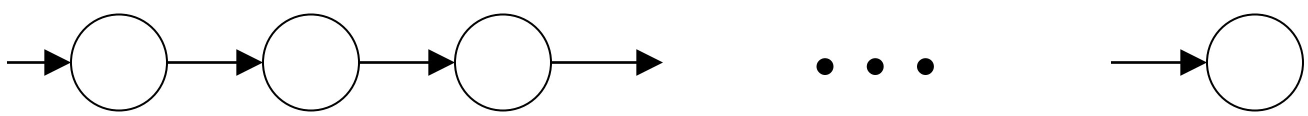
Complessità temporale

Considera una Turing Machine deterministica M che decide un linguaggio L

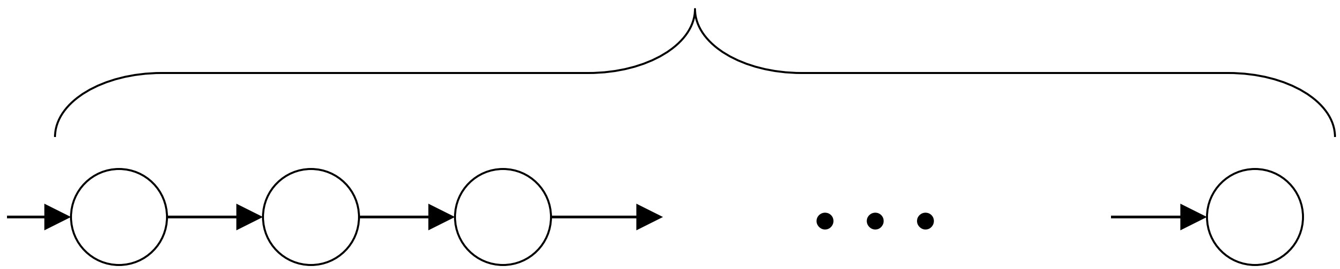
Per ogni stringa w la computazione di M termina usando una quantità finita di transizioni



Iniziale
stato

Accetta
o rifiuta w

Tempo di Decisione = #transizioni



stato
Iniziale

accettare
or rigettare w

Consideriamo ora, tutte le stringhe di Lunghezza n

$T_M(n)$ = massimo tempo richiesto per decidere (calcolare) Una qualsiasi stringa di lunghezza n

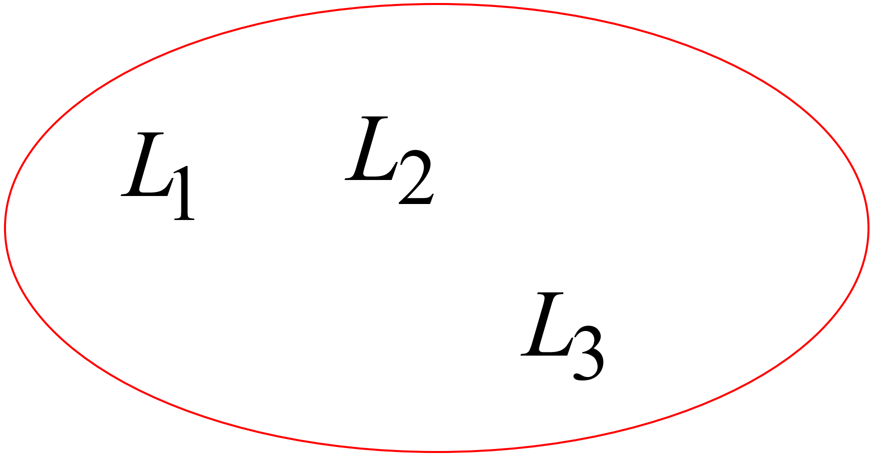
Let f and g be functions $f, g : \mathcal{N} \longrightarrow \mathcal{R}^+$. Say that $\boldsymbol{f(n) = O(g(n))}$ if positive integers c and n_0 exist such that for every integer $n \geq n_0$

$$f(n) \leq c\, g(n).$$

When $f(n) = O(g(n))$ we say that $g(n)$ is an ***upper bound*** for $f(n)$, or more precisely, that $g(n)$ is an ***asymptotic upper bound*** for $f(n)$, to emphasize that we are suppressing constant factors.

Classe Complessità temporale : $TIME(T(n))$

Tutti i linguaggi decidibili da una Turing Machine deterministica in tempo $O(T(n))$



Esempio: $L_1 = \{a^n b : n \geq 0\}$

Può essere deciso in tempo $O(n)$

$TIME(n)$

$L_1 = \{a^n b : n \geq 0\}$

Altri esempi nella stessa classe

$$TIME(n)$$

$$L_1 = \{a^n b : \ n \geq 0\}$$

$$\{ab^n aba : \ n,k \geq 0\}$$

$$\{b^n : \ n \text{ è pari}\}$$

$$\{b^n : \ n = 3k\}$$

Esempi nella classe

$$TIME(n^2)$$

$$\{a^n b^n : n \geq 0\}$$

$$\{ww^R : w \in \{a,b\}\}$$

$$\{ww : w \in \{a,b\}\}$$

M_1 = "On input string w:

1. Scan across the tape and *reject* if a 0 is found to the right of a 1.
2. Repeat if both 0s and 1s remain on the tape:
3. Scan across the tape, crossing off a single 0 and a single 1.
4. If 0s still remain after all the 1s have been crossed off, or if 1s still remain after all the 0s have been crossed off, *reject*. Otherwise, if neither 0s nor 1s remain on the tape, *accept*."

In stages 2 and 3, the machine repeatedly scans the tape and crosses off a 0 and 1 on each scan. Each scan uses $O(n)$ steps. Because each scan crosses off two symbols, at most $n/2$ scans can occur. So the total time taken by stages 2 and 3 is $(n/2)O(n) = O(n^2)$ steps.

Esempi nella classe:

$$TIME(n^3)$$

Anche su sipster

CYK algorithm

$$L_2 = \{\langle G, w \rangle : w \text{ è generata da una grammatica context-free } G\}$$

Moltiplicazione tra matrici

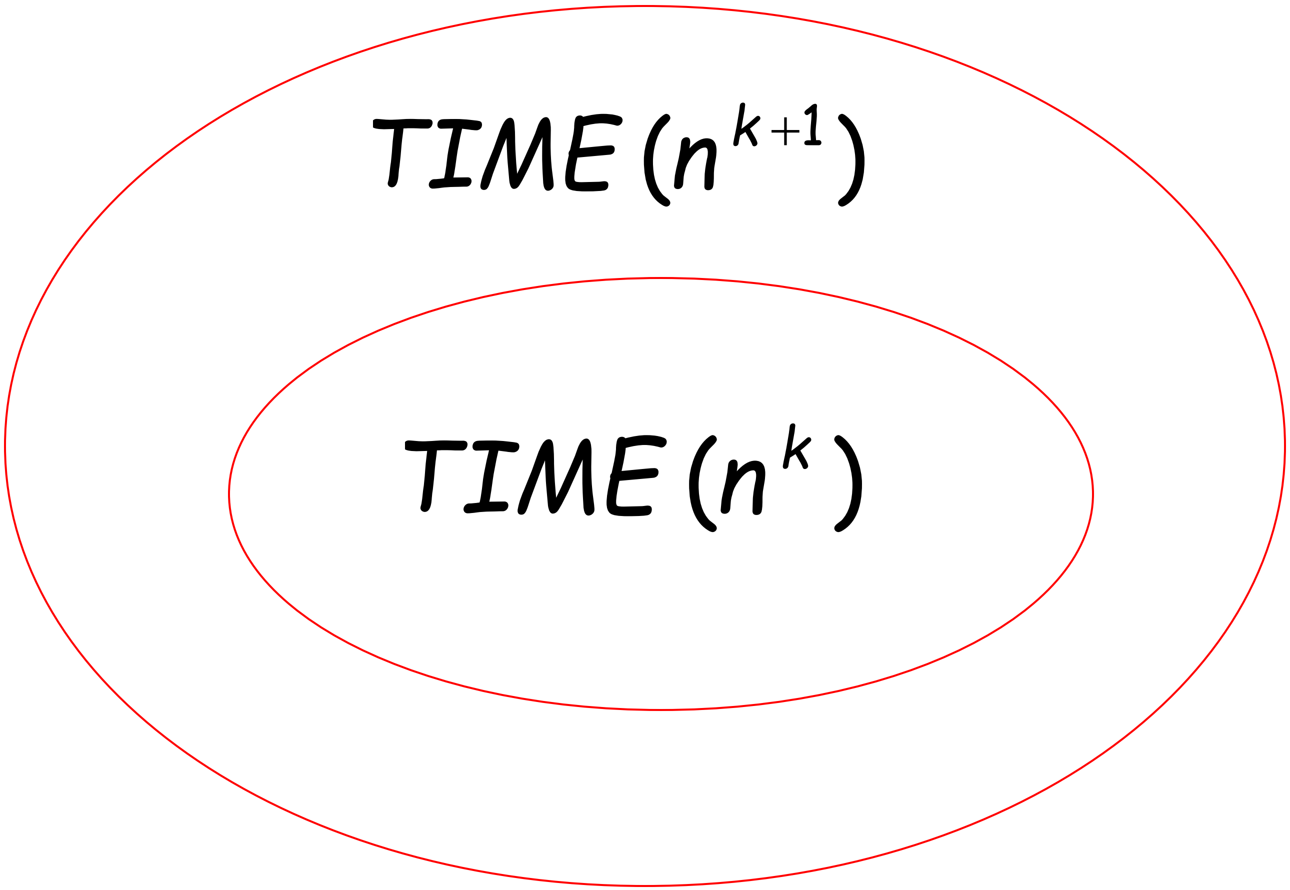
$$L_3 = \{\langle M_1, M_2, M_3 \rangle : n \times n \text{ matrices and } M_1 \times M_2 = M_3\}$$

algoritmi tempo Polinomiale : $TIME(n^k)$

costante $k > 0$

Rappresentano gli algoritmi trattabili:

Per piccoli k possiamo decidere il risultato velocemente chiaramente: $TIME(n^{k+1}) \supset TIME(n^k)$



architetture

Multitape = singolo tape in tempo quadratico

Anche sul sipster

La Classe di Complessità temporale P

$$P = \bigcup_{k>0} TIME(n^k)$$

Representa:

- Algoritmi che usano tempo polinomiale
- Problemi "trattabili"

Class P

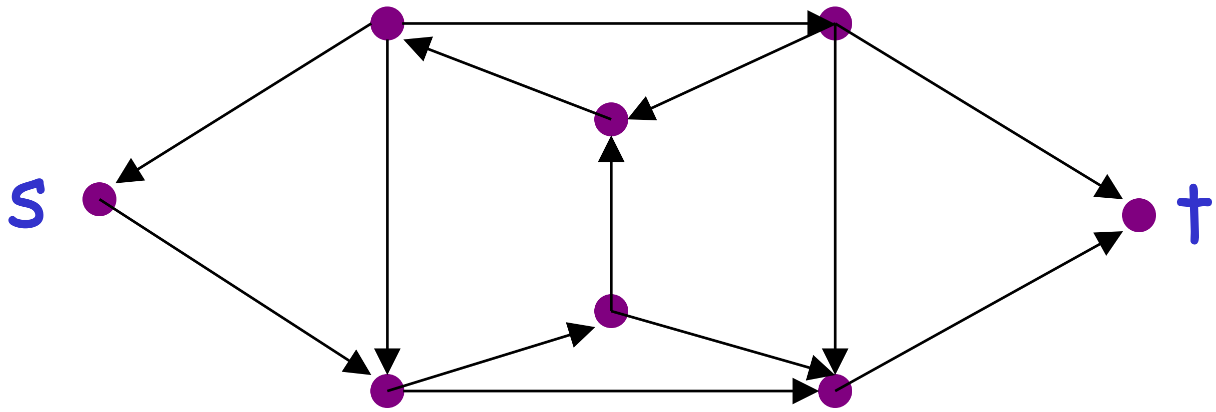
$\{a^n b\}$

$\{a^n b^n\}$

$\{ww\}$

CYK-algorithm

Dato un grafo e due nodi: esiste un cammino da un nodo all'altro?



Dato un grafo e due nodi: esiste un cammino da un nodo all'altro?

PROOF A polynomial time algorithm M for $PATH$ operates as follows.

M = "On input $\langle G, s, t\rangle$ where G is a directed graph with nodes s and t:

1. Place a mark on node s.
2. Repeat the following until no additional nodes are marked:
3. Scan all the edges of G. If an edge (a, b) is found going from a marked node a to an unmarked node b, mark node b.
4. If t is marked, *accept*. Otherwise, *reject*."

Step 1 : 1
Step 4: 1
Step 2,3 : m (numero dei nodi)

Qualche problema non in P

algoritmi calcolabili
in tempo esponenziale

$$TIME(2^{poly(k)})$$

$$TIME(2^{n^k})$$

Rappresentano algoritmi intrattabili

Per alcuni input ci possono volere
secoli per trovare la soluzione

Ricordiamo:

Let $t(n)$ be a function, where $t(n) \geq n$. Then every $t(n)$ time nondeterministic single-tape Turing machine has an equivalent $2^{O(t(n))}$ time deterministic single-tape Turing machine.

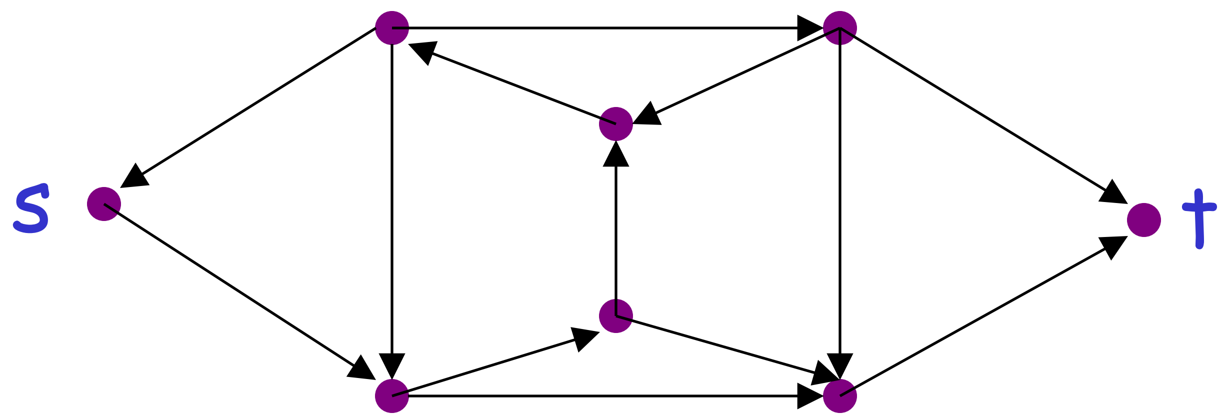
NonDeterminismo = Determinismo in tempo esponenziale (dim anche sipster)

uno

Problema: Cammino Hamiltonian

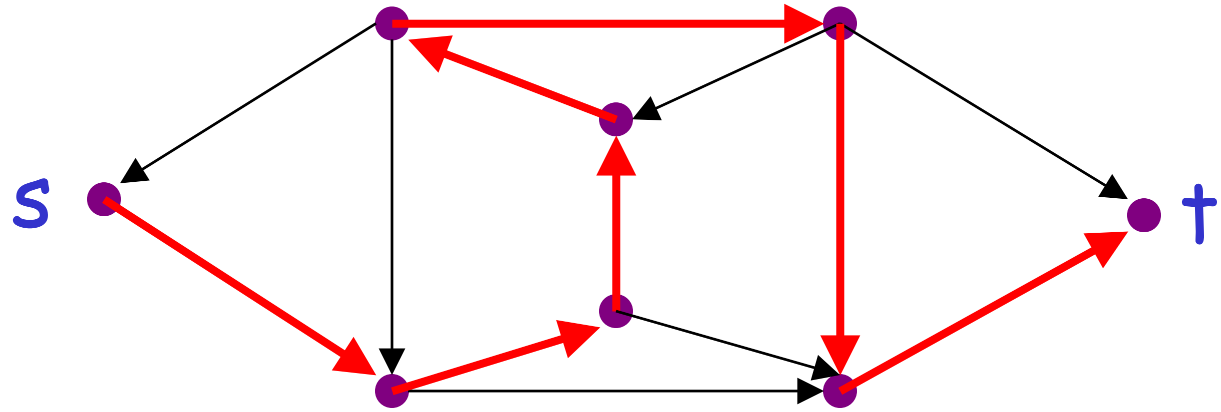
un cammino in un grafo
(orientato o non orientato)
è detto **hamiltoniano** se
esso tocca tutti i vertici
del grafo
una e una sola volta.

Problema del Cammino Hamiltonian



domanda: vi è un Cammino Hamiltoniano da s a t?

tocca tutti i vertici del grafo una e una sola volta.



si

tocca tutti i vertici del grafo una e una sola volta.

Non esistono algoritmi efficienti per la risoluzione del «problema del cammino hamiltoniano», l'unico metodo di risoluzione è rappresentato dall'enumerazione totale, ovvero nell'enumerazione di tutti i possibili cammini sul grafo fino a che non si trovi il cammino voluto.

ab, ba   Permutazione di n elementi
Arriva c
Possibili posizioni: Avanti, Dietro, mezzo(posizione 2);tre nuovi elementi per ogni elemento di partenza
cab, acb,abc, cba, bca, bac (3!)
Arriva d prendiamo per esempio abc
Avanti, Dietro, posizione 2, posizione 3 (Aa2b3cD); sono 4 nuovi elementi per ogni elemento di partenza
Per ogni tripla
6*4= 24=4!

$$TIME(2^{poly(k)})$$

Una soluzione :

Lavorare su tutti i cammini

L = {<G,s,t>: vi è un Cammino Hamiltonian in G da s a t}

n!=nx(n-1)x.... nxnxn...

$$L \in TIME(n!) \approx TIME(2^{n^k})$$ k=2

Permutazione di n elementi

tempo Esponenzale

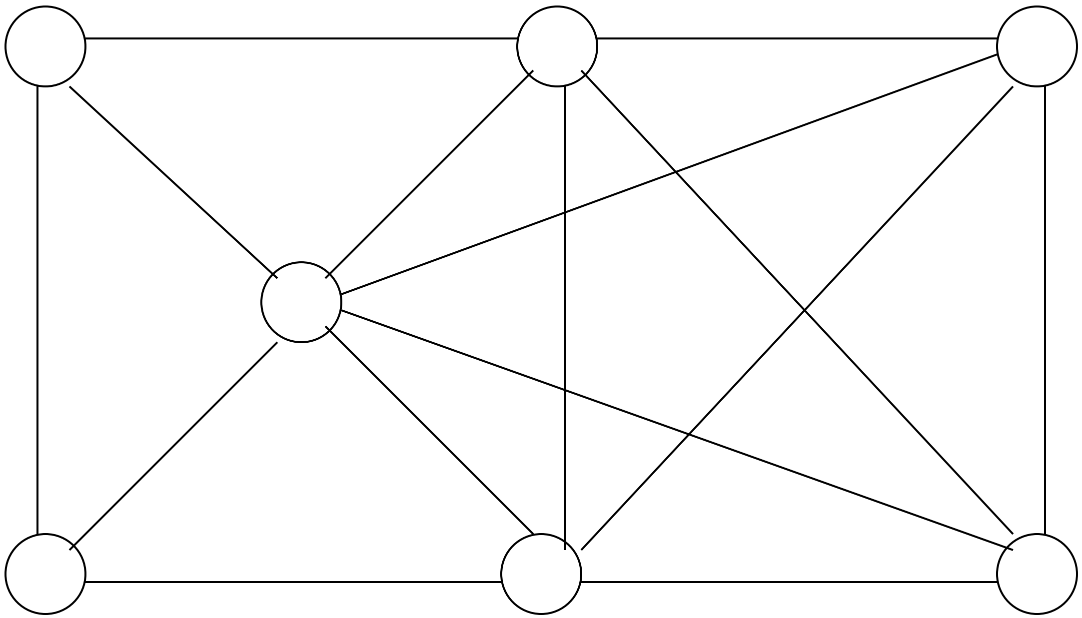
problema Intrattabile

due

problema della cricca
dato un grafo e dato un k

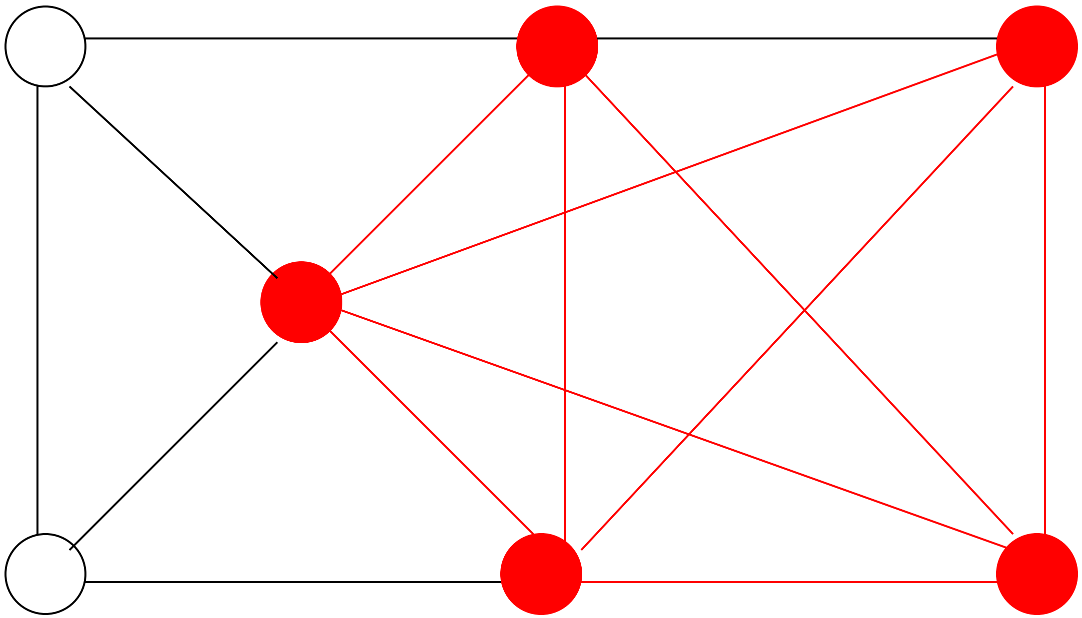
trovare un insieme di k nodi dove ciascun elemento è connesso con tutti gli altri.

Esempio: Il problema della cricca



Esiste una cricca di grado 5?

Il problema della cricca



Esiste una cricca di grado 5.

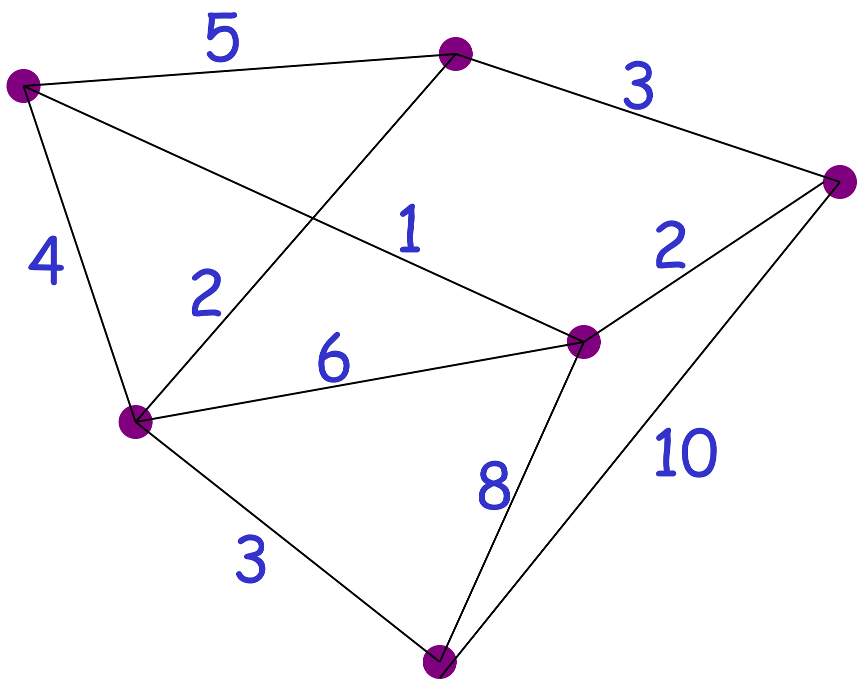
Tutte le permutazioni di 5 elementi e verificare che ogni elemento è connesso con tutti gli altri.

5! X 5!

Per precisione 5!x(5-1)x(4-1)…x1

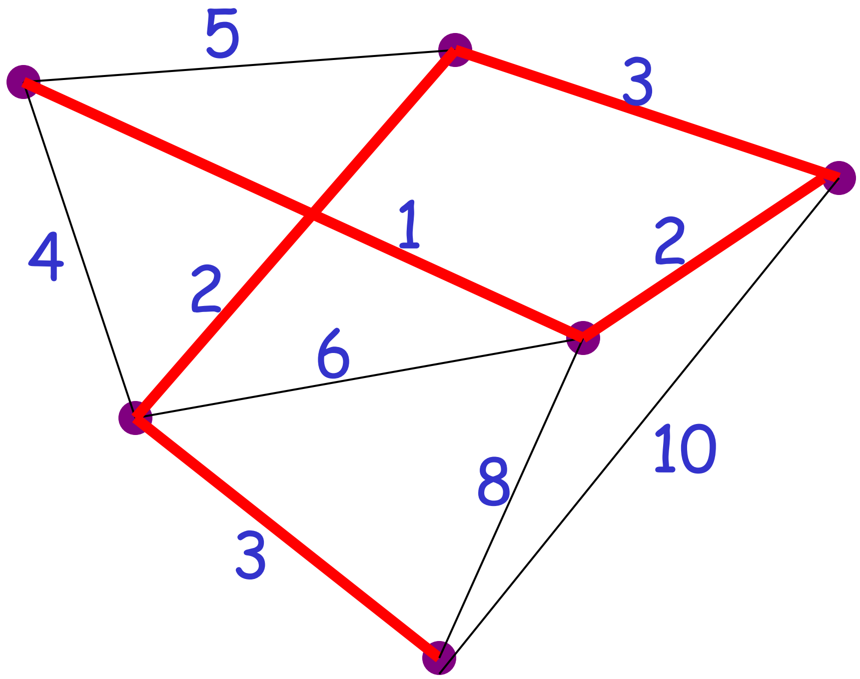
tre

Esempio: commesso viaggiatore



domanda: quale è la via più veloce

Per connettere tutte le città?



quale è la via più veloce
Per connettere tutte le città?

Una soluzione : ricerca tutti i cammini, Hamiltoniani, ovvero tutti i cammini che toccano tutti i vertici una sola volta.

ricorda $L \in TIME(n!) \approx TIME(2^{n^k})$

quindi

L = {shortest hamiltonian paths}

quattro: il Problema della soddisfacibilità

espressioni Booleani in
Conjunctive Normal Form:

$$t_1 \wedge t_2 \wedge t_3 \wedge \cdots \wedge t_k$$ clausole

$$t_i = x_1 \vee \bar{x}_2 \vee x_3 \vee \cdots \vee \bar{x}_p$$

Variabili

domanda: è l'espressione soddisfacibile?

Esempio: $$(\bar{x}_1 \vee x_2) \wedge (x_1 \vee x_3)$$

soddisfacibile: $$x_1 = 0,\ \ x_2 = 1,\ \ x_3 = 1$$

$$(\bar{x}_1 \vee x_2) \wedge (x_1 \vee x_3) = 1$$

Esempio: $(x_1 \vee x_2) \wedge \bar{x}_1 \wedge \bar{x}_2$

Non soddisfacibile

$$(x_1 \vee x_2) \wedge \bar{x}_1$$

soddisfacibile

$$L = \{w : \ w \ \text{soddisfacibile}\}$$

$$L \in TIME(2^{n^k})$$

Algoritmo:

ricerca , in modo esaustivo,
su tutti i possibili valori delle variabili
Tavole di verità, n variabili , 2 ^ n

Non-determinismo: prima definizione

La classe dei linguaggi: $NTIME(T(n))$

Turing Machine Non-Deterministica:
i rami di computazione sono limitati
da un T(n)

Linguaggi decidibili da una mdTuring
non deterministica in tempo $O(T(n))$

Non-determinismo: seconda definizione

A ***verifier*** for a language A is an algorithm V, where

$$A = \{w \mid V \text{ accepts } \langle w, c\rangle \text{ for some string } c\}.$$

We measure the time of a verifier only in terms of the length of w, so a ***polynomial time verifier*** runs in polynomial time in the length of w. A language A is ***polynomially verifiable*** if it has a polynomial time verifier.

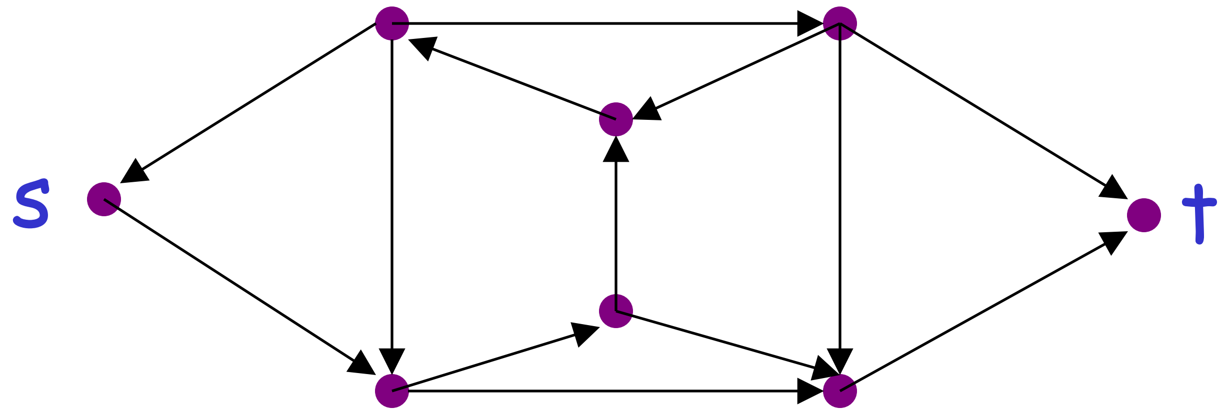
Decide per ogni stringa (w) di lunghezza n con l'aiuto di una stringa c in tempo O(T(n))

uno

Problema: Cammino Hamiltonian

un cammino in un grafo
(orientato o non orientato)
è detto **hamiltoniano** se
esso tocca tutti i vertici
del grafo
una e una sola volta.

Problema del Cammino Hamiltonian



domanda: vi è un Cammino Hamiltoniano
da s a t?

tocca tutti i vertici del grafo
una e una sola volta.

Problema del Cammino Hamiltoniano

Esempio

Stringa giusta che ci dà il cammino.

due

problema della cricca
dato un grafo e dato un k

trovare un insieme di k nodi dove ciascun elemento è connesso con tutti gli altri.

Esempio

.

problema della cricca, dato un k trovare un insieme di k elementi dove ciascun elemento è connesso con tutti gli altri.

Stringa giusta che ci dà la cricca.

il Problema della soddisfacibilità

Valori di verità giusti

quattro: il Problema della soddisfacibilità

espressioni Booleani in
Conjunctive Normal Form:

$$t_1 \wedge t_2 \wedge t_3 \wedge \cdots \wedge t_k$$ clausole

$$t_i = x_1 \vee \bar{x}_2 \vee x_3 \vee \cdots \vee \bar{x}_p$$

Variabili

domanda: è l'espressione soddisfacibile?

Esempio

•

il Problema della soddisfacibilità

Valori di verità giusti

Due definizioni sono uguali?

Verificatore $\Longrightarrow$ NdT

Assumiamo che V è limitata da n^k.
Prendiamo tutte le stringhe di lunghezza n^k :
NdT: Su input w di lunghezza n:

1. Non deterministicamente seleziona stringa c di lng al massimo n^k;
2. Calcola V su (w,c);
3. Se V accetta allora *accetta*
   altrimenti *rifiuta*

NdT $\Rightarrow$ Verificatore

Trovare la stringa

Ricorda che se NdT accetta s allora esiste un cammino, c, che ci porta all'accettazione.

Sia Ndt N costruiamo un verificatore V

V: input (w,c)

1. Simula NdT sull'input w scegliendo il cammino indicato da c.
2. Se V accetta allora accetta,
   altrimenti rigetta

Esempio: $L = \{ww\}$

algoritmo Non-Deterministico
per accettare una stringa ww:

- Usiamo una two-tape Turing machine

- Congetturiamo la metà della stringa e la copiamo w sul secondo nastro

- Compariamo i due nastri

$$L = \{ww\}$$

tempo necessario

Usiamo una two-tape Turing machine

Congetturiamo la metà della stringa e la copiamo w sul secondo nastro $O(|w|)$

Compariamo i due nastri $O(|w|)$

tempo totale: $O(|w|)$

$$NTIME(n)$$

$$L = \{ww\}$$

$$L = \{w : \text{expression } w \text{ is satisfiable}\}$$

$$L \in NP$$

Il problema della soddisfacibilità è un NP - Problem

$$L = \{w : \text{expression } w \text{ is satisfiable}\}$$

Tempo necessario per n variabili:

- Congetturiamo un assegnazione di valore alle variabili $O(n)$

- Verifichiamo che questo assegnamento sia soddisfacibie $O(n)$

Total tempo: $O(n)$

In modo simile possiamo definire

$$NTIME(T(n))$$

Per ogni funzione temporale : $T(n)$

Esempi: $NTIME(n^2), NTIME(n^3), \ldots$

La classe NP
Non-Deterministic Polynomial time

$$NP = \cup NTIME(n^k)$$

Per ogni k

$$P = \cup TIME(n^k)$$

osservazione:

$$P \subseteq NP$$

Deterministic Polynomial

Non-Deterministic Polynomial

Problema aperto: $P = NP\,?$

Non conosciamo la risposta

qui

Problema aperto: $P = NP\,?$

Esempio: il problema della sodisfacibilità
ha un algoritmo deterministico
che lo risolva in tempo polinomiale?

Non conosciamo la risposta

Fine cap.

NP-Completezza

Un problema è NP-complete se:

- E' in NP

- Ogni NP problema si può ridurre
- al problema di partenza

(in tempo polinomiale)

Osservazione:

Se possiamo risolvere un problema
NP-complete
in tempo Deterministic Polynomial (P tempo)
Allora avremo che :

$$P = NP$$

Osservazione :

Se proviamo che
non possiamo risolvere un problema
NP-complete
in tempo Deterministic Polynomial (P tempo)
Allora abbiamo:

$$P \neq NP$$

Teorema diCook':

Il problema della soddisfacibilità è NP-complete

Dimostrazione:

Convertire una Non-Deterministic Turing Machine

In una espressione booleana

in (congiuntiva) conjunctive normal form

Altri problemi NP-Complete :

- Il commesso viaggiatore
- Vertex cover
- Hamiltonian Path

Tutti questi problemi si possono ridurre al problema della soddisfacibilità

Osservazione:

sarebbe molto strano che NP-complete problemi sono in P

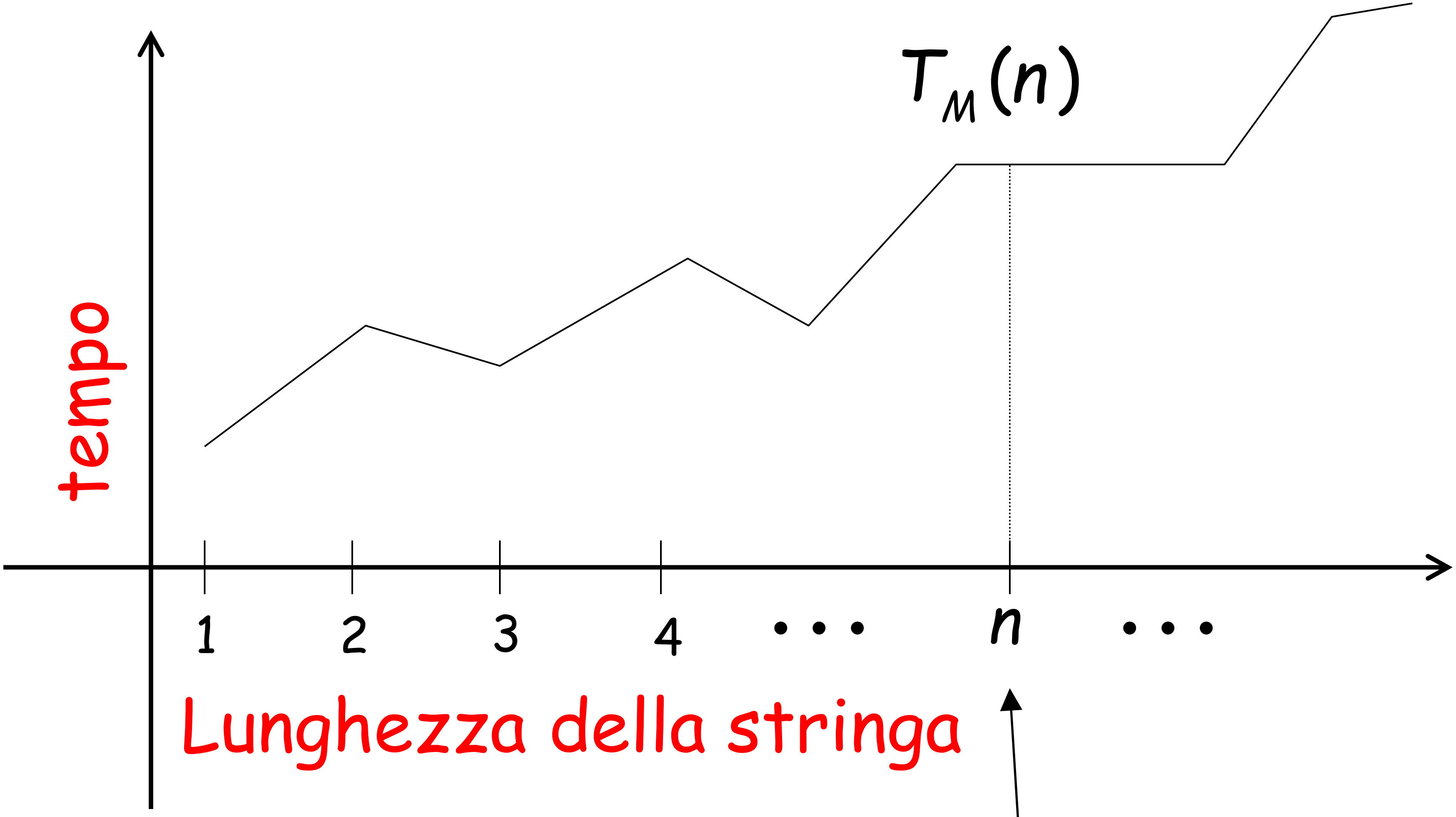
I problemi NP-complete hanno algoritmi in tempo esponenziale

Approssimazioni di questi problemi Sono in P tempo complessità:

Il numero di passi (step)
durante una computazione

spazio complessità:

spazio usato
durante una computazione



Massimo tempo per accettare una stringa di lunghezza n

algoritmi calcolabili
in tempo esponenziale

$$TIME(2^{n^k})$$

$$TIME(2^{2^{poly(k)}})$$

Let $t(n)$ be a function, where $t(n) \geq n$. Then every $t(n)$ time nondeterministic single-tape Turing machine has an equivalent $2^{O(t(n))}$ time deterministic single-tape Turing machine.

Rappresentano algoritmi intrattabili

Per alcuni input ci possono volere secoli per trovare la soluzione

tempo algoritmi Non-Deterministic Polynomial :

$$L \in NTIME(n^k)$$

algoritmi

Tempo Polinomiale Non-Deterministico

$$L \in NTIME(n^k)$$