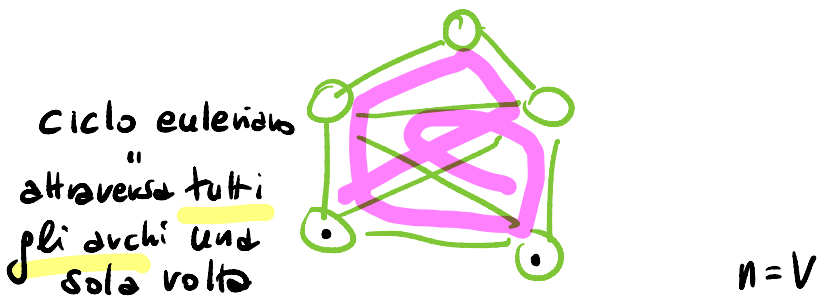


(IN)TRATTABILITÀ COMPUTAZIONALE

(trattabile = richiede tempo polinomiale)



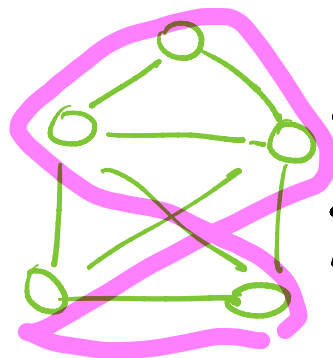
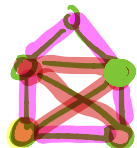
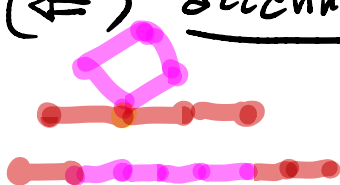
$G = (V, E)$ connesso $O(\text{poly}(n))$
 $\hookrightarrow O(m+n)$

[G euleriano sse tutti i nodi
hanno due (eventualmente) sono di
grado pari]

(\Rightarrow) immediata



(\Leftarrow) accorto



ciclo hamiltoniano
"attraversa tutti
i nodi una
sola volta"

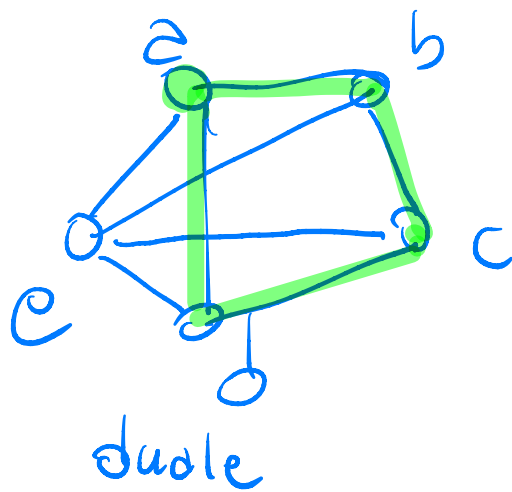
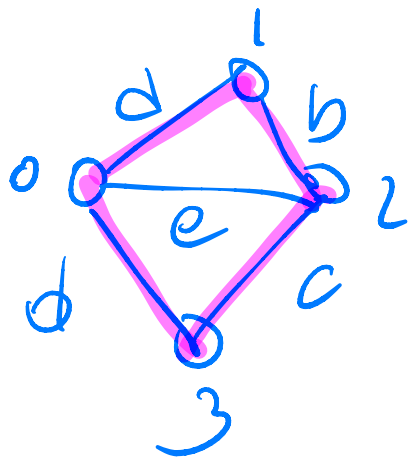
Genera tutte le $n!$ permutazioni dei nodi

Verifica se esiste una permutazione
 v_1, v_2, \dots, v_n t.c. $(v_i, v_{i+1}) \in E$ ed
(eventualmente) $(v_n, v_1) \in E$

costo
esponenziale

D1: ma è esponenziale? boh

D2: ma allora è polinomiale? boh



Problemi decisionali \rightarrow Sì o No

Ogni input può essere visto come una opportuna sequenza binaria in Σ^* , $\Sigma = \{0,1\}$

Problema decisionale $\Pi \in \Sigma^*$

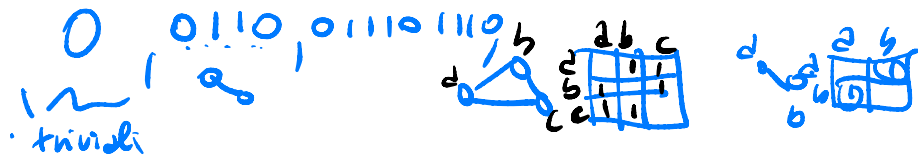
$$\pi = \{x \in \Sigma^* \mid \pi(x) = s\}$$

Esempio: $G = (V, E)$ può essere visto come una sequenza binaria:

prendiamo la matrice di adiacenze di G e concateniamo le sue righe \Rightarrow sequenza di $|V|^2$ bit

HAY = problema di stabilità se G è hamiltoniano

$$HAM = \{x \in \Sigma^{n^2} \mid \text{il grafo ottenuto da } x \text{ come matrice d'adjacenza} \\ \text{è hamiltoniano}\} \subseteq \Sigma^*$$



Classi P e NP

$\pi \in P$ se esiste un algoritmo deterministico polinomiale $|x|$
per stabilire se $x \in \pi$

es. EULERIANO $\in P$, SORTING (decisionale) $\in P$, RICERCA BINARIA $\in P$

$\pi \in NP$ se esiste un verificatore V polinomiale per π :

\uparrow
non-deterministico
polinomiale

$\forall x \in \Sigma^*$ $\left\{ \begin{array}{l} \text{se } x \in \pi \text{ allora } \exists y \in \Sigma^*, |y| = \text{poly}(|x|) \text{ t.c. } V(x, y) = \text{SI} \\ \text{se } x \notin \pi \text{ allora } \forall y \in \Sigma^* : V(x, y) = \text{NO} \end{array} \right.$
certificato polinomiale

$P \subseteq NP$: un problema in P non solo si verifica in tempo poly
ma si trova la soluzione in tempo poly

$P \stackrel{?}{=} NP$: BIG OPEN PROBLEM

Colorazione mappe planari ; colore diverso
a paesi confinanti.
 $K = \# \text{ colori}$

$K=2$

Sì/No

paese = nodo

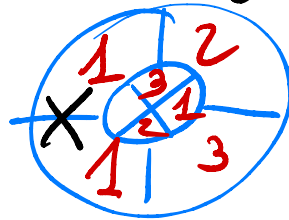
paese i confinante con
paese j \Rightarrow arco (i,j)

polinomiale
(pensateci!)

$K=3$

Sì/No

NP
(complete)



colori = $\{1, 2, 3\}$

$K \geq 4$

Sì

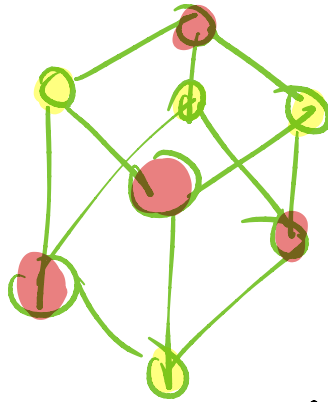
famoso teorema

Appel-Haken '77

ogni mappa planare
è 4-colorabile

poly

$k=2$



assegnare colore opposto
a quello del nodo
connesso

L'assegnamento di colori funziona se cerchiamo
di assegnare due colori diversi allo stesso nodo
L'unica decisione è il colore del primo nodo da
cui si parte, ma non cambia l'esito perché
sono 2 colori!

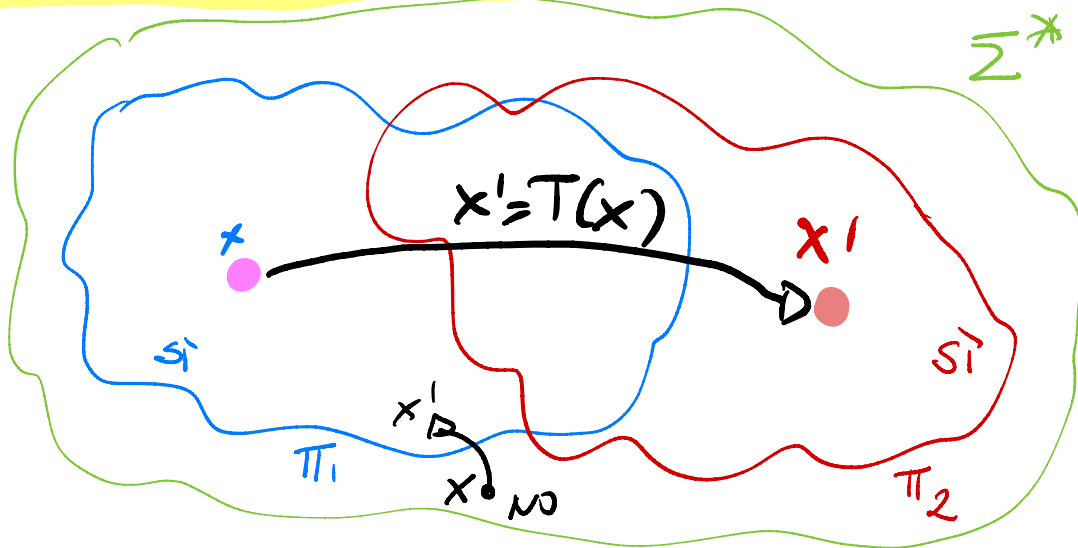
RIDUZIONE / TRASFORMAZIONE POLINOMIALE

$\Pi_1, \Pi_2 \in NP$

$$\Pi_1 \leq \Pi_2$$

se esiste un algoritmo deterministico polinomiale $T(x)$

t.c. $\forall x \in \Sigma^*: x \in \Pi_1 \iff \underbrace{T(x)}_{x'} \in \Pi_2$ (xx)



Note Non è richiesto che sia iniettiva o surgettiva

Esempio (non usiamo direttamente Σ^* ma ciò che rappresentano le sue stringhe)

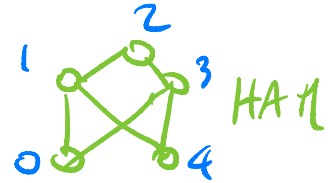
$\Pi_1 = \text{HAM}$ (ciclo hamiltoniano)

$\Pi_2 = \text{TSP}$ (travel salesperson problem)

n città numerate da 1 a n

costo D_{ij} = costo per andare dalla città i alla città j

↳ distanza km
↳ pedaggio
↳ tempo di percorrenza
↳ costo del carburante



$n=5$

TSP

	0	1	2	3	4
0			2	1	2
1				2	
2					2
3			1	2	
4		2		1	

D_{ij}

$K=n=5$

tour = permutazione c_1, c_2, \dots, c_n delle città

$$\text{costo}(\text{tour}) = \sum_{i=1}^{n-1} D_{c_i c_{i+1}} + D_{c_n c_1}$$

decisione: input n, D, K

output sì se $\exists \text{ tour} : \text{costo}(\text{tour}) \leq K$

Trasformazione T per π_1 & π_2

NON RISOLVE!

1) Input $G = (V, E)$ (lo stesso di $\pi_1 = \text{HAM}$)

2) Output: (input per $\pi_2 = \text{TSP}$)

► $n = |V|$

► $D_{ij} = \begin{cases} 0 & \text{se } i=j \\ 1 & \text{se } ij \in E \\ 2 & \text{altrimenti} \end{cases}$

► $K = |V|$

T è un algoritmo deterministico polinomiale
 $O(|V|^2)$ tempo

(x)

Per vedere (2.2)

G hamiltoniano \Leftrightarrow esiste tour di costo $\leq k = n$ in D

(\Rightarrow) G hamiltoniano $\Rightarrow \exists$ permutazione dei nodi v_1, v_2, \dots, v_n

t.c. $\forall i \in [n-1] : v_i v_{i+1} \in E, v_n v_1 \in E$

$\Rightarrow \exists$ tour v_1, v_2, \dots, v_n t.c. $D_{v_i v_{i+1}} = 1$ e $D_{v_n v_1} = 1$

$\Rightarrow \text{costo}(\text{tour}) = n \leq k$ per costruzione

(un ciclo di n nodi)
usa n archi

(\Leftarrow) G non hamiltoniano $\Rightarrow \forall$ permutazione c_1, c_2, \dots, c_n dei suoi nodi, esiste sempre una coppia $(c_i c_{i+1}$ oppure

$c_n c_1)$ non collegata da un arco: $\exists c_i c_{i+1} \notin E$ or

(altrimenti G sarebbe hamiltoniano!) $\exists c_n c_1 \notin E$

$\Rightarrow \forall$ tour $c_1, c_2, \dots, c_n : (\exists c_i c_{i+1} \text{ con } D_{c_i c_{i+1}} = 2 \text{ or}$

$\exists c_n c_1 \text{ con } D_{c_n c_1} = 2$
 $\text{costo}(\text{tour}) \geq 2 + (n-1) \cdot 1 > n$ (poiché $D_{ij} \geq 1$ per $i \neq j$)

Riduzione α :

① Riflessiva : $\pi \alpha \pi$ ($T = \text{id}$)

② Transitiva : $\pi_1 \alpha \pi_2 \alpha \pi_3$
 T_1 T_2

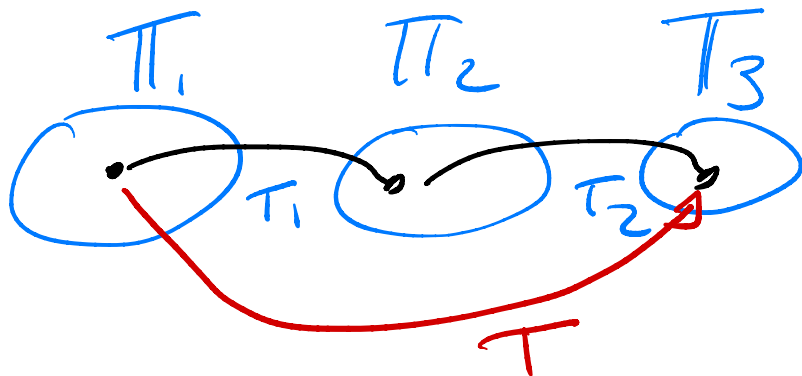
$$T = T_2 \circ T_1$$

$$T(x) = T_2(T_1(x))$$

T det. polinomiale

$$x \in \pi_1 \Leftrightarrow T(x) \in \pi_3$$

$$\begin{array}{ccc} \updownarrow & & \updownarrow \\ \underline{T_1(x)} \in \pi_2 & & T_2(\underline{T_1(x)}) \in \pi_3 \\ & \underbrace{\hspace{10em}}_{z = T_1(x)} & \end{array}$$



③ Simmetrica?

$$\pi_1 \alpha \pi_2 \quad \text{BOH?}$$

$$??? \Rightarrow \pi_2 \alpha \pi_1$$

Prop $\Pi_1 \leq \Pi_2 : \Pi_2 \in P \Rightarrow \Pi_1 \in P$

Se A_2 l'algoritmo polinomiale per Π_2

costruiamo un algoritmo polinomiale per Π_1 :

$\forall x \in \Sigma^*$ ① $x' = T(x)$ dove T è la trasform.
polin. $\Pi_1 \leq \Pi_2$

② Esegui A_2 su x'

③ rispondi $SI \Leftrightarrow A_2(x') = SI$

Letture interessate:

$\Pi_1 \notin P \Rightarrow \Pi_2 \notin P$

Cook '71, Levin '71

π è NP-completo: (NPC)

1) $\pi \in NP$

2) $\forall \pi' \in NP : \pi' \leq \pi$

Th Cook-Levin: $SAT \in NPC$

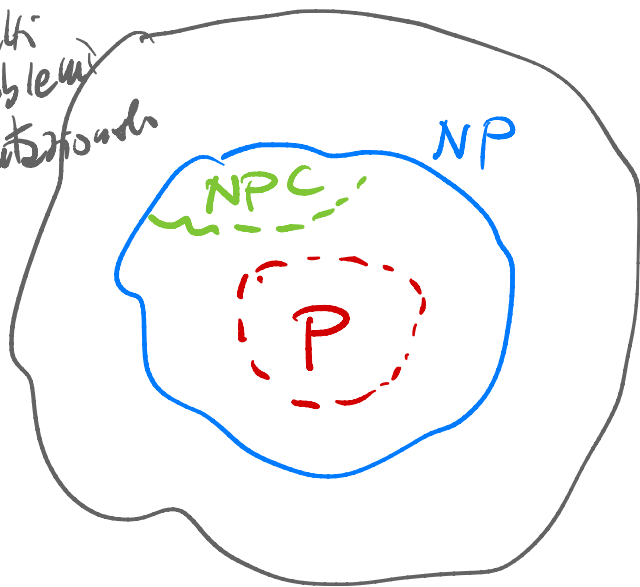
SAT n variabili booleane x_1, x_2, \dots, x_n , $x_i \in \{0, 1\}$
F T

Letterali: x_i, \bar{x}_i

Clausa: $(x_i \vee \bar{x}_j \vee x_n)$ OR

Formula CNF: AND di clause: $(x_1 \vee \bar{x}_4) \wedge (x_2 \vee \bar{x}_3 \vee x_4)$
 $F(x_1, \dots, x_n)$ C_1 C_2

tutti
i problemi
computazionali



F è soddisfacibile se \exists un assegnamento di verità $\tau: [n] \rightarrow \{0,1\}$
t.c. alla variabile x_i viene assegnato il valore $\tau(i)$

$$F = (x_1 \vee x_2) \wedge (x_3 \vee \overline{x_4} \vee x_5) \wedge (\overline{x_1} \vee \overline{x_3} \vee \overline{x_5})$$

$$\tau: x_1 \leftarrow 0 \quad x_2 \leftarrow 1 \quad x_3 \leftarrow 0 \quad x_4 \leftarrow 0 \quad x_5 \leftarrow 1$$

SAT: F, n

stabilire se F è soddisfacibile

ci sono 2^n possibili τ

Karp '72

$\pi \in \text{NPC}$:

1) $\pi \in \text{NP}$

2) $\exists \pi^* \in \text{NPC}$ t.c. $\pi^* \leq \pi$

Lo siccome vale la proprietà transitive

$\forall \pi' \in \text{NP} : \pi' \leq \pi^*$ perché $\pi^* \in \text{NPC}$

$\pi^* \leq \pi \Rightarrow \forall \pi' \in \text{NP} : \pi' \leq \pi$

La proprietà simmetrica vale per i problemi in NPC