Complessità Strutturale e NP-Completezza

Una breve introduzione

Francesco De Rose Alberto Giorgi

Università degli Studi di Brescia
Facoltà di Ingegneria
Corso di Laurea Specialistica in Ingegneria Informatica

Anno Accademico 2005-2006

Il problema dei problemi

Non tutti problemi sono uguali...

- ...alcuni non sono risolvibili da un calcolatore
  - es: problema della fermata
- ...di quelli risolvibili:
  - alcuni lo sono in tempo polinomiale ($O(n^k)$)
  - altri richiedono un tempo superpolinomiale, cioè $\forall k : T = \omega(n^k)$ (problemi intrattabili)
  - per altri ancora non conosciamo lo stato (NP-completi)

Voci di corridoio

- È stato dimostrato che se $\exists$ un algoritmo t.p. per un problema NP-completo (NPC) allora esistono algoritmi polinomiali per tutti i problemi NPC.
  - Per questo motivo la ricerca sull'argomento è molto intensa.
- Non ci sono ancora risultati definitivi, la maggior parte della comunità scientifica crede tuttavia che i problemi NPC siano intrattabili.

Motivazioni

Perché studiare la NP-completezza?

- Stabilire che un problema è NPC ci porta alla quasi certezza pratica della sua intrattabilità.
- Può convenire quindi, qualora si debba affrontare un problema NPC, dedicarsi alla ricerca di un algoritmo approssimato.

La classe degli algoritmi tempo-polinomiali

Quello di classe di algoritmi tempo-polinomiali **P** è un concetto più filosofico che matematico, ma non privo di interesse pratico.

La classe degli algoritmi tempo-polinomiali

Infatti:

- Nella realtà il grado del polinomio è praticamente limitato
- Algoritmi eseguiti in t.p. in un modello computazionale (e.g. Macchina di Turing) spesso rimangono t.p. se eseguiti in un altro modello computazionale (e.g. macchine seriali)
- Poiché $\mathbb{P}$ (spazio dei polinomi) è chiuso rispetto a $+, \cdot$ e $\circ$ (composizione) la classe di algoritmi **P** eredita interessanti proprietà di chiusura.
  $\Rightarrow$ la composizione di due algoritmi t.p. è sempre t.p.

Problema astratto

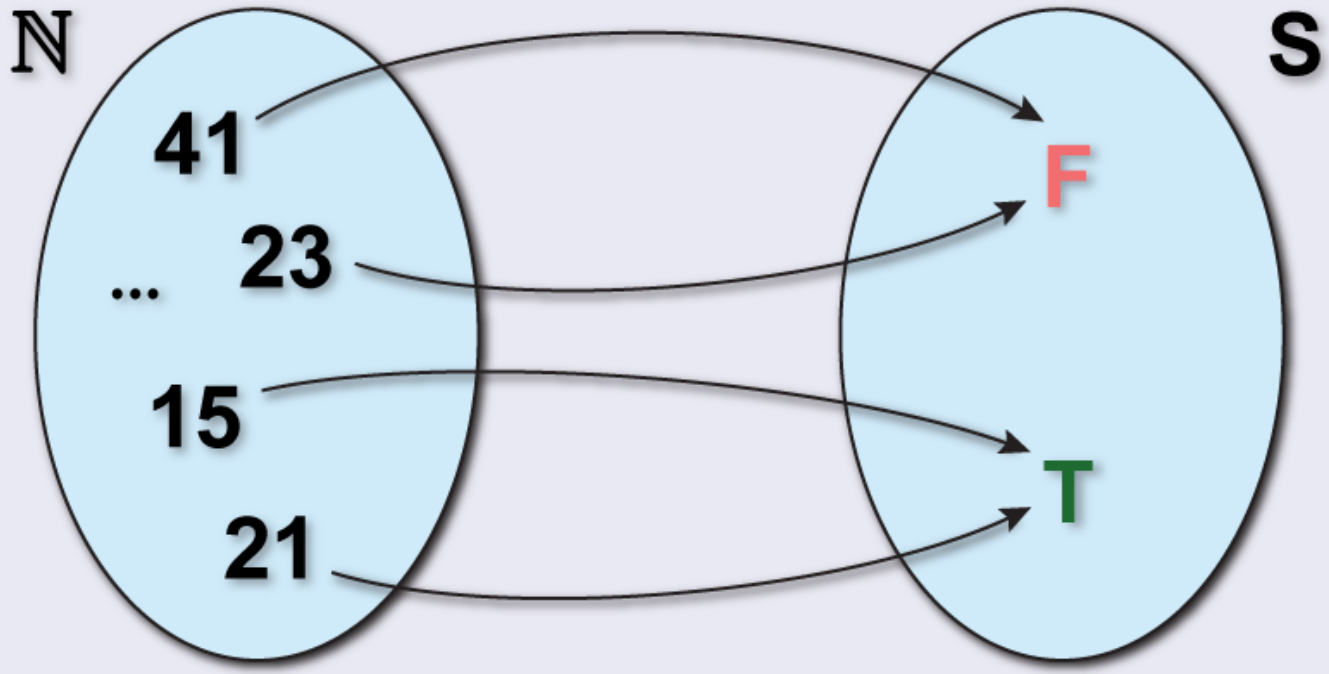
Definizione

- *Un problema astratto è una relazione tra l'insieme delle istanze I e quello delle soluzioni S, vale a dire: $Q \subseteq I \times S$.*
- *Se $S = \{true, false\}$ allora è un problema di decisione.*

La classe degli algoritmi tempo-polinomiali

Esempio

Dato $I = \mathbb{N}, S = \{true, false\}$, vogliamo decidere se un naturale è divisibile per 3.



Dall'astratto al concreto: la codifica

- La definizione di problema astratto è bella ed elegante...
- ...ma noi dobbiamo risolvere problemi reali con macchine reali!
- Ci serve quindi un modo di trasformare un problema astratto in un problema concreto, tramite qualche opportuna funzione di codifica.

Esempio (Codifica binaria dei naturali)

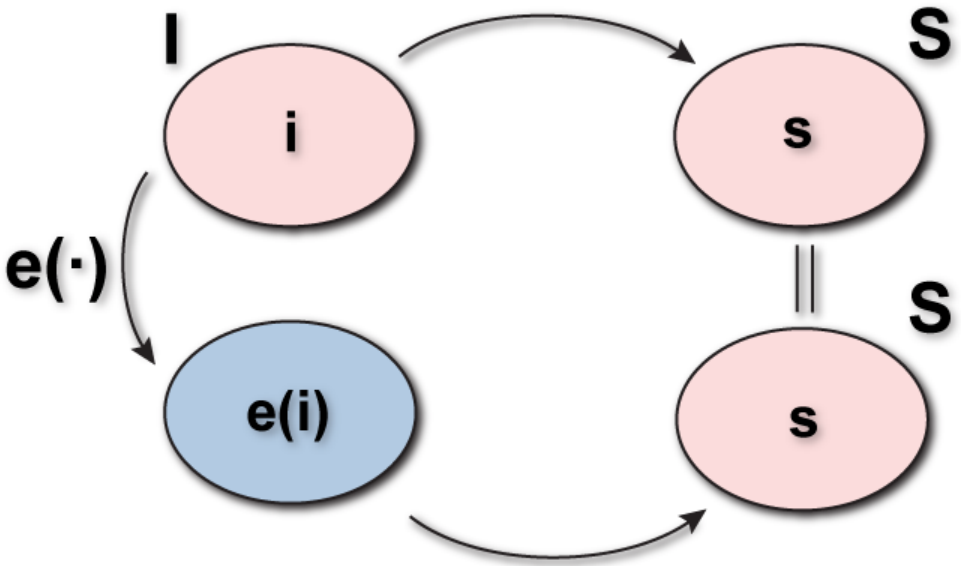
$\Sigma = \{0, 1\}$
$e : \mathbb{N} \rightarrow \Sigma^*$

$1 \longmapsto < 1 >$
$2 \longmapsto < 10 >$
$3 \longmapsto < 11 >$
$4 \longmapsto < 100 >$

Codifica dei problemi

Definizione

- *Sia dato un alfabeto Σ. Allora una funzione $e : I \rightarrow \Sigma^*$ è detta funzione di codifica.*
- *Solitamente $\Sigma = \{0, 1\}$ ma, come vedremo in seguito, ciò non è particolarmente rilevante.*



Problema concreto

Definizione (Problema concreto)

- *Sia $Q \subseteq I \times S$ un problema astratto;*
- *definiamo problema concreto P_c l'insieme delle coppie ordinate istanza codificata-soluzione, ovvero:*

$$P_c = \{(e(i), s) : (i, s) \in Q\}.$$

Equivalenza delle codifiche ragionevoli

Definizione

Una codifica $e : I \rightarrow \Sigma^*$ è detta ragionevole se:

1. $|\Sigma| \geq 2$;
2. *non introduce dati irrilevanti, né richiede generazione esponenziale di dati per rappresentare un'istanza del problema.*

- In realtà quasi tutte le codifiche "naturali" sono ragionevoli.
  - L'esempio più comune di codifica irragionevole è l'unario.

Equivalenza delle codifiche ragionevoli

Esempio

n	Codifica binaria	Codifica unaria
1	$< 1 >$	$< 1 >$
2	$< 10 >$	$< 11 >$
3	$< 11 >$	$< 111 >$
4	$< 100 >$	$< 1111 >$
5	$< 101 >$	$< 11111 >$
6	$< 110 >$	$< 111111 >$
7	$< 111 >$	$< 1111111 >$

Equivalenza per problemi astratti

Teorema (Equivalenza per problemi astratti)

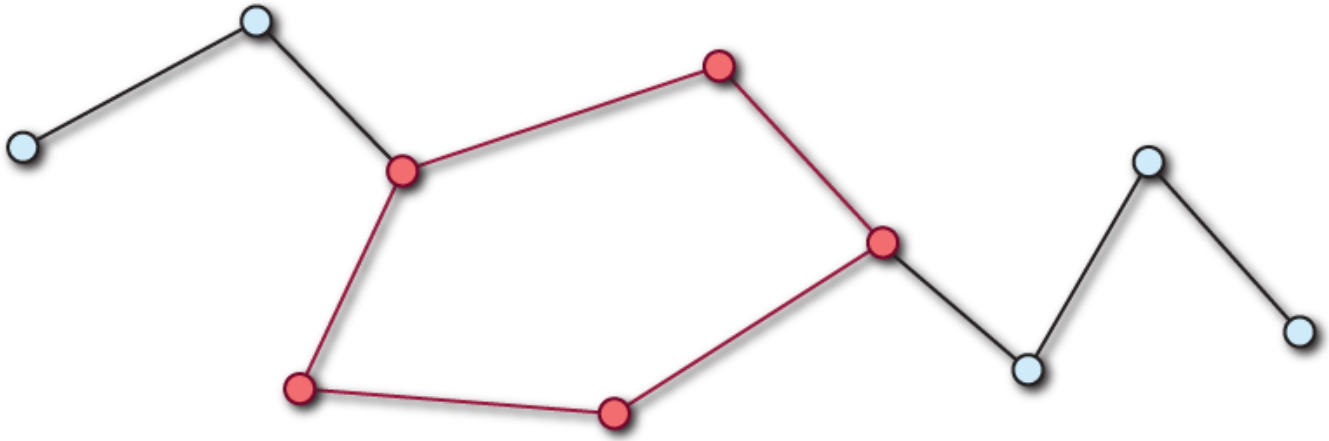
- *Sia Q un problema di decisione astratto su un insieme I di istanze.*
- *Siano e_1, e_2 due codifiche di I correlate polinomialmente*
  - *è possibile trasformare l'una nell'altra in tempo polinomiale*
- *Allora:*

$$e_1(Q) \in P \iff e_2(Q) \in P$$

Ciclo Hamiltoniano

Ripassino introduttivo (per chi dormiva durante R.O.)

Un ciclo di un grafo x è hamiltoniano se è semplice e contiene tutti i vertici di x



Algoritmi di verifica I

Cos'è un algoritmo di verifica?

- Supponiamo siano dati un grafo x e un cammino y.
- E' molto semplice verificare se y è un ciclo hamiltoniano di x (Come si fa?).
- Si dice che y è un certificato che x è hamiltoniano (cioè che ammette un ciclo hamiltoniano).

Algoritmi di verifica II

- I certificati esistono anche per i problemi di decisione su un linguaggio L.
- In questo caso l'algoritmo di verifica A è tale che

$$\forall \text{ grafo } x, \exists \text{ cammino } y : A(x, y) = 1 \iff x \in L.$$

Classe NP

- La classe di complessità NP comprende tutti e soli i linguaggi verificabili con un algoritmo in tempo polinomiale.
- In realtà è necessario anche limitare la lunghezza dei certificati $|y| = O(|x|^c)$, per qualche c.

Esempio

Il linguaggio ciclo-hamiltoniano $\in$ NP.

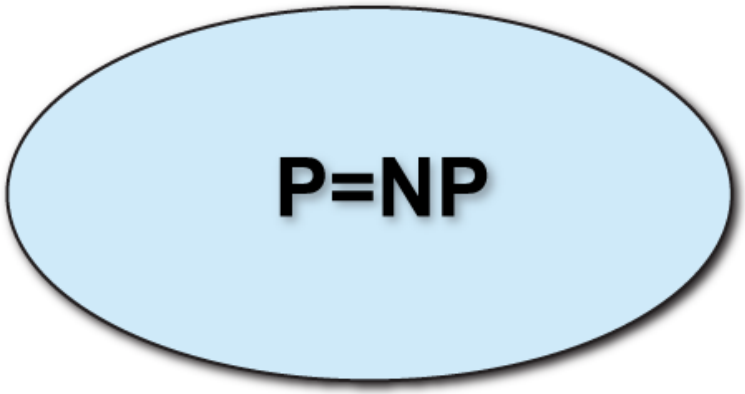
- Osserviamo inoltre che per qualsiasi linguaggio $L \in P \Rightarrow L \in NP$: l'algoritmo di verifica può semplicemente ignorare il certificato e risolvere il problema...

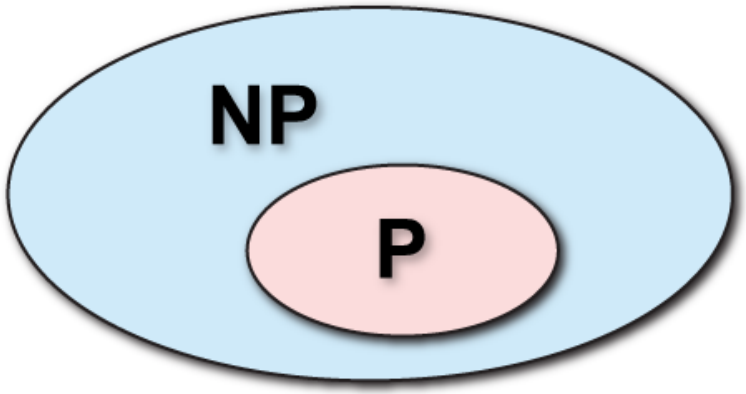
Qualche considerazione

Per ora possiamo concludere che:

- abbiamo mostrato che $P \subseteq NP$. Vale anche il viceversa?
- finora nessuno è riuscito a dimostrare che $NP \subseteq P$ (ovvero $P = NP$) ...
- ...ma neppure che $NP \not\subseteq P$ (ovvero $P \neq NP$)

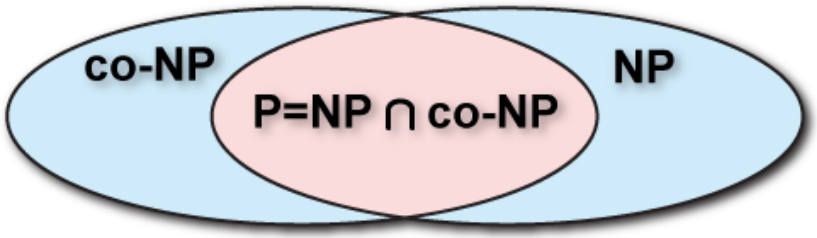
P = NP?

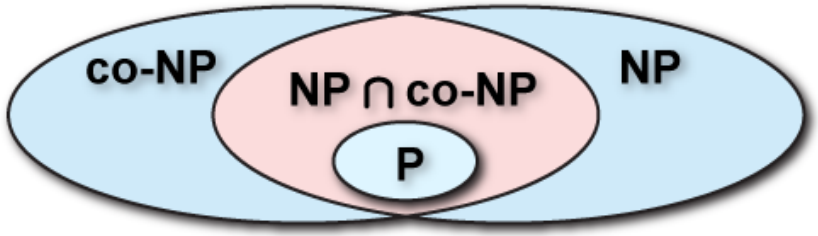




- $\Rightarrow$ La tendenza generale è di ritenere che $P \neq NP$.
- $\Rightarrow$ Ciò è d'accordo con l'idea intuitiva che verificare una soluzione è spesso più semplice che costruirla.

P = NP ∩ co-NP?





Un ulteriore problema aperto è il seguente: NP = co-NP?

- La classe di complessità co-NP è tale per cui

$$L \in \text{co-NP} \iff \overline{L} \in \text{NP}.$$

- Ancora una volta non è noto se P = NP ∩ co-NP o se c'è qualche linguaggio ∈ NP ∩ co-NP – P.

Perché studiare la classe NP

La teoria dell'NP-completezza studia soltanto i problemi di decisione:

- questo aspetto non è lesivo della generalità della teoria;
- infatti ogni problema di ottimizzazione non è più facile del relativo problema di decisione;
- come vedremo dopo, si dimostra che il problema di decisione è difficile.

Riducibilità in tempo polinomiale I

Definizione (Riducibilità in tempo polinomiale)

- *Un linguaggio L_1 è polinomialmente riducibile a L_2 (e si scrive $L_1 \leq_p L_2$) se:*

$$\exists f : \{0,1\}^* \rightarrow \{0,1\}^*,$$

calcolabile in tempo polinomiale, tale che

$$\forall x : x \in L_1 \iff f(x) \in L_2$$

- *Si dice che f è la funzione di riduzione, calcolata da un algoritmo di riduzione.*

Riducibilità in tempo polinomiale II

- In altre parole, decidere se $x \in L_1$ equivale a decidere se $f(x) \in L_2$ pagando, dal punto di vista del tempo di computazione, un sovrapprezzo polinomiale.
- Vale a dire, se $L_1 \leq_p L_2$ allora L_1 è non più arduo di L_2, a meno di un fattore polinomiale.
  - Ah, ecco il perché del simbolo $\leq_p$!

Linguaggio NP-completo

Definizione (Linguaggio NP-completo)

L è NP completo se:

1. $L \in NP$
2. $L' \leq_p L \quad \forall L' \in NP$
   - *ogni altro problema NP non è più arduo di L*

Se L soddisfa solo la prop. 2 si dice NP-arduo

Problemi aperti e prospettive

Sia dato $L \in NPC$:

- supponiamo di aver scoperto un algoritmo tempo–polinomiale per risolvere L;
- poiché $\forall L' \in NP, L' \leq_p L$, allora avremmo dimostrato che $P = NP$!
- inoltre, se ogni L' non è risolvibile in tempo polinomiale, allora tutti gli NP-completi non sono risolvibili in t.p.

Problemi NP-completi

Passiamo ora in rassegna alcuni tra i più studiati problemi NP-completi:

- ciclo hamiltoniano
- SAT e 3-SAT
- problema della cricca
- soddisfacibilità dei circuiti
- somma di sottoinsieme

Il problema SAT

Problema

- Assegnata una formula booleana in forma normale congiuntiva della logica del primo ordine, stabilire se esiste un assegnamento delle variabili che rende vera la formula.
- Questo problema è noto come SAT (SATisfiability).
- In particolare un *n*-SAT è un SAT in cui ogni clausola è formata da esattamente *n* letterali.

Esempio

Trovare un assegnamento che soddisfa :

$$F = (\neg x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee \neg x_5)$$

Teorema di Cook

Teorema (Cook)

SAT è NP-completo

Dimostrazione.

`http://portal.acm.org/citation.cfm?id=805047`

3-SAT $\in$ NPC

Il problema 3-SAT

- Una formula 3-CNF è un caso particolare di CNF, quindi un assegnamento di verità può essere verificato in tempo polinomiale. Pertanto 3-SAT $\in$ NP.
- Resta da dimostrare che $\forall L \in NP, L \leq_p$ 3-SAT.
- Ci proponiamo di farlo riconducendoci a SAT, in altre parole, dimostrando che SAT $\leq_p$ 3-SAT.
- Poiché SAT è NP-completo, conseguirà che

$$L \leq_p SAT \leq_p 3\text{-SAT}$$

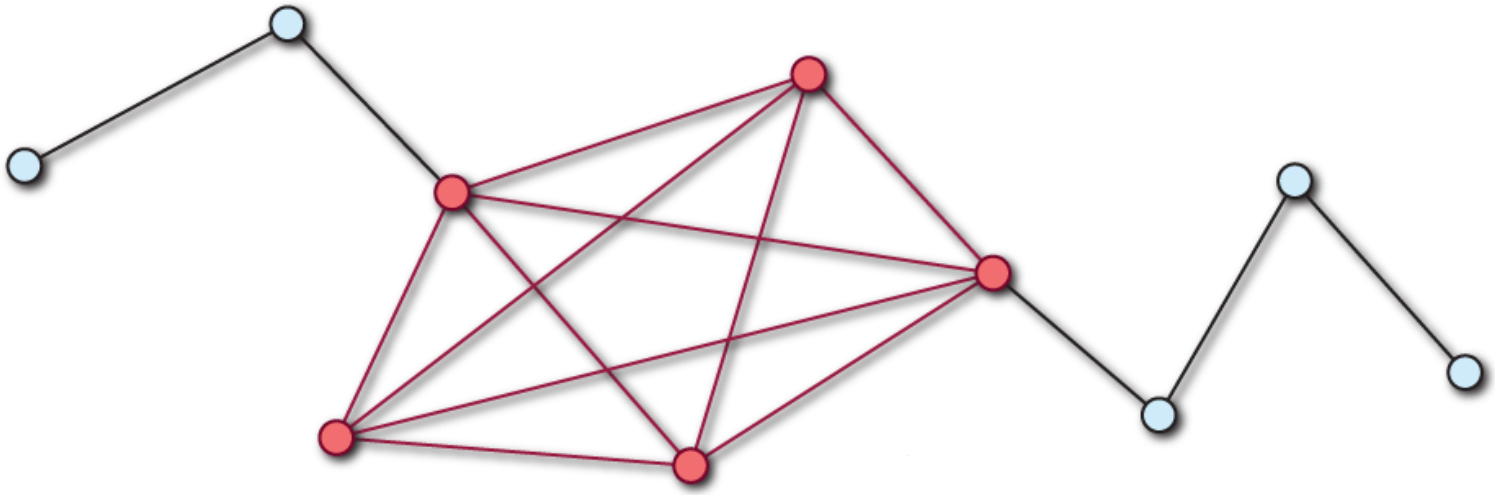
3-SAT $\in$ NPC

Dimostrazione.

- A partire da una generica clausola C $\in$ SAT, vogliamo costruire un'istanza del problema 3-SAT.
- Per ogni clausola C si distinguono tre casi:
  - se C ha esattamente 3 letterali, non necessita di conversione;
  - se C ha meno di tre letterali, si ripete un qualunque letterale di C tante volte quanto basta affinché C' abbia tre letterali; es: $C = (x_1 \vee x_2) \Rightarrow C' = (x_1 \vee x_2 \vee x_2)$
  - se C ha più di 3 letterali $C = (\alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_n)$, convertiamo la clausola nell'insieme $C' = (\alpha_1 \vee \alpha_2 \vee z_1) \wedge (\neg z_1 \vee \alpha_3 \vee z_2) \wedge \cdots \wedge (\neg z_{n-3} \vee \alpha_{n-1} \wedge \alpha_n)$

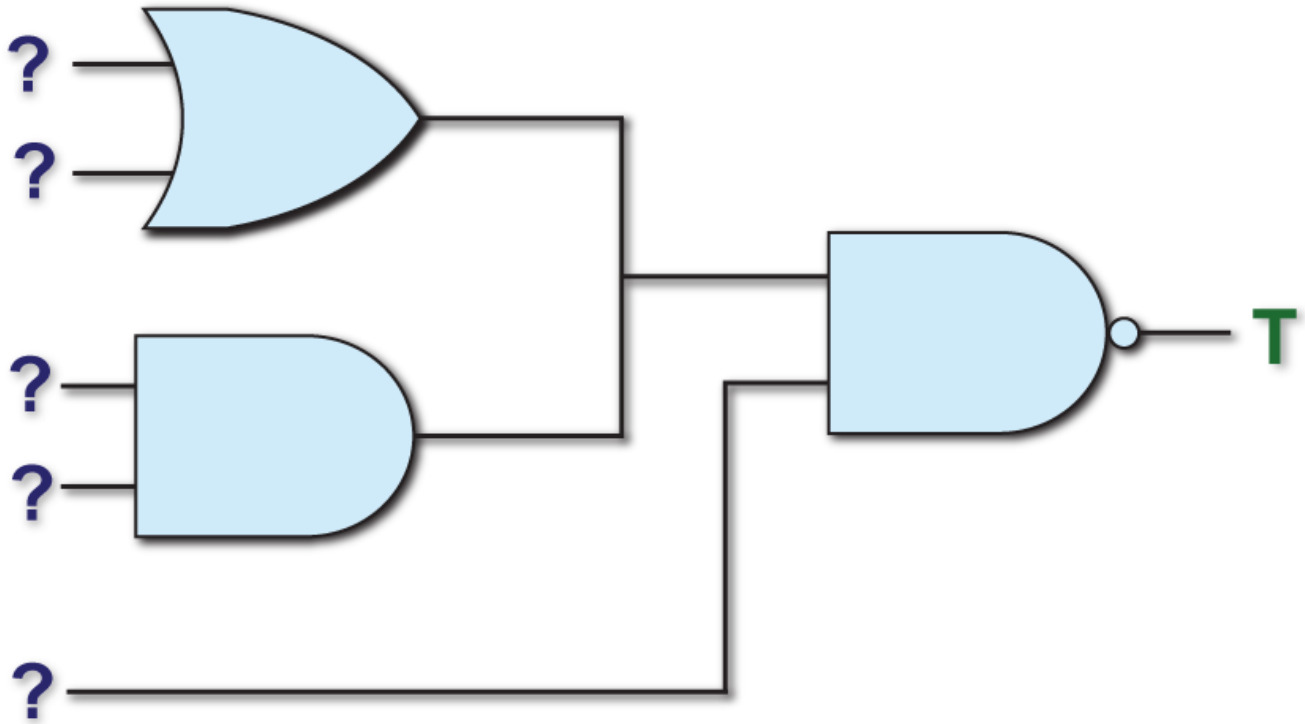
Il problema della cricca

- Dato un grafo non orientato G, esiste un sottografo C (cricca) di dimensione k che è completamente magliato?



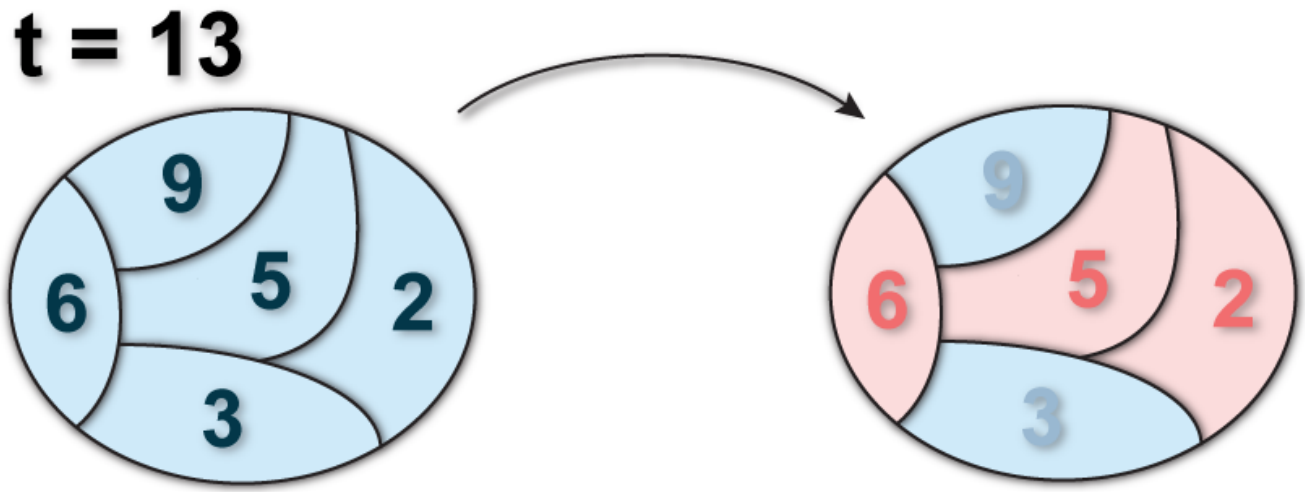
Soddisfacibilità di circuiti

- Dato un circuito combinatorio booleano, esiste un assegnamento degli ingressi che rende vera l'uscita?



Somma di sottoinsieme

- Dato un insieme di naturali S ed un obiettivo t, esiste un sottoinsieme di S i cui elementi hanno per somma esattamente t?



Bibliografia ed approfondimenti

Th. H. Cormen, et al.
Introduzione agli algoritmi.
2ª edizione italiana, Jackson Libri, 1994.

Stephen A. Cook
The complexity of theorem-proving procedures.
ACM Press, New York, NY, USA , 1971.