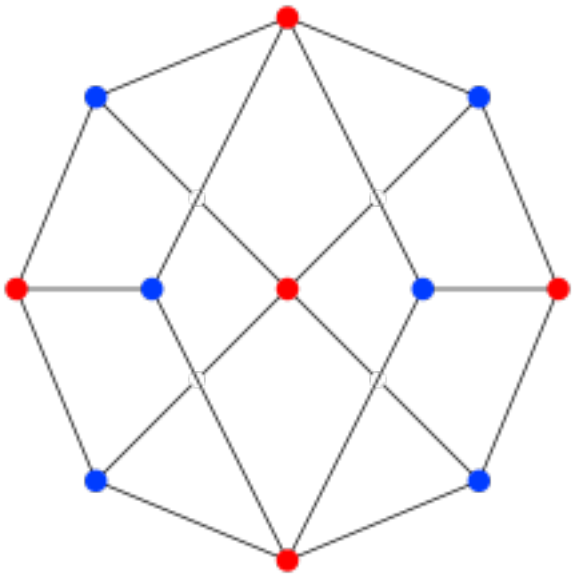


istanza sì

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.

Problemi decisionali e ricerca di una soluzione

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

Esempio: Verifichiamo se un cammino hamiltoniano tra i due vertici dati è presente nel grafo.

Se la risposta è no il processo è terminato, altrimenti

rimuoviamo un arco se il grafo ottenuto dopo la rimozione ha ancora un cammino hamiltoniano.

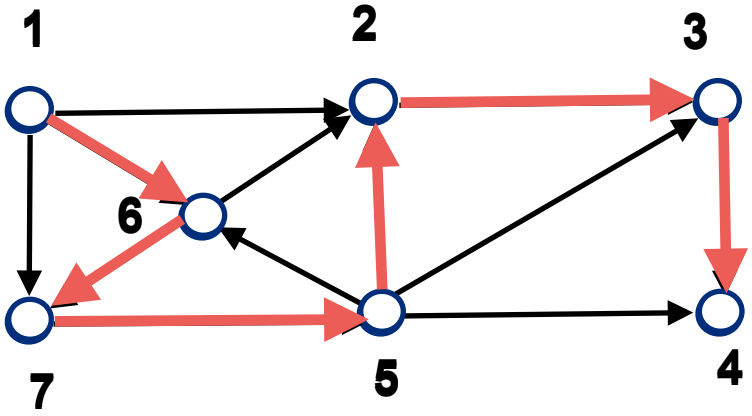
Al termine del processo resta giusto il cammino hamiltoniano.

Ogni verifica è naturalmente eseguita utilizzando l'algoritmo che risolve il problema decisionale.

Esempio Hamilton

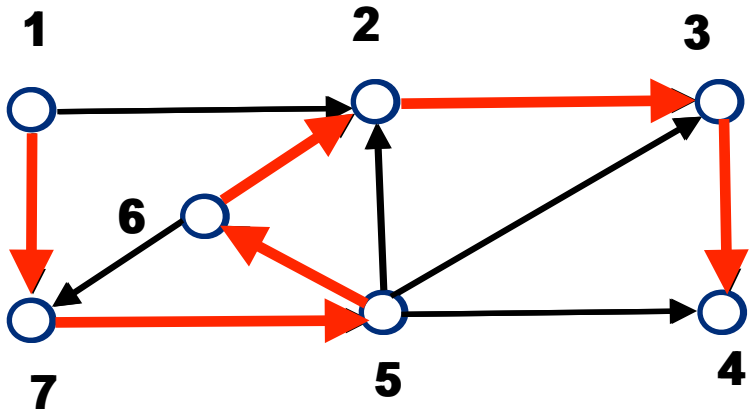
input G, 1,4

G=

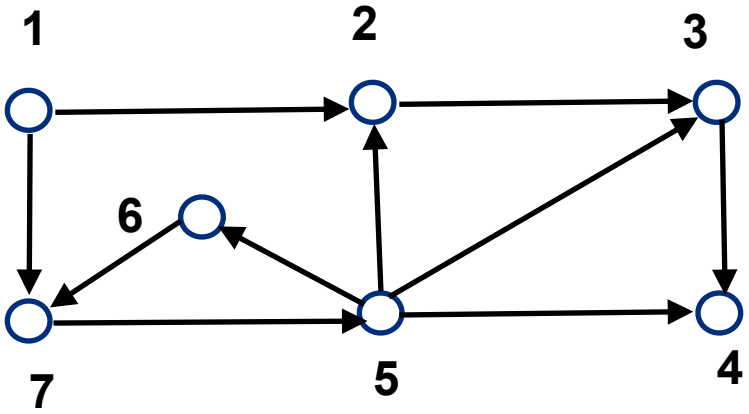


Considera (1,6)

G=



G=



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

SAT

SAT ={ φ | φ è una formula booleana soddisfacibile}.

Esempio di formula booleana :

$$\varphi = (x1 \vee \neg x2) \wedge (x2 \vee x3 \vee \neg x1) \wedge (x4 \vee \neg x3)$$

Problema SAT

input: φ, dove φ una formula booleana

Problema di decisione: esiste un assegnamento di valori di verità che soddisfa φ?

Risposta: Sì o no

Però la soddisfacibilità di una formula booleana è un tipico problema di ricerca (search): cerchiamo un assegnamento di valori di verità che renda vera la formula, tra tutti quelli possibili.

RSAT

Esempio di formula booleana :

$$\varphi = (x1 \vee \neg x2) \wedge (x2 \vee x3 \vee \neg x1) \wedge (x4 \vee \neg x3)$$

Problema RSAT

input: φ, dove φ una formula booleana

Problema di ricerca: trovare un assegnamento di valori di verità, se esiste, che soddisfa φ.

Risposta: un assegnamento che soddisfa φ o "φ non è soddisfacibile"

Decisione vs ricerca per SAT

Supponiamo di disporre di un algoritmo per SAT, M_{SAT} tale che:

M_{SAT}

input: φ, dove φ una formula booleana

risposta sì se φ è soddisfacibile

altrimenti risposta no.

Possiamo usarlo come "subroutine" in un algoritmo che calcola un assegnamento che soddisfa φ.

IDEA:

Si esegue M_{SAT} su φ se la risposta è no, si termina con risposta "φ non è soddisfacibile", altrimenti

si assegna 0 alla prima variabile e si esegue M_{SAT} sulla formula ottenuta φ_0, se φ_0 non è soddisfacibile si pone la prima variabile a 1, e si controlla se φ_1 è soddisfacibile;

in entrambi i casi

si prosegue nello stesso modo per la seconda variabile, e così via fino ad esaurimento delle variabili.

Algoritmo di ricerca per SAT:esempio di esecuzione

Si esegue M_{SAT} su φ se la risposta è no, si termina con risposta "φ non è soddisfacibile", altrimenti

si assegna 0 alla prima variabile e si esegue M_{SAT} sulla formula ottenuta φ_1, se φ_1 non è soddisfacibile si pone la prima variabile a 1,

in entrambi i casi si prosegue nello stesso modo per la seconda variabile, e così via fino ad esaurimento delle variabili.

Esempio:

$\varphi(x1, x2, x3, x4) = (x1 \vee \neg x2) \wedge (x2 \vee x3 \vee \neg x1) \wedge (x4 \vee \neg x3)$

Si esegue M_{SAT} su

$\varphi(x1, x2, x3, x4) = (x1 \vee \neg x2) \wedge (x2 \vee x3 \vee \neg x1) \wedge (x4 \vee \neg x3)$, poichè è soddisfacibile, si costruisce

$\varphi(0, x2, x3, x4) = \varphi_0(x2, x3, x4) = (0 \vee \neg x2) \wedge (x2 \vee x3 \vee 1) \wedge (x4 \vee \neg x3)$

e si esegue M_{SAT} su φ_0

se la risposta è sì, allora si prosegue assegnando 0 a x2,

se la risposta è no, allora si considera

$\varphi(1, x2, x3, x4) = \varphi_1(x2, x3, x4) = (1 \vee \neg x2) \wedge (x2 \vee x3 \vee 0) \wedge (x4 \vee \neg x3)$

perchè visto che φ è soddisfacibile uno dei due valori è quello giusto!

Complessità problemi decisionali e di ricerca

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

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

Questa osservazione giustifica la concentrazione su problemi di decisione.

Decisione vs ottimalità

Spesso i problemi di ricerca si presentano "naturalmente" come problemi di ottimizzazione.

Anche in questo caso gli algoritmi per i problemi di decisione si possono utilizzare per i problemi di ottimizzazione, con una complessità aggiuntiva polinomiale.

Si utilizza la ricerca binaria.

Esempio del MIN-TSP

Dato un grafo G, si vuole determinare un ciclo hamiltoniano di costo minimo nel grafo.

Idea: Si basa sulla ricerca binaria.
Un limite superiore al costo di un ciclo hamiltoniano, c_{max}, è n volte il costo dell'arco di costo massimo e quello inferiore, c_{min}, è n volte il costo dell'arco di costo minimo.

Si verifica se c'è un ciclo hamiltoniano di costo m pari a circa la metà tra minimo e massimo.
Se si trova un ciclo hamiltoniano di costo m, per capire se è minimale si cerca un ciclo hamiltoniano di costo minore nell'intervallo $[c_{min}, m-1]$, altrimenti, cioè se un ciclo hamiltoniano di dimensione m non c'è nel grafo, dobbiamo cercarne uno nell'intervallo $[m+1, c_{max}]$. Come nel caso del clique reiterando il ragionamento si ottiene l'algoritmo desiderato

Esempio del MIN-TSP

Dato un grafo G, si vuole determinare un ciclo hamiltoniano di costo minimo nel grafo.

O_{TSP}
Input <G=(V,E)> G ha n vertici
low = c_{min}*n
high = c_{max}*n;
while low ≤ high do
    m = (low+high)/2
    if M_{TSP} <G=(V,E),m> = 1 then m'= m; high = m - 1 else low = m + 1
Output C_{TSP} <G=(V,E), m' >

l'output di C_{TSP}(G,k) è un ciclo hamiltoniano di costo k

Complessità: Siano n e m il numero dei vertici rispettivamente degli archi di un grafo, $t_M(n,m)$ la complessità della procedura M_{TSP} e $t_C(n,m)$ la complessità di C_{TSP}
Allora O_{TSP} ha complessità $t_O(n,m) = O(m + lg\ n^* t_M(n,m) + t_C(n,m))$

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 per i quali esiste 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 elementi 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**