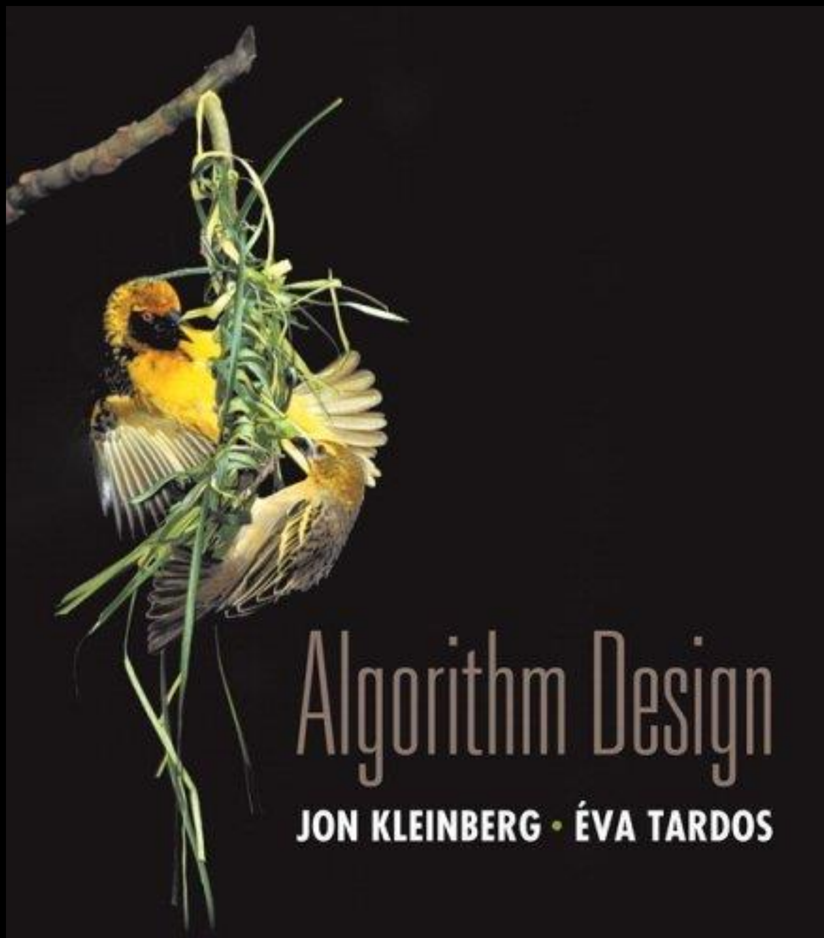


Chapter 8

NP e Computational Intractability

Supponete di lavorare per un distributore di posta. Un giorno il vostro capo vi chiede di di trovare un nuovo metodo per la distribuzione della posta/pacchi per ognuno dei "postini" in modo da minimizzare il costo.

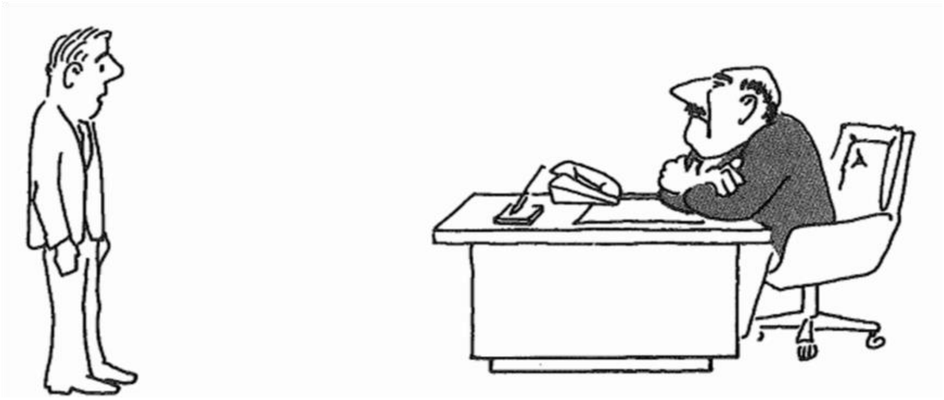
Vi mettete a lavoro

Guardate un pò gli algoritmi noti

Pensate a vari algoritmi

Non vi viene in mente niente di meglio che …

…. potete andare dal capo



Mi spiace, ma non ci riesco

Pessima idea, vi può credere un incapace e licenziarvi!

Oppure…

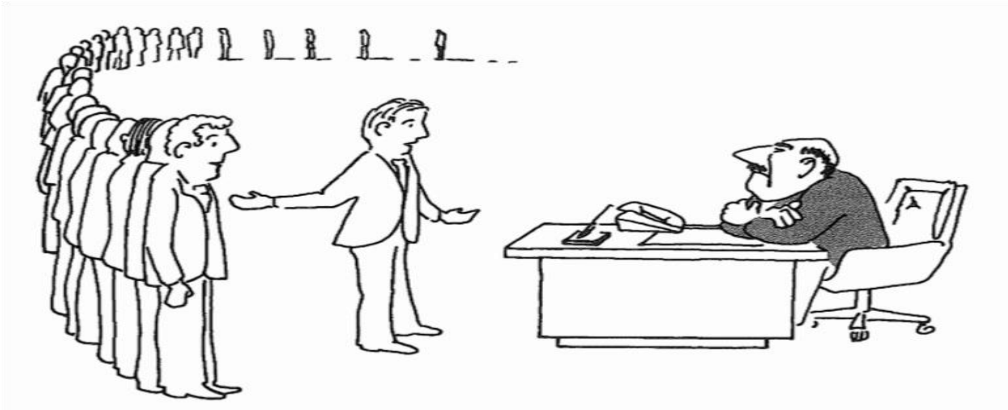
…. potete andare dal capo

Non ci riesco, non è possibile

Cattiva idea, non avete prove!

Oppure…

… potete dimostrare che il problema è "NP-Completo"



Non ci riesco, ma

anche nessuno di questi famosi scienziati ci è riuscito!

Illustrazioni da Garey & Johnson, A Guide to the Theory of NP-completeness, 1979

Complessità Algoritmi

- **Indecidibilità** — Nessun algoritmo possibile
- **Classe P** — $O(n^k)$ possibile
- NP-completezza — $O(n^k)$ non probabile

Classificazione dei problemi

Desiderata. Classificazione dei problemi in accordo un quelli che possono essere risolti in tempo polinomiale e quelli che non possono essere risolti in tempo polinomiale .

Abbiamo definito la classe P

Non è stato possibile classificare per decenni tantissimi problemi fondamentali

Vedremo che questi problemi sono "computazionalmente equivalenti"

Riduzioni Polinomiali

Desiderata. Supponiamo che possiamo risolvere X in tempo polinomiale. Quali altri problemi possiamo risolvere in tempo polinomiale?

Non confondere con si riduce da

Riduzione. Problema X si riduce in tempo polinomiale al problema Y se arbitrarie istanze di X possono essere risolte usando:
- Un numero polinomiale di passi di computazione, più
- un numero polinomiale di chiamate ad un oracolo che risolve Y.

Una "black box" che risolve istanze di Y in un solo passo

Notazione. $X \leq_P Y$.

Nota.
- Si usa tempo anche per scrivere l'istanza data in input alla black box

  $\Rightarrow$ instanze di Y devono avere dimensione polinomiale.

Riduzioni Polinomiali

Scopo. Classificare i problemi in accordo alla difficoltà relativa.

Fornire algoritmi. Se $X \leq_P Y$ e Y ammette soluzione in tempo polinomiale, allora anche X ammette soluzione in tempo polinomiale.

Stabilire intrattabilità. Se $X \leq_P Y$ e X non ammette soluzione in tempo polinomiale, allora Y non ammette soluzione in tempo polinomiale.

Stabilire equivalenza. Se $X \leq_P Y$ e $Y \leq_P X$, allora usiamo $X \equiv_P Y$.

a meno del costo della riduzione

Riduzione mediante equivalenza semplice

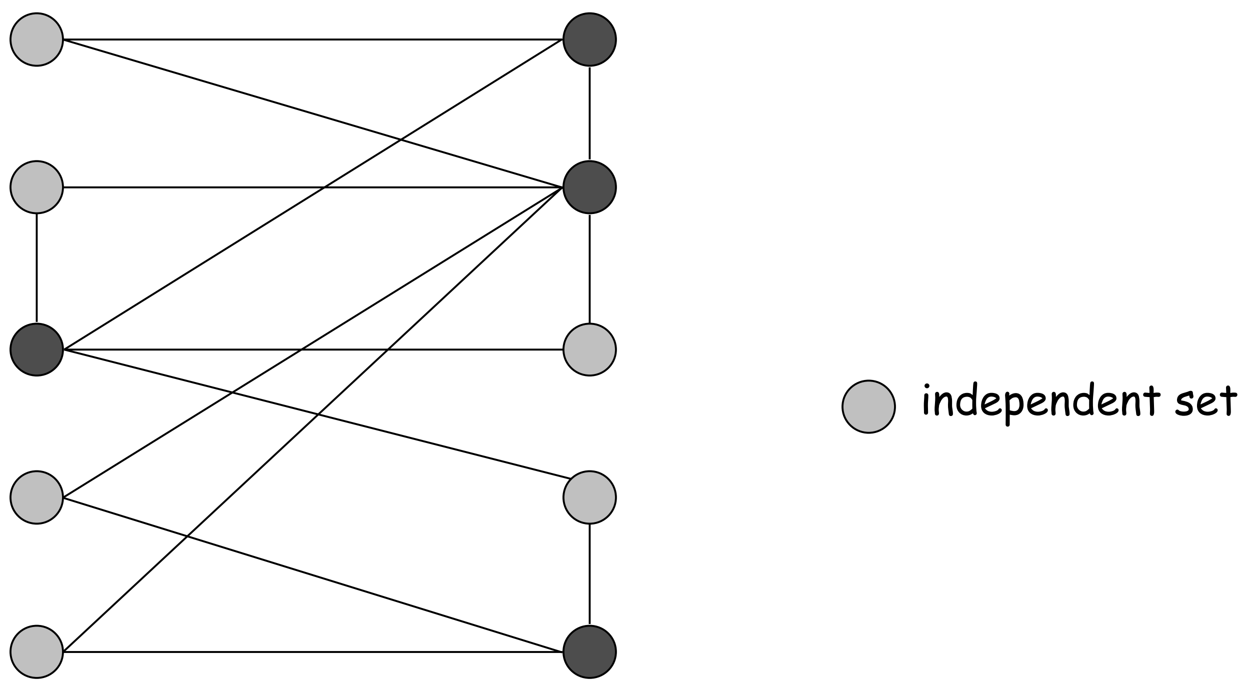
Riduzioni: strategie di base.

- Riduzione mediante equivalenza semplice.
- Riduzione da caso speciale a caso generale.
- Riduzione mediante codifica con gadgets.

Independent Set (Insieme Indipendente)

INDEPENDENT SET: dato un grafo $G = (V, E)$ e un intero k, esiste un sottinsieme di vertici $S \subseteq V$ tale che $|S| \geq k$, e per ogni edge al più uno di suoi estremi è in S?

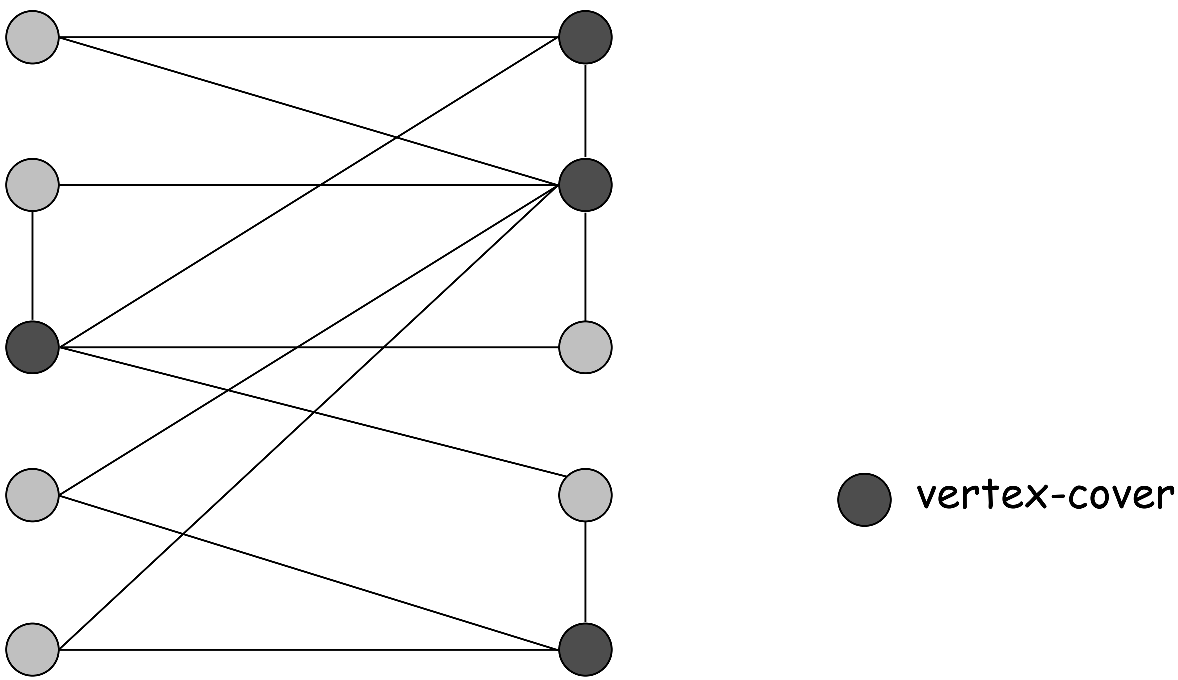
Es. esiste un insieme indipendente di dimensione ≥ 6? Si.
Es. esiste un insieme indipendente di dimensione ≥ 7? No.



Vertex-cover

Vertex-cover: dato un grafo $G = (V, E)$ e un intero k, esiste un sottinsieme di vertici $S \subseteq V$ tale che $|S| \le k$, e per ogni edge, al meno uno di suoi estremi è in S?

Ex. esiste un vertex-cover di dimensione ≤ 4? Si.
Ex. esiste un vertex-cover di dimensione ≤ 3? No.

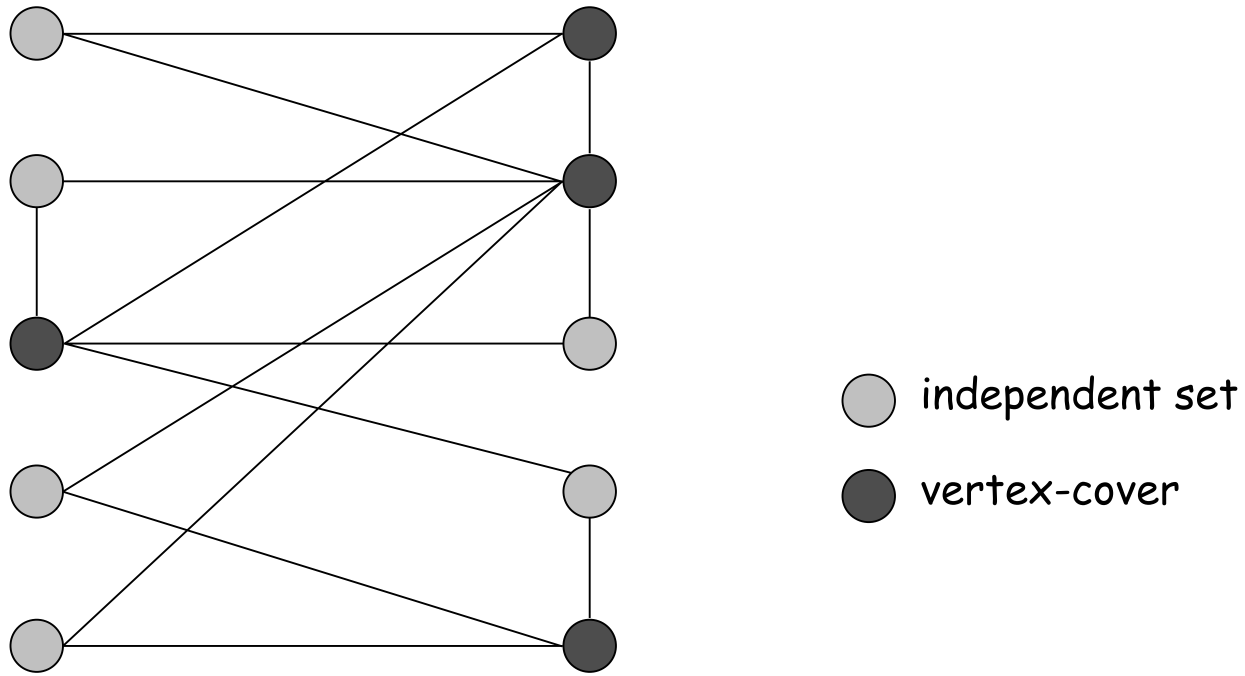


Vertex-cover e Independent set

Fatto. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.

Dim. Mostriamo che

S è un insieme indipendente sse V – S è un vertex-cover.



Vertex-cover e Independent set

Fatto. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.

Dim. Mostriamo S è un insieme indipendente sse $V - S$ è un vertex-cover.

$\Rightarrow$

- Sia S un qualsiasi independent set.
- Consideriamo un arbitrario edge (u, v).
- S indipendente $\Rightarrow u \notin S$ o $v \notin S$ $\Rightarrow u \in V - S$ o $v \in V - S$.
- Quindi, $V - S$ copre (u, v).

$\Leftarrow$

- Sia $V - S$ un qualsiasi vertex-cover.
- Consideriamo due nodi $u \in S$ e $v \in S$.
- Notiamo che $(u, v) \notin E$ poichè $V - S$ è un vertex-cover.
- Quindi, non esistono due nodi in S connessi da un edge $\Rightarrow$ S independent set.

Riduzione da caso speciale a caso generale

Strategie di riduzione

- Riduzione mediante equivalenza semplice.
- Riduzione da caso speciale a caso generale.
- Riduzione mediante codifica con gadgets.

Set Cover

SET-COVER: dato un insieme U di elementi, una collezione $S_1, S_2, \ldots, S_m$ di sottoinsiemi di U e un intero k, esiste una collezione di $\leq k$ di questi insiemi la cui unione è uguale a U?

$U = \{1, 2, 3, 4, 5, 6, 7\}$	
$k = 2$	
$S_1 = \{3, 7\}$	$S_4 = \{2, 4\}$
$S_2 = \{3, 4, 5, 6\}$	$S_5 = \{5\}$
$S_3 = \{1\}$	$S_6 = \{1, 2, 6, 7\}$

Esempio di applicazione.

- m parti di software disponibili
- Insieme U di n specifiche che desideriamo per il nostro sistema.
- La parte i-ma di software fornisce l' insieme $S_i \subseteq U$ di specifiche.
- Goal: ottenere tutte le n specifiche usando il minimo numero di parti di software.

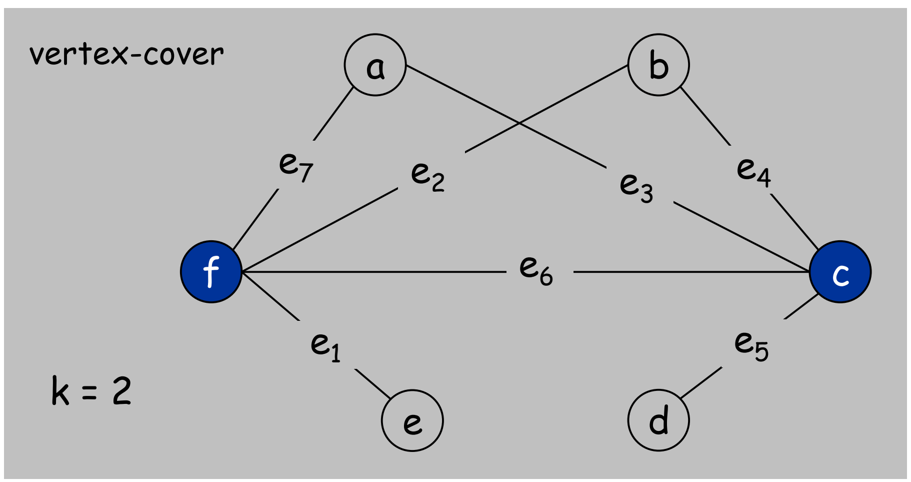
Vertex-cover si riduce a Set-Cover

Fatto. VERTEX-COVER $\leq_P$ SET-COVER.
Dim. Data un' istanza di VERTEX-COVER $G = (V, E)$, k,
costruiamo un' istanza di set-cover la cui dimensione è uguale alla dimensione dell' istanza di vertex-cover .

Costruzione.

- Creiamo istanza di SET-COVER :

$$k = k, \quad U = E, \quad S_v = \{e \in E : e \text{ incidente su } v\}$$



SET-COVER

$U = \{1, 2, 3, 4, 5, 6, 7\}$
$k = 2$
$S_a = \{3, 7\}$ $\quad S_b = \{2, 4\}$
$S_c = \{3, 4, 5, 6\}$ $\quad S_d = \{5\}$
$S_e = \{1\}$ $\quad S_f = \{1, 2, 6, 7\}$

Set-cover di dimensione $\leq k$ sse vertex-cover di dimensione $\leq k$. ▪

8.2 Riduzioni via "Gadgets"

Strategie di riduzione

- Riduzione mediante equivalenza semplice.
- Riduzione da caso speciale a caso generale.
- Riduzione via "gadgets."

Soddisfacibilità

Letterale: una variabile Booleana o la sua negazione. x_i or $\overline{x_i}$

Clausola: un OR (o disgiunzione) di letterali. $C_j = x_1 \vee \overline{x_2} \vee x_3$

Forma Normale Congiuntiva (CNF): una formula proposizionale Φ che è un AND (o congiunzione) di clausole. $\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

Es: $\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_2 \vee x_3\right) \wedge \left(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}\right)$

Nota. Ogni formula logica può essere espressa in CNF

Soddisfacibilità

Letterale: una variabile Booleana o la sua negazione. x_i or $\overline{x_i}$

Clausola: un OR (o disgiunzione) di letterali. $C_j = x_1 \vee \overline{x_2} \vee x_3$

Forma Normale Congiuntiva (CNF): una formula proposizionale Φ che è un AND (o congiunzione) di clausole. $\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

Es: $\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_2 \vee x_3\right) \wedge \left(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}\right)$

SAT: **data formula CNF Φ, esiste un'assegnazione di verità che la soddisfa ?**

Ognuno corrispondente ad una variabile diversa

3-SAT: **SAT dove ogni clausola contiene esattamente 3 letterali.**

Si: x_1 = true, x_2 = true x_3 = false.

3-Soddisfacibilità si riduce a Independent set

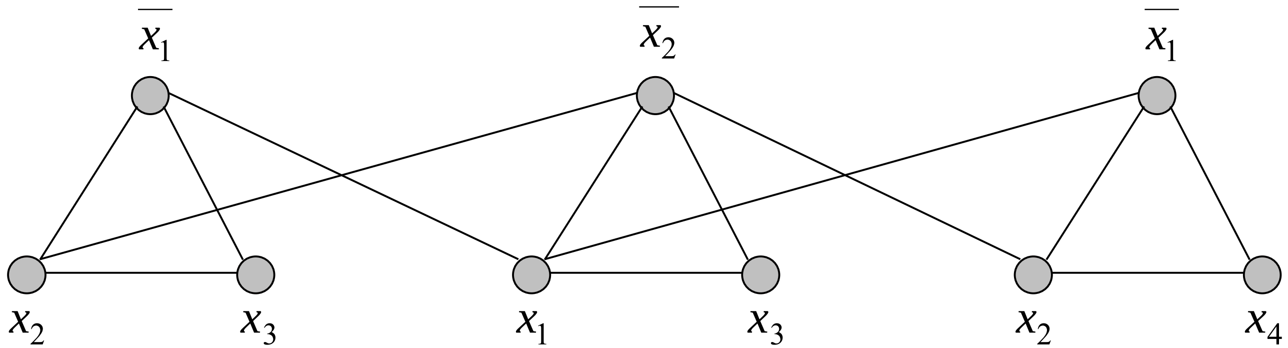
Fatto. 3-SAT $\leq_P$ INDEPENDENT-SET.

Dim. data istanza Φ di 3-SAT, costruiamo un' istanza (G, k) di INDEPENDENT-SET che ha un insieme indipendente di dimensione k sse Φ è soddisfacibile.

Costruzione.

- G contiene 3 vertici per ogni clausola, uno per ogni letterale.
- connettiamo i 3 letterali in un clausola in un triangolo.
- connettiamo ogni letterale a ogni suo negato.

G



k = 3

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

3-SATsi riduce a Independent-Set

Fatto. G contiene insieme indipendente di dimensione k = $|\Phi|$ sse Φ è soddisfacibile.

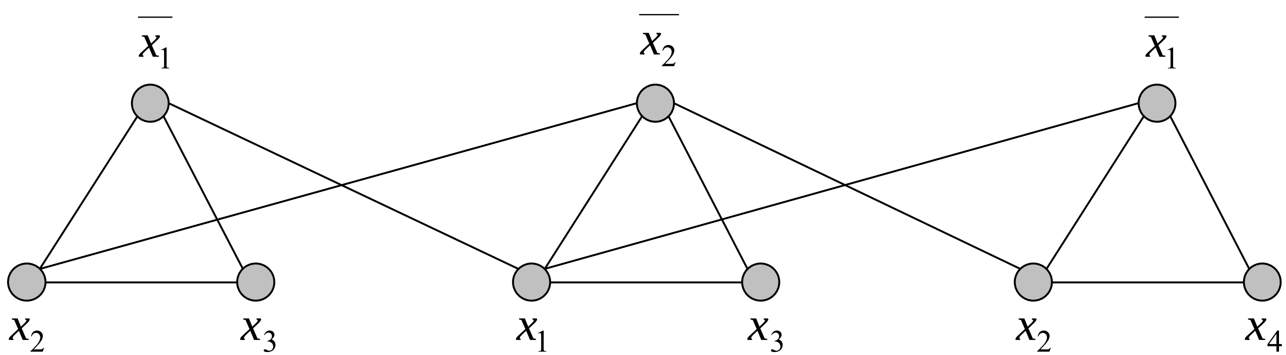
Dim.

$\Rightarrow$ Sia S l' insieme indipendente di dimensione k.

- ✎ S deve contenere esattamente un vertice in ogni triangolo.
- ✎ Non può contenere un letterale ed il suo negato

- Poniamo questi letterali a true.
- Assegnazione di verità è consistente e tutte le clausole sono soddisfatte.

G



k = 3

$$\Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$$

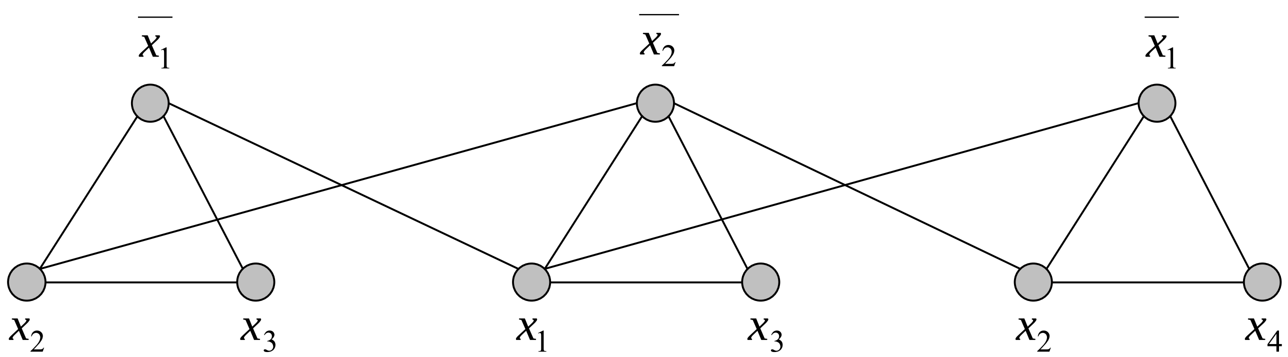
3 Soddisfacibilità si riduce a Independent set

Fatto. G contiene insieme indipendente di dimensione k = $|\Phi|$ sse Φ è soddisfacibile.

Dim.

$\Leftarrow$ data un'assegnazione che soddisfa,
selezioniamo un letterale true da ogni triangolo.
Questo è un insieme indipendente di dimensione k. ▪

G



k = 3

$$\Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$$

Riepilogo

Strategie viste.

- equivalenza semplice: INDEPENDENT-SET $\equiv_P$ VERTEX-COVER.
- caso speciale a caso generale: VERTEX-COVER $\leq_P$ SET-COVER.
- Codifica con gadgets: 3-SAT $\leq_P$ INDEPENDENT-SET.

transitività. If $X \leq_P Y$ e $Y \leq_P Z$, then $X \leq_P Z$.
Dim idea. Componiamo i due algoritmi.

Es: 3-SAT $\leq_P$ INDEPENDENT-SET $\leq_P$ VERTEX-COVER $\leq_P$ SET-COVER.

Self-Reducibility

Problema di decisione. Esiste un vertex-cover di dimensione $\leq$ k?
Problema di ricerca. Trovare vertex-cover di minima cardinalità.

Self-reducibility. Problema di ricerca $\leq_P$ versione di decisione.
- si applica a tutti i problemi (NP-completi) che vedremo.
- Giustifica la focalizzazione sui problemi di decisione.

Es: trovare vertex-cover di min cardinalità .
- Ricerca (binaria) per la cardinalità k^* di min vertex-cover.

- trovare un vertice v tale che $G - \{ v \}$ ha un vertex-cover di dimensione $\leq k^* - 1$.
  - un qualsiasi vertice in un qualsiasi min vertex-cover avrà tale proprietà
- Includiamo v nel vertex-cover. Cancelliamo v e tutti edge incidenti
- Ricorsivamente troviamo un min vertex-cover in $G - \{ v \}$.

P

P. Problemi di decisione per cui esiste a algoritmo polinomiale

problema	Description	algoritmo	Yes	No
MULTIPLO	è x multiplo di y?	Divisione	51, 17	51, 16
RELPRIMI	x e y relativamente primi?	Euclide (300 A.C)	34, 39	34, 51
PRIMO	è x primo?	AKS (2002)	53	51
EDIT-DISTANCE	La edit distance tra x e y minore di 5?	Programmazione dinamica	niether neither	acgggt ttttta
Sistema-eq	Esiste un vettore x che soddisfa Ax = b?	Eliminazione di Gauss-Edmonds	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

NP

Algoritmo di certificazione: idea.

- Il certificatore vede le cose da punto di vista "manageriale" .
- Il certificatore non determina se $s \in X$ (linguaggio associato al problema X); semplicemente, controlla una data dimostrazione t che $s \in X$.

Def. algoritmo $C(s, t)$ è un certificatore per problema X se per ogni stringa s, $s \in X$ sse esiste una stringa t tale che $C(s, t) =$ `yes`.

"certificato" o "witness"

NP. Problema di decisione per cui esiste un certificatore (avente tempo) polinomiale.

$C(s, t)$ è algoritmo polinomiale e
$|t| \leq p(|s|)$ per qualche polinomio $p(\cdot)$.

Nota NP sta per Nondeterministic Polynomial-time.

Certificatori e certificati : composito

COMPOSITES. Dato intero s, è s composito?

Certificato. Un fattore non banale t di s. Nota che tale certificato esiste sse s è composito. Inoltre $|t| \leq |s|$.

Il certificatore.

```
boolean C(s, t) {
   se (t ≤ 1 or t ≥ s)
      Restituisci  false
   else se (s è a multiplo di t)
      Restituisci  true
   else
      Restituisci  false
}
```

Istanza. s = 437,669.

Certificato. t = 541 o 809. ← 437,669 = 541 × 809

Conclusione. COMPOSITES è in NP.

Certificatori e certificati: 3-Satisfiability

SAT. Data una CNF formula Φ, esiste un' assegnazione che la soddisfa?

Certificato. un assegnamento di valori di verità alle n variabili booleane.

Certificatore. Controlla che ogni clausola in Φ ha almeno un letterale true.

Es.

$$\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_1 \vee x_2 \vee x_4\right) \wedge \left(\overline{x_1} \vee \overline{x_3} \vee \overline{x_4}\right)$$

istanza s

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$$

certificato t

Conclusione. SAT è in NP.

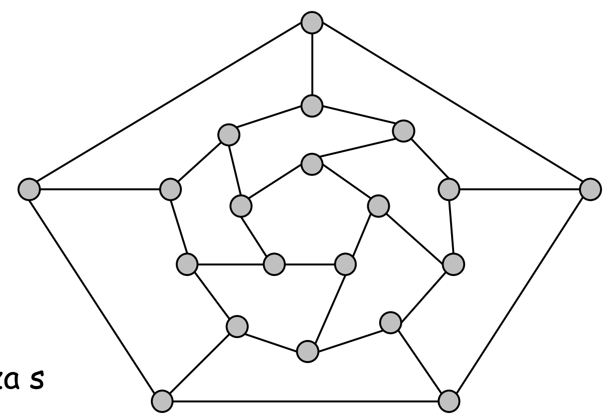
Certificatori e certificati: Hamiltonian ciclo

HAM-CYCLE. Dato un grafo non orientato $G = (V, E)$, esiste un ciclo semplice C che visita ogni nodo ?

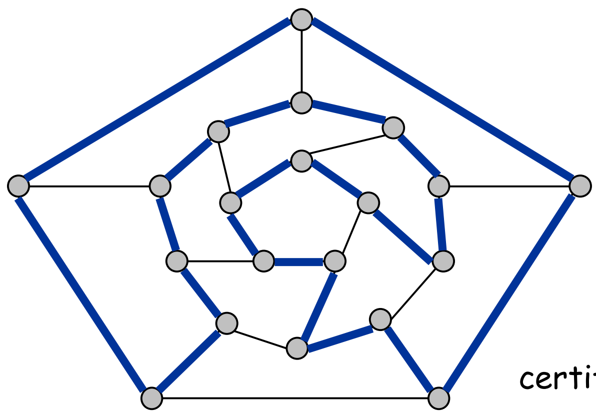
Certificato. Una permutazione di n nodi.

Il certificatore. Controlliamo che la permutazione contiene ogni nodo in V esattamente una volta, e che esiste un edge tra ogni coppia di nodi adiacenti nella permutazione.

Conclusione. HAM-CYCLE è in NP.



istanza s



certificato t

P, NP, EXP

P. Problema didecisione per cui esiste un algoritmo polinomiale.
EXP. Problema didecisione per cui esiste un algoritmo esponenziale.
NP. Problema didecisione per cui esiste certificatore polinomiale .

Fatto. P $\subseteq$ NP.
Dim. Consideriamo un qualsiasi problema X in P.

- Per Definizione, esiste algoritmo polinomiale A(s) che risolve X.
- Certificato: $t = \varepsilon$, certificatore $C(s, t) = A(s)$. ▪

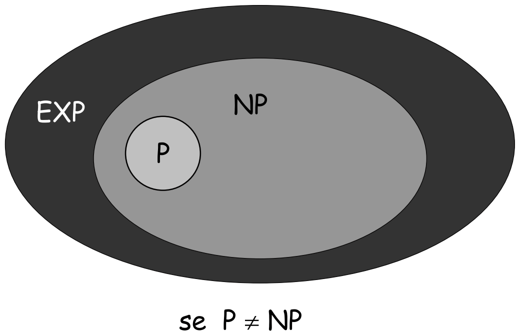
Fatto. NP $\subseteq$ EXP.
Dim. Consideriamo un qualsiasi problema X in NP.

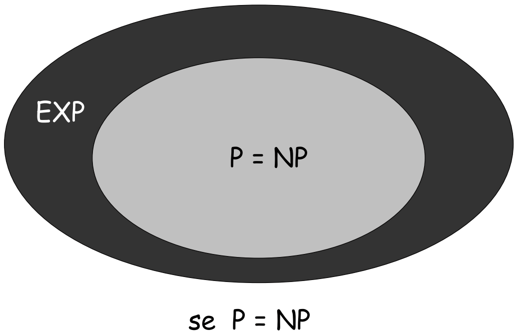
- Per definizione, esiste certificatore polinomiale C(s, t) per X.
- Per risolvere su input s, usiamo C(s, t) su tutte le stringhe t con $|t| \le p(|s|)$.
- Restituisci `yes`, se C(s, t) restituisce `yes` per una qualsiasi stringa. ▪

Domanda Aperta: P = NP?

P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

- Un problema di decisione è facile quanto il problema di certificazione?





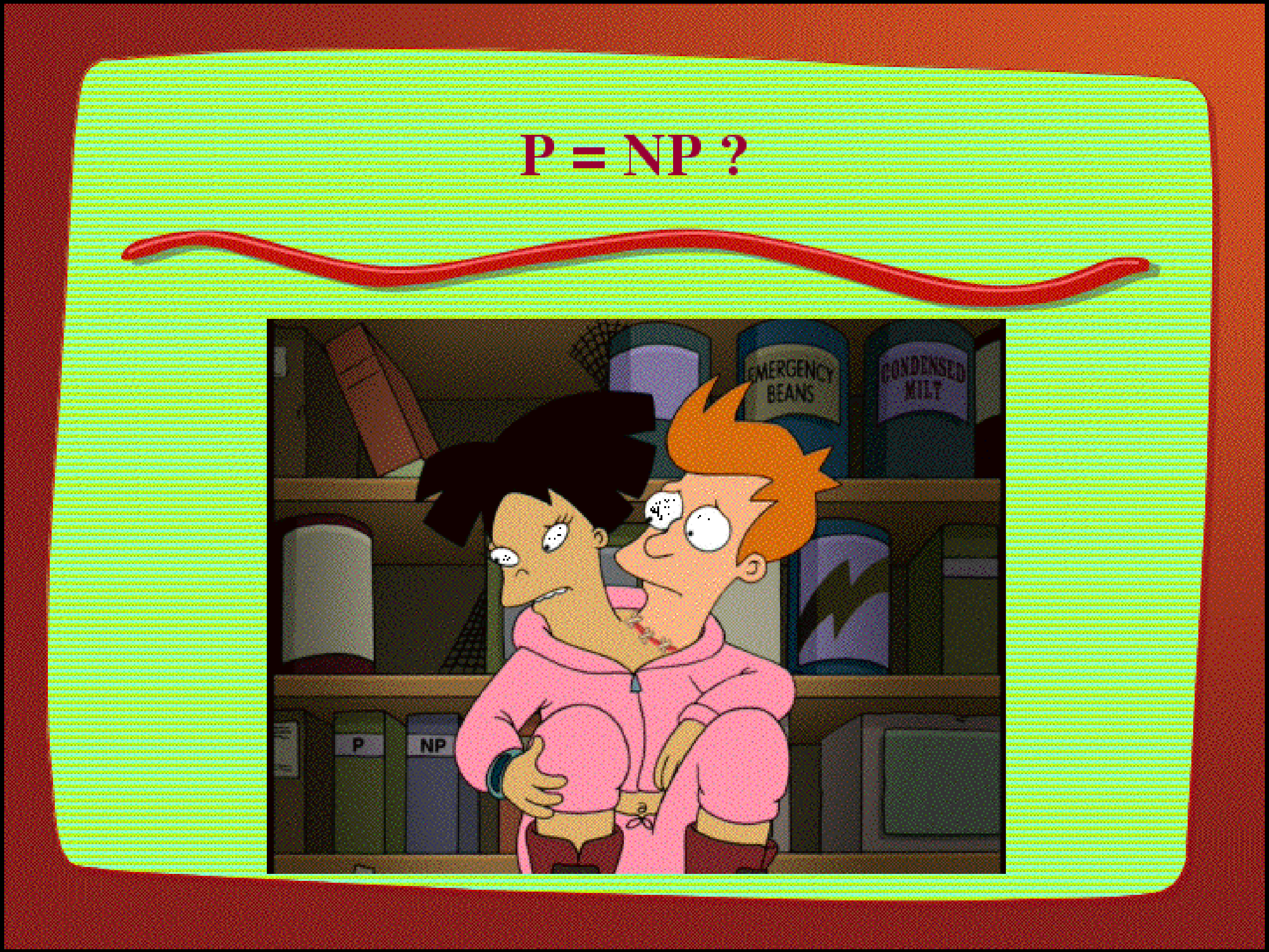
Romperebbe sistema crittografico RSA

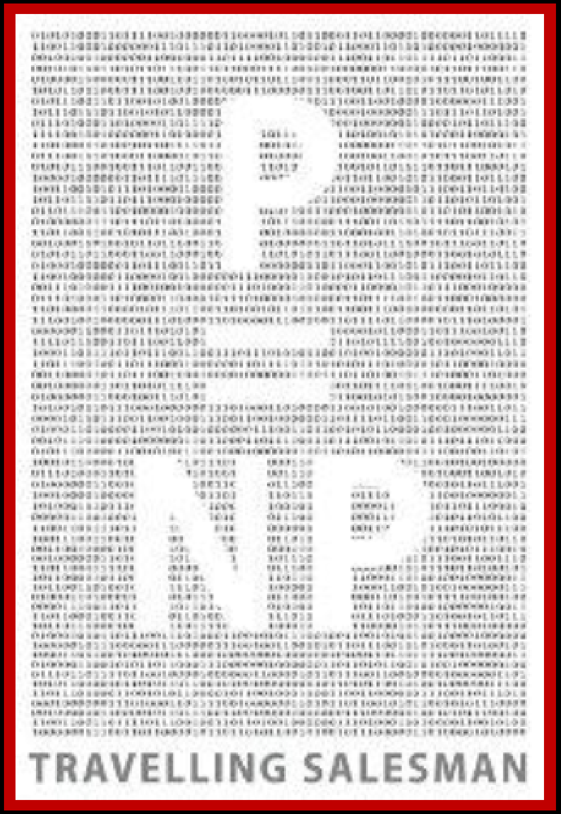
se si: algoritmi efficienti per 3-COLOR, TSP, FACTOR, SAT, …
se no: No algoritmi efficienti possibli per 3-COLOR, TSP, SAT, …

Opinione generale su P = NP? **Probabilmente no.**

The Simpson's: P = NP?

Futurama: P = NP?





FILM

Directed by	Timothy Lanzone
Produced by	Preston Clay Reed
Screenplay by	Andrew Lanzone Timothy Lanzone
Starring	Matt Lagan Steve West Danny Barclay Marc Raymond Tyler Seiple David John Cole Malek Houlihan Eric Bloom
Music by	Benjamin Krause
Studio	Fretboard Pictures
Release date(s)	•June 16, 2012
Country	United States
Language	English

ELEMENTARY

La serie è una rilettura in chiave moderna del celebre Sherlock Holmes di Sir Arthur Conan Doyle

Dopo essere stato per anni consulente per Scotland Yard e dopo essere uscito da una clinica per disintossicarsi dall'alcool e dalle droghe, Sherlock Holmes si stabilisce a New York City dove accetta di collaborare con il New York City Police Department e risolvere diversi casi con l'aiuto della logica e del suo intuito, affiancato, a suo malgrado, dall'ex chirurgo Joan Watson.

SOLVE FOR X

The murder of a mathematician reveals a deadly race to solve the great mystery of
P vs. NP

8.4 NP-Completezza

Trasformazioni polinomiali

Def. Il problema X si riduce in modo polinomiale (Cook) al problema Y se istanze arbitrarie del problema X possono essere risolte usando:

- un numero polinomiale di passi di computazione standard, più
- un numero polinomiale di chiamate ad oracolo che risolve problema Y.

Def. Il problema X si trasforma in modo polinomiale (Karp) al problema Y se dato un qualsiasi input x di X, possiamo costruira un input y tale che x è istanza `yes` di X sse y è a istanza `yes` di Y.

Richiediamo |y| di dimensione polinomioale in |x|

Nota. Trasformazioni polinomiali sono riduzioni polinomiali con una sola chiamata all'oracolo per Y, esattamente alla fine dell'algoritmo per X. Quasi tutte le precedenti riduzioni erano di questa forma.

Domanda (irrisolta). Queste due concetti sono equivalenti rispetto a NP?

Noi usiamo la stessa notazione $\leq_p$

NP-Completo: problema Y in NP con la proprietà che per ogni problema X in NP, $X \leq_p Y$.

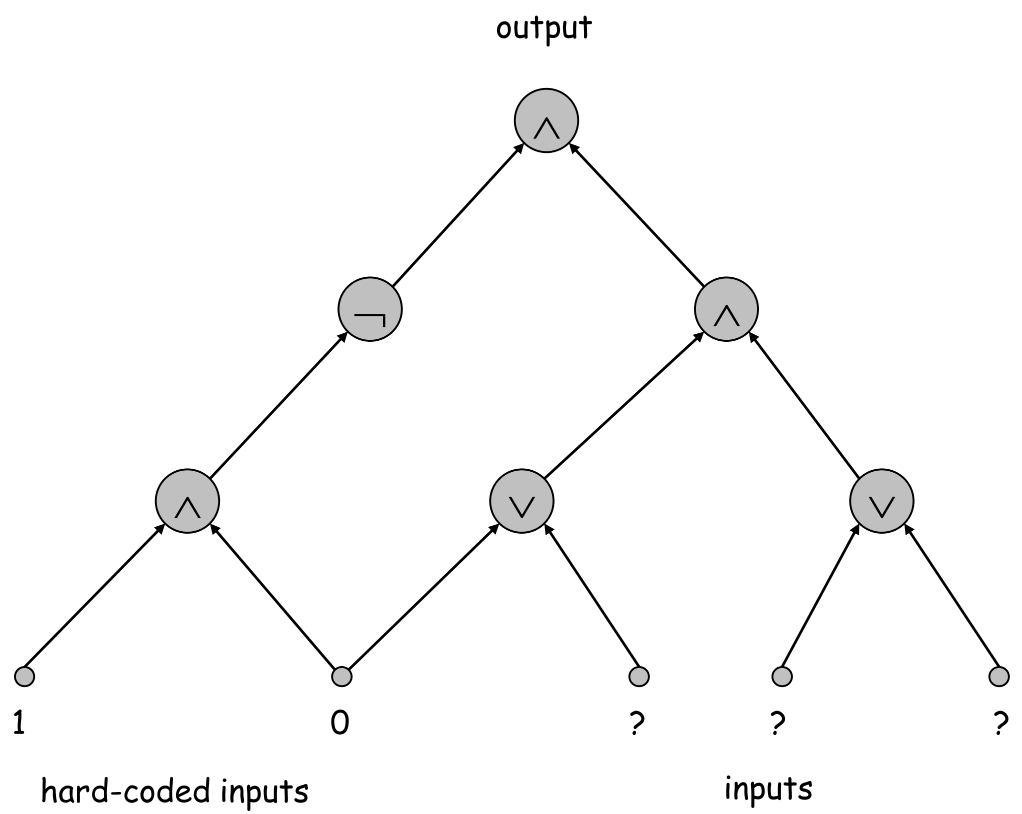
Teorema. Assumiamo che Y è un problema NP-Completo. Allora Y è risolvibile in tempo polinomiale sse P = NP.
Dim. $\Leftarrow$ se P = NP allora Y può essere risolto in tempo polinomiale poichè Y è in NP.

Dim. $\Rightarrow$ Assumiamo Y può essere risolto in tempo polinomiale.
- Sia X un qualsiasi problema in NP. Poichè $X \leq_p Y$, possiamo risolvere X in tempo polinomiale. Ciò implica NP $\subseteq$ P.
- Gia sappiamo P $\subseteq$ NP. Quindi P = NP. ▪

circuito Satisfiability

CIRCUIT-SAT. Dato a circuito combinatoriale composto da porte AND, OR, e NOT, c'e un modo di settare gli input del circuito in modo che l'output sia 1?

yes: 1 0 1



Il primo problema NP-Completo

Teorema. CIRCUIT-SAT è NP-Completo . [Cook 1971, Levin 1973]

Dim. (idea)

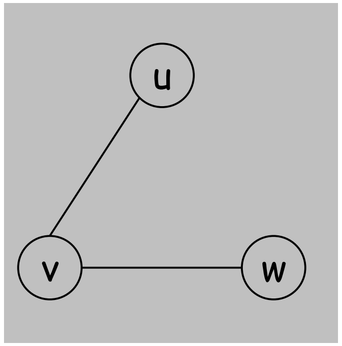
- un qualsiasi algoritmo che prende in input un numero fissato n di bits e produce risposta yes/no può esserere rappresentato con tale circuito
- inoltre, se l'algoritmo prende tempo polinomiale, allora il circuito è di dimensione polinomiale

Fissare il numero di bit è importante; differenza base tra algoritmo e circuito

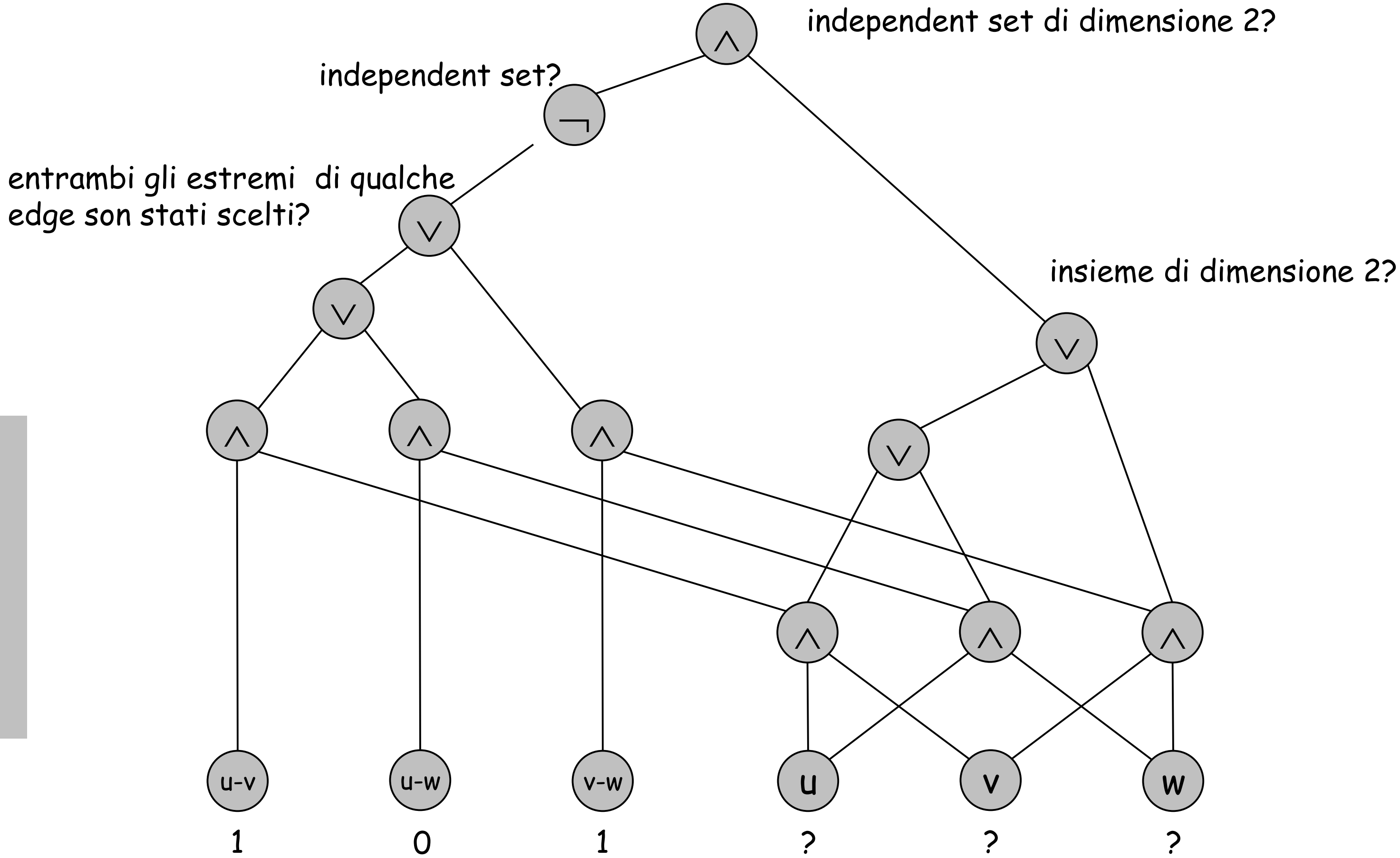
- Consideriamo un problema X in NP. Ha certificatore polinomiale C(s, t). Per determinare se s è in X, dobbiamo sapere se esiste un certificato t di lunghezza p(|s|) tale che C(s, t) = `yes`.
- Consideriamo C(s, t) come un algoritmo su |s| + p(|s|) bits (input s, certificato t) e convertiamolo in circuito di dimensione polinomiale K.
  - primi |s| bits sono hard-coded con s
  - restanti p(|s|) bits rappresentano i bits di t
- circuito K è soddisfacibile sse C(s, t) = `yes`.

Esempio

Es. La costruzione crea un circuito K i cui inputs possono essere settati in modo che l' output è true sse il grafo G ha un independent set di dimensione 2.o



G = (V, E), n = 3



$\binom{n}{2}$ hard-coded inputs (descrizione grafo) n inputs (nodi in independent set)

Stabilire NP-Completezza

Nota una volta che troviamo il primo problema NP-Completo, gli altri seguono.

Passi per stabilire NP-Completezza di problema Y.

- passo 1. Mostrare che Y è in NP.
- passo 2. Scegliere un problema NP-Completo X.
- passo 3. Provare che $X \leq_p Y$.

Giustificazione. Se X è un problema NP-Completo e Y è un problema in NP tale che $X \leq_P Y$, allora Y è NP-Completo.

Dim. Sia W un qualsiasi problema in NP. Allora $\underbrace{W \leq_P X}_{\text{Per definizione di NP-Completo}} \; \underbrace{\leq_P Y}_{\text{Per ipotesi}}$.

- Per la transitività, $W \leq_P Y$.
- Quindi Y è NP-Completo. ▪

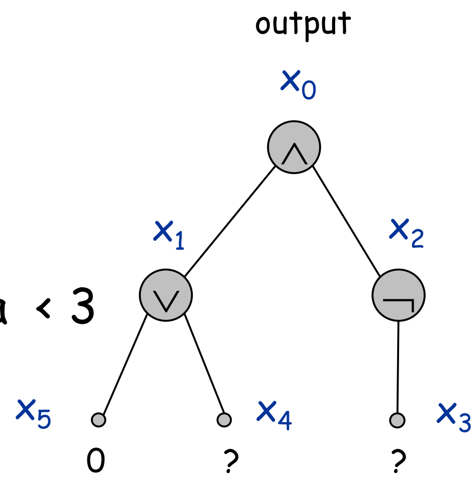
3-SAT è NP-Completo

Teorema. 3-SAT è NP-Completo .

Dim. Basta mostrare che CIRCUIT-SAT $\leq_P$ 3-SAT poichè 3-SAT è in NP.

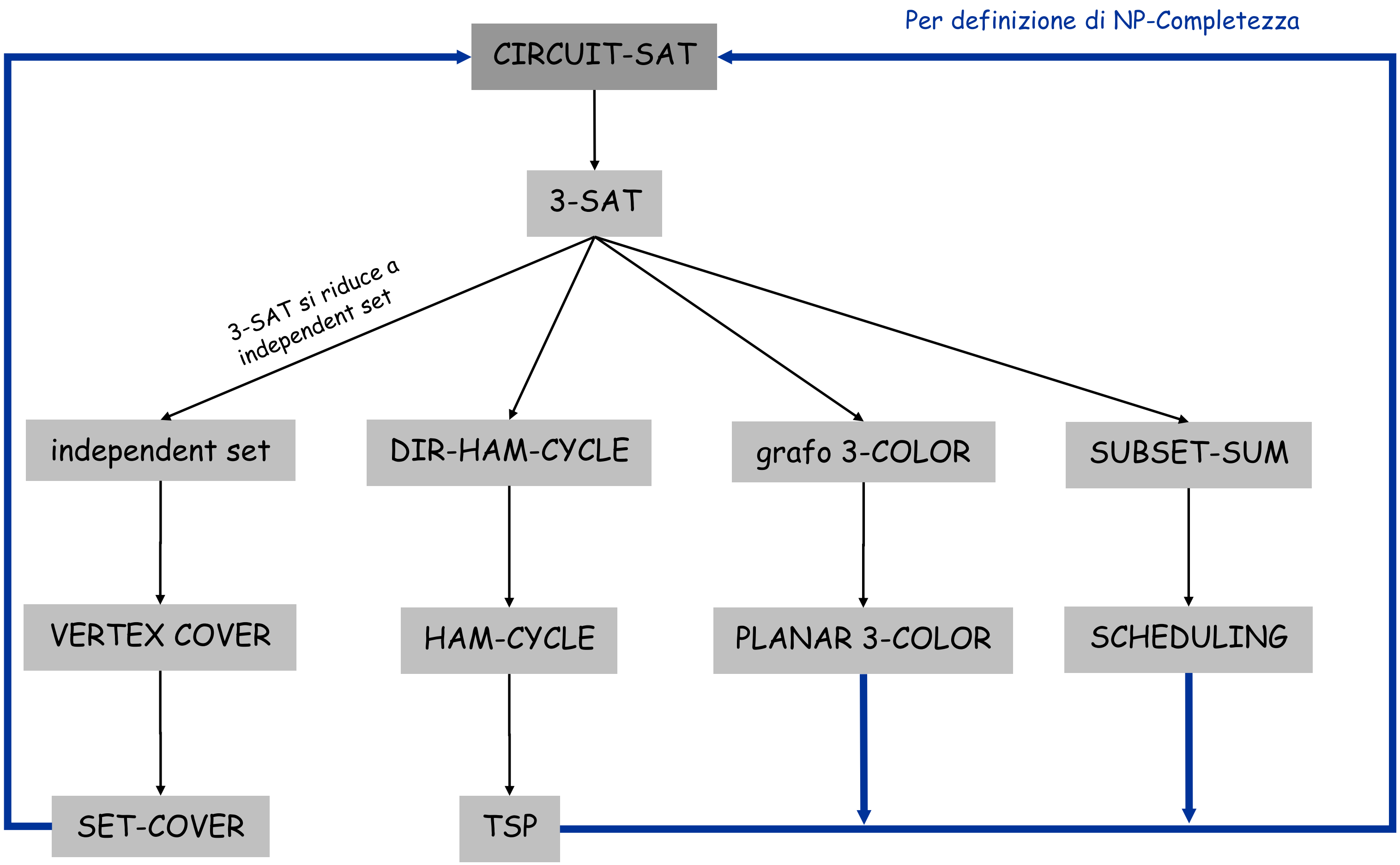
- Sia K un qualsiasi circuito .
- Creiamo una variabile 3-SAT x_i per ogni elemento i del circuito.
- Facciamo computare al circuito i valori corretti ad ogni nodo :
  - $x_2 = \neg x_3$ $\Rightarrow$ aggiungiamo 2 clausole: $x_2 \vee x_3$, $\overline{x_2} \vee \overline{x_3}$
  - $x_1 = x_4 \vee x_5$ $\Rightarrow$ aggiungiamo 3 clausole: $x_1 \vee \overline{x_4}$, $x_1 \vee \overline{x_5}$, $\overline{x_1} \vee x_4 \vee x_5$
  - $x_0 = x_1 \wedge x_2$ $\Rightarrow$ aggiungiamo 3 clausole: $\overline{x_0} \vee x_1$, $\overline{x_0} \vee x_2$, $x_0 \vee \overline{x_1} \vee \overline{x_2}$

- Hard-coded input e output.
  - $x_5 = 0$ $\Rightarrow$ aggiungiamo 1 clausola: $\overline{x_5}$
  - $x_0 = 1$ $\Rightarrow$ aggiungiamo 1 clausola: x_0



- Passo finale: trasformiamo clausole di lunghezza < 3 into clausole di lunghezza esattamente 3.
  (servono 4 nuove variabili) ▪

Osservazione. Tutti i problemi sono NP-Completi e ammettono riduzione polinomiale da uno all'altro!



Alcuni problemi NP-Completi

Tipologie base ed esempi paradigmatici:

- Problemi di Packing: SET-PACKING, independent set.
- Problemi di Covering : SET-COVER, VERTEX-COVER.
- Problemi di Soddisfacibilità con vincoli: SAT, 3-SAT.
- Problemi di sequenzazione: HAMILTONIAN-CYCLE, TSP.
- Problemi di Partizionamento: 3D-MATCHING 3-COLOR.
- Problemi Numerici : SUBSET-SUM, KNAPSACK.

Pratica. La maggior parte dei problemi NP sono noti essere in P oppure essere NP-Completi

Eccezioni notevoli. Fattorizzazione, isomorfismo di grafi.

Altri problemi computazionalmente difficili

Biologia: folding delle proteine
(processo di ripiegamento molecolare attraverso il quale le proteine ottengono la loro struttura tridimensionale; problema: predire la struttura tridimensionale di una proteina partendo dalla sequenza amminoacidica)

Ingegneria chimica: reti di scambio termico

Ingegneria civile: equilibrio del flusso del traffico urbano.

Economia: calcolo di arbitrati nei mercati finanziari

Ingegneria elettrica: layout VLSI.

Ingegneria ambientale: disposizione ottima di sensori

Ingegneria finanziaria: trova il portfolio a minimo rischio di dato …

Teoria dei giochi: trova l'equilibrio di Nash che massimizza il welfare sociale

8.9 co-NP

Asimmetria di NP

Es 1. **SAT**

Possiamo provare che una formula CNF è soddisfacibile dando un assegnazione.

- Come possiamo provare che una formula non è soddisfacibile ?

Es 2. **HAM-CYCLE**

Possiamo provare che un grafo è Hamiltoniano dando il circuito Hamiltoniano

- Come possiamo provare che un grafo non è Hamiltoniano?

NP e co-NP

NP. Problema di decisione per cui esiste certificatore polinomiale
Es. SAT, HAM-CYCLE.

Def. Dato problema di decisione X, il complemento $\overline{X}$ è lo stesso problema con risposte `yes` e `no` invertite.

co-NP. Complemento di un problema di decisione in NP.

Es. TAUTOLOGY (complemento di SAT),
NO-HAM-CYCLE (complemento di HAM-CYCLE),

Domanda fondamentale. NP = co-NP?

- istanze `yes` hanno certificati succinti sse li hanno le istanze `no`?
- Opinione comune: no.

Teorema. se NP $\neq$ co-NP, allora P $\neq$ NP.

Dim (idea).
Mostriamo che se P = NP, allora NP = co-NP.

- P è chiusa per il complemento.
- se P = NP, allora NP è chiusa per il complemento
- cioè NP = co-NP.

Una buona caratterizzazione di NP $\cap$ co-NP.

[Edmonds 1965]

se problema X è sia in NP e co-NP, Allora :

- per istanza `yes` esiste un certificato succinto
- per istanza `no` esiste "disqualifier" succinto

Es. Dato a grafo bipartito, esso ammette un perfect matching.

- se yes, possiamo fornire un perfect matching.
- se no, possiamo fornire un insieme di nodi S tale che $|N(S)| < |S|$

Nota: un grafo ha un perfect matching sse per ogni
S risulta $|N(S)| \geq |S|$

.

Osservazione. $P \subseteq (NP \cap \text{co-}NP)$.

Problema aperto: $P = NP \cap \text{co-}NP$?

- opinioni diverse

PRIMES è in NP $\cap$ co-NP

Teorema. PRIMES è in NP $\cap$ co-NP.
Dim. Gia sappiamo che PRIMES è in co-NP. Basta provare che PRIMES è in NP.

Teorema. Un intero dispari s è primo sse esiste intero $1 < t < s$ t.c.

$$\begin{aligned} t^{s-1} &\equiv 1 \pmod{s} \\ t^{(s-1)/p} &\not\equiv 1 \pmod{s} \end{aligned}$$

per tutti i divisori primi p di s-1

Input. s = 437,677
Certificato. t = 17, $2^2 \times 3 \times 36{,}473$

Fattorizzazione di s-1:
richiede certificati per asserire che
3 e 36,473 sono primi

Il certificatore.
- Controlliamo s-1 = $2 \times 2 \times 3 \times 36{,}473$.
- Controlliamo $17^{s-1} = 1 \pmod{s}$.
- Controlliamo $17^{(s-1)/2} \equiv 437{,}676 \pmod{s}$.
- Controlliamo $17^{(s-1)/3} \equiv 329{,}415 \pmod{s}$.
- Controlliamo $17^{(s-1)/36{,}473} \equiv 305{,}452 \pmod{s}$.

FACTOR è in NP $\cap$ co-NP

FACTORIZE. Dato intero x, trova fattorizzazione.

FACTOR. Dati due interi x e y, x ha un fattore minore di y?

Teorema. FACTOR è in NP $\cap$ co-NP.

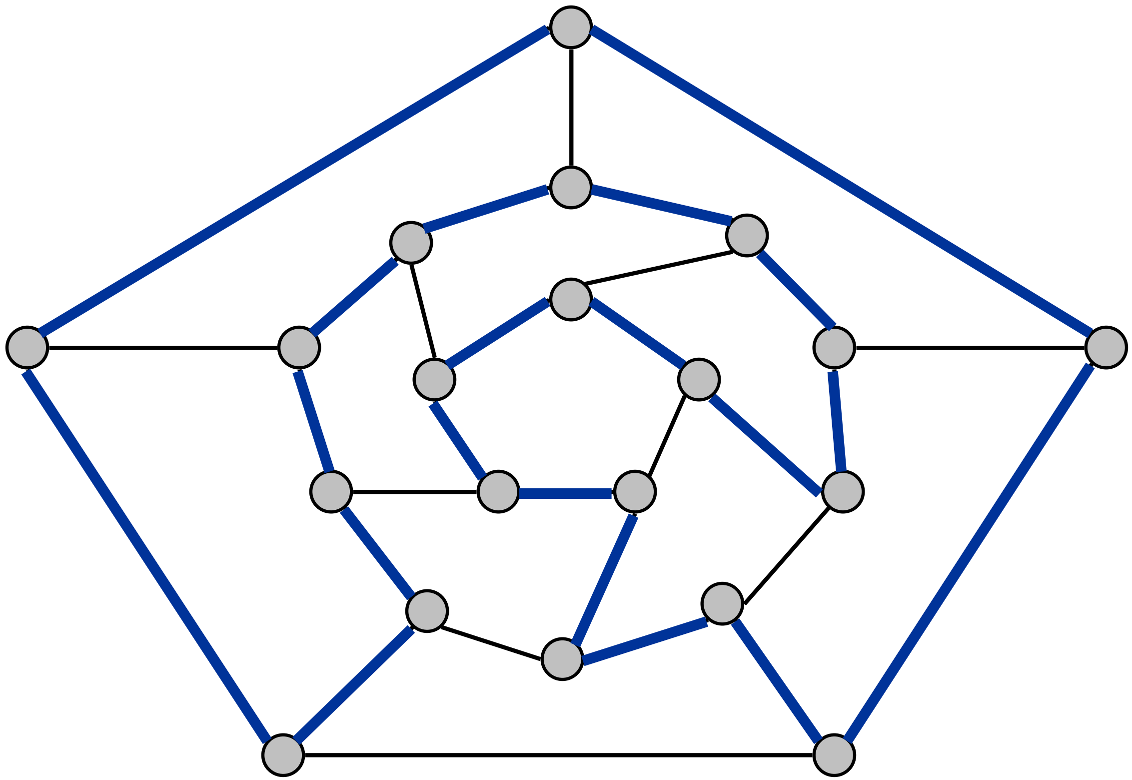
Dim.

- Certificato: un fattore p di x minore di y.
- Disqualifier: fattorizzazione di x (senza fattore minore di y), e certificato che ogni fattore è primo

8.5 Problemi di Sequencing

Circuito Hamiltoniano

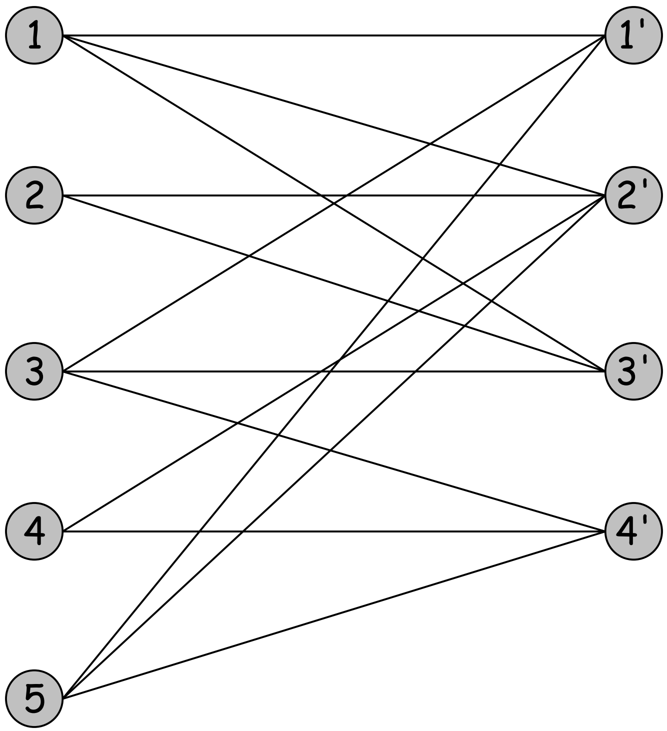
HAM-CYCLE: dato un grafo non orientato $G = (V, E)$, esiste un ciclo semplice Γ che contiene ogni nodo in V.



YES

Circuito Hamiltoniano

HAM-CYCLE: dato un grafo non orientato $G = (V, E)$, esiste un ciclo semplice Γ che contiene ogni nodo in V.



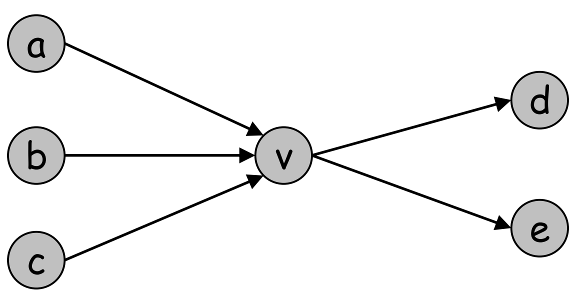
NO: grafo bipartito con numero dispari di nodi.

Circuito Hamiltoniano Orientato

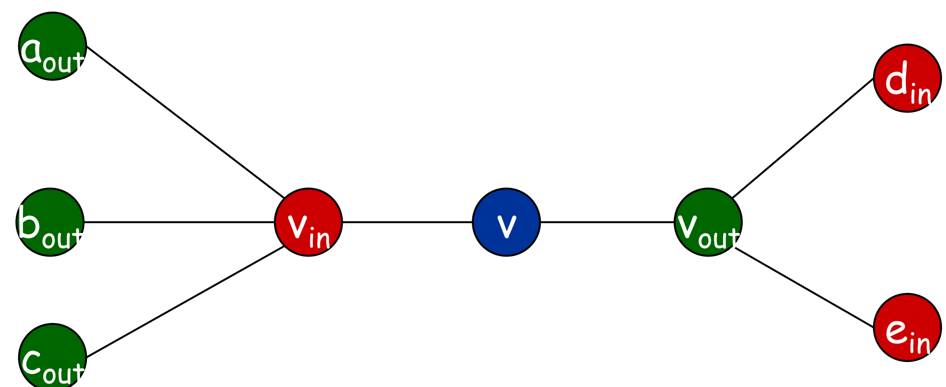
DIR-HAM-CYCLE: dato digrafo $G = (V, E)$, esiste una ciclo orientato semplice Γ che contiene ogni nodo in V?

Fatto. DIR-HAM-CYCLE $\leq_P$ HAM-CYCLE.

Dim. Dato un grafo orientato $G = (V, E)$, costruiamo un grafo non orientato G' con 3n nodi.



G



G'

Circuito Hamiltoniano Orientato

Fatto. G ha ciclo Hamiltoniano sse G' lo ha

Dim. $\Rightarrow$

- Assumiamo G ha a ciclo Hamiltoniano orientato Γ.
- Allora G' ha un ciclo Hamiltoniano non orientato (stesso ordine).

Dim. $\Leftarrow$

- Assumiamo G' ha un ciclo Hamiltoniano non orientato Γ'.
- Γ' deve visitare nodi in G' usando uno dei due ordini :
  - …, B, G, R, B, G, R, B, G, R, B, …
  - …, B, R, G, B, R, G, B, R, G, B, …
- nodi blue in Γ' formano ciclo Hamiltoniano orientato Γ in G, o inverso di un ciclo Hamiltoniano orientato. ▪

3-SAT si riduce a Circuito Hamiltoniano Orientato

Fatto. 3-SAT $\leq_P$ DIR-HAM-CYCLE.

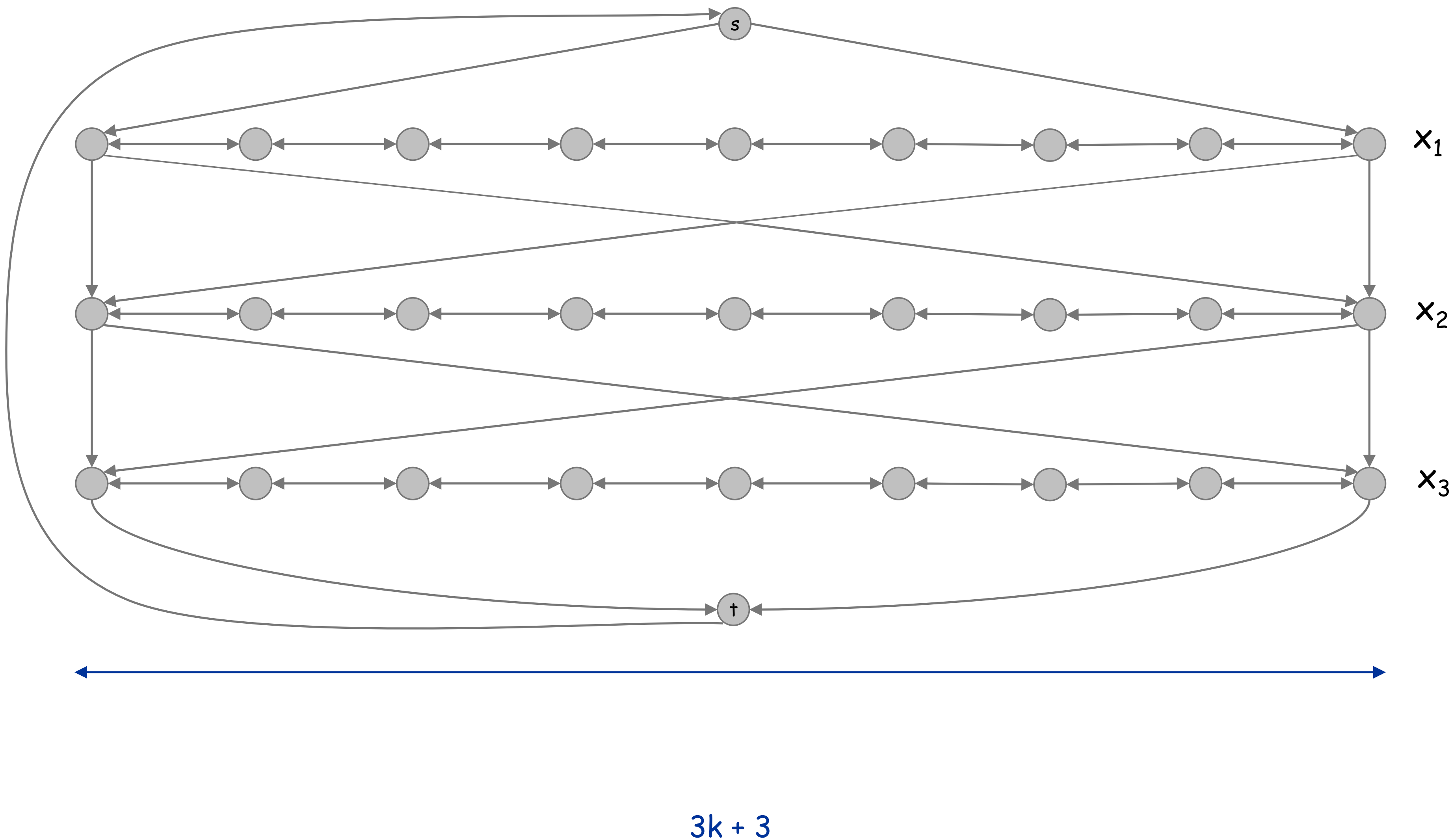
Dim. Data istanza Φ di 3-SAT, costruiamo istanza di DIR-HAM-CYCLE che ammette un circuito Hamiltoniano sse Φ è soddisfacibile .

Costruzione . Creiamo grafo che ha 2^n cicli Hamiltoniani che corrispondono in modo naturale alle 2^n possibili assegnazioni di verità per Φ

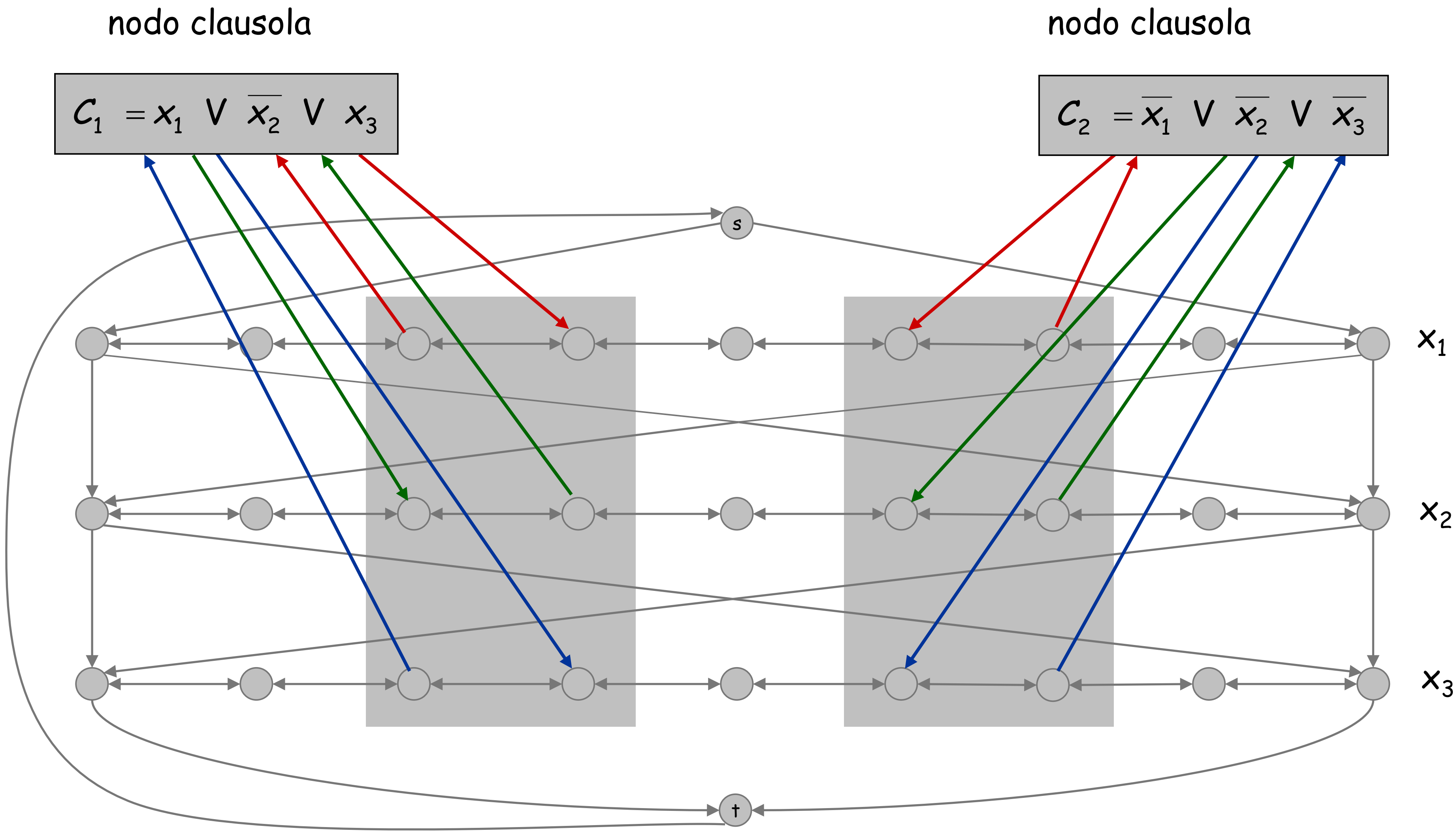
Dato istanza Φ di 3-SAT con n variabili x_i $(i=1,\ldots,n)$ e k clausole.

- Creiamo grafo G con 2^n cicli Hamiltoniani che corrispondono alle 2^n possibili assegnazioni di verità per Φ

Idea: attraversa path i da sinistra a destra $\Leftrightarrow$ porre variabile $x_i = 1$.

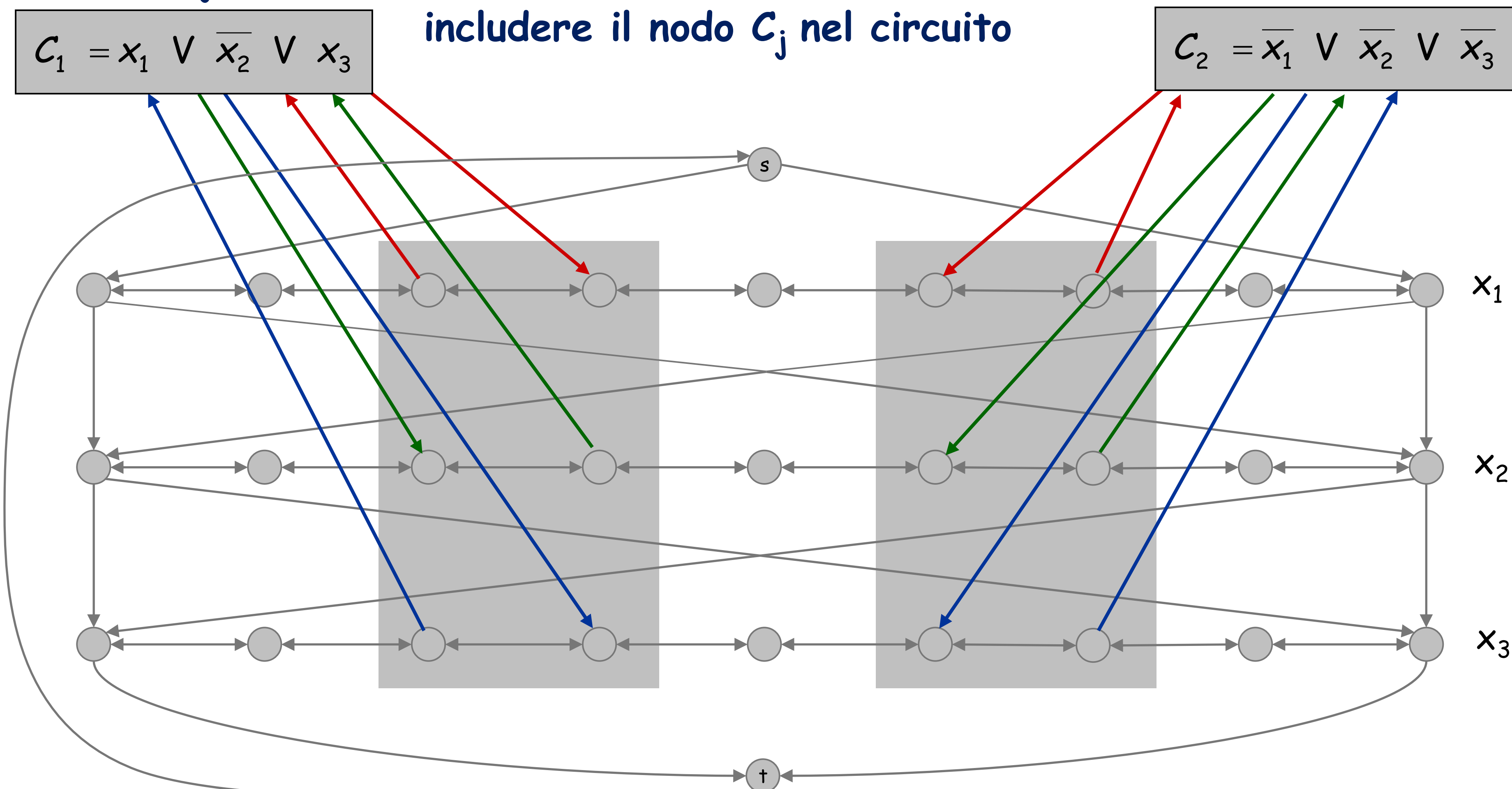


- per ogni clausola: aggiungiamo 1 nodo e 6 edges.



Fatto. Φ è soddisfacibile sse G ha a ciclo Hamiltoniano.

Dim. $\Rightarrow$ Assumiamo istanza 3-SAT ha assegnazione che la soddisfa, sia x^*.
Costruiamo il circuito Hamiltoniano in G.
direzione attraversam. riga i: sinistra- destra se $x^*_i = 1$, destra-sinistra se $x^*_i = 0$,
per ogni C_j, esiste riga i che attraversiamo nella direzione "corretta" per includere il nodo C_j nel circuito



Dim. $\Leftarrow$ **Assumiamo G ha ciclo Hamiltoniano Γ.**

se Γ entra nel nodo C_j, deve uscirne su edge corrispondente.

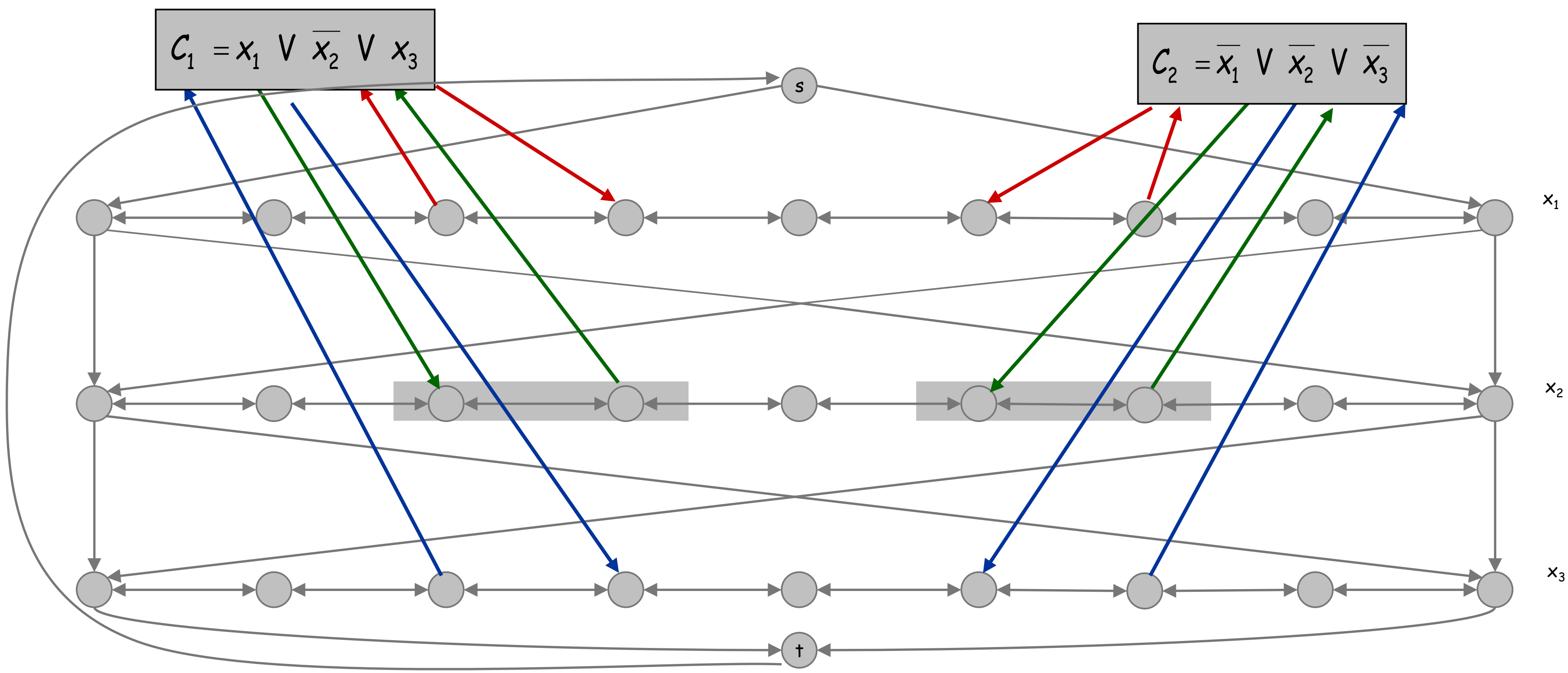
Quindi, nodi immediatamente prima e dopo C_j sono connessi da un edge in G

Rimuoviamo C_j da ciclo sostituendolo con edge, abbiamo ciclo Ham. su $G - \{ C_j \}$

Iterando, rimaniamo con con ciclo Ham. Γ' in $G - \{ C_1, C_2, \ldots, C_k \}$.

Poniamo $x^*_i = 1$ sse Γ' attraversa riga i da sinistra a destra:

poichè Γ visita ogni C_j, almeno una delle path è attraversata nel verso "giusto", e ogni clausola è soddisfatta. ▪



Longest Path

SHORTEST-PATH. Dato a digrafo $G = (V, E)$, esiste una path semplice di lunghezza al più k edges?

LONGEST-PATH. Dato a digrafo $G = (V, E)$, esiste path semplice di lunghezza almeno k edges?

Fatto. 3-SAT $\leq_P$ LONGEST-PATH.

Esercizio 1. Ripetere dimostrazione per DIR-HAM-CYCLE (ignorando il back-edge da t a s)

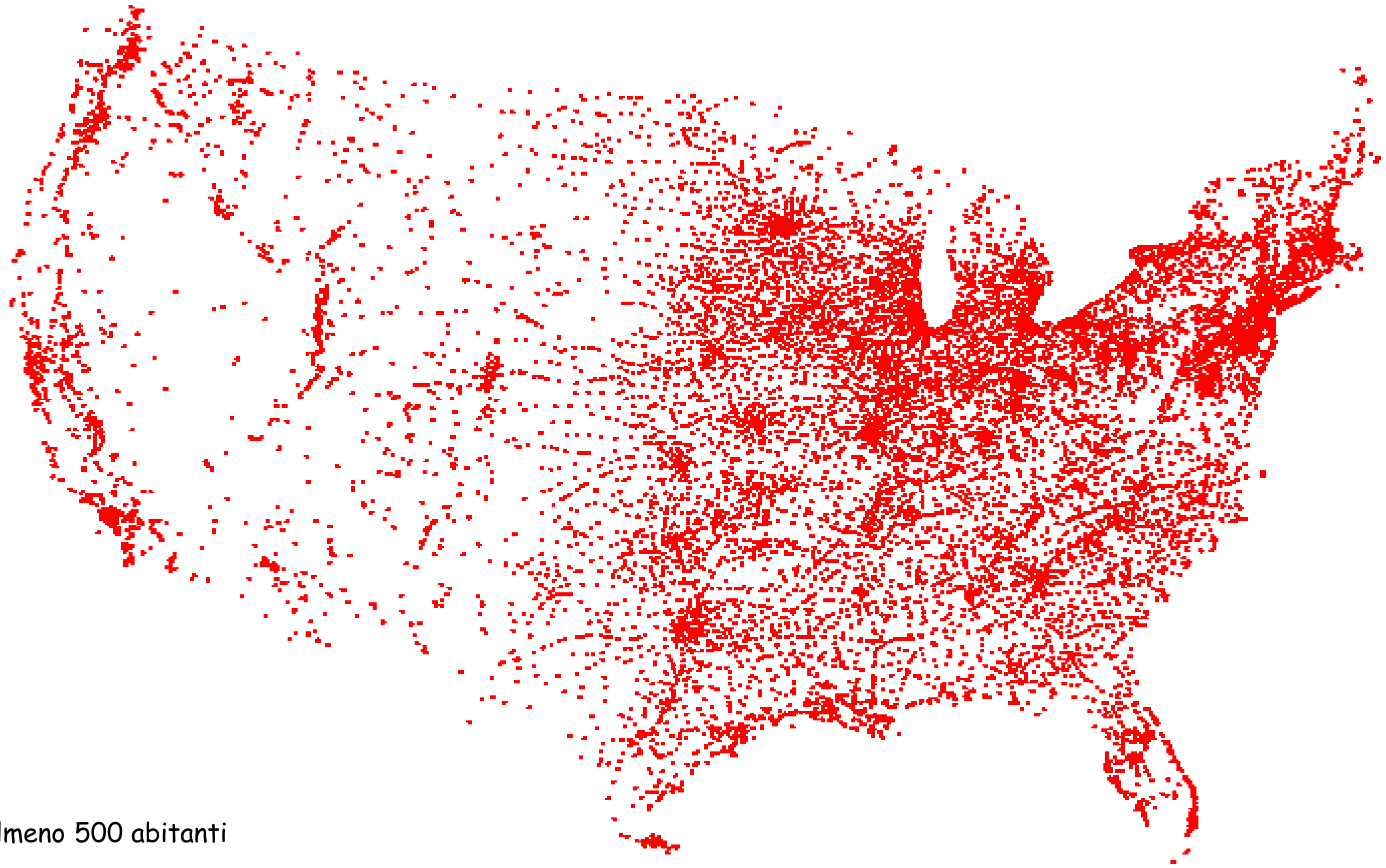
Esercizio 2. Mostrare HAM-CYCLE $\leq_P$ LONGEST-PATH.

Traveling Salesperson

TSP (problema del Commesso Viaggiatore).

Dato un insieme di n città e distanze d(u, v), esiste un **tour** di lunghezza $\leq$ D?

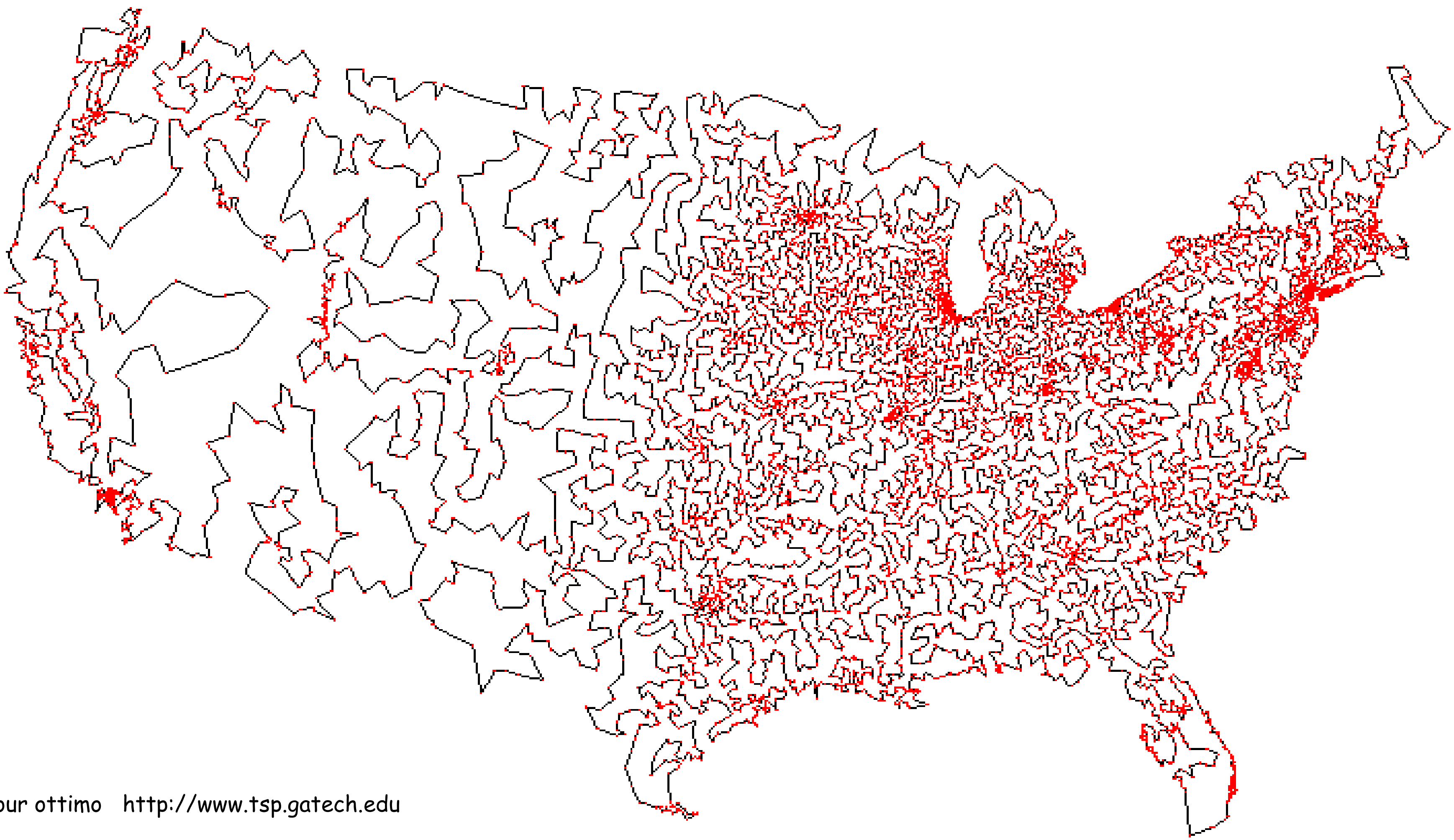
(circuito che tocca tutte le città **almeno** una volta)



tutte le 13,509 città in USA con almeno 500 abitanti
http://www.tsp.gatech.edu

Traveling Salesperson

TSP. Dato un insieme di n città e una funzione distanza d(u, v), esiste un tour di lunghezza $\leq$ D?



Tour ottimo   http://www.tsp.gatech.edu

Traveling Salesperson problema

TSP. Dato a insieme di n città e una funzione distanza d(u, v), esiste un tour di lunghezza $\leq$ D?

HAM-CYCLE: dato a grafo G = (V, E), esiste un ciclo semplice che contiene ogni nodo in V?

Fatto. HAM-CYCLE $\leq_P$ TSP.

Dim.

- Data istanza G = (V, E) di HAM-CYCLE, creiamo n città con

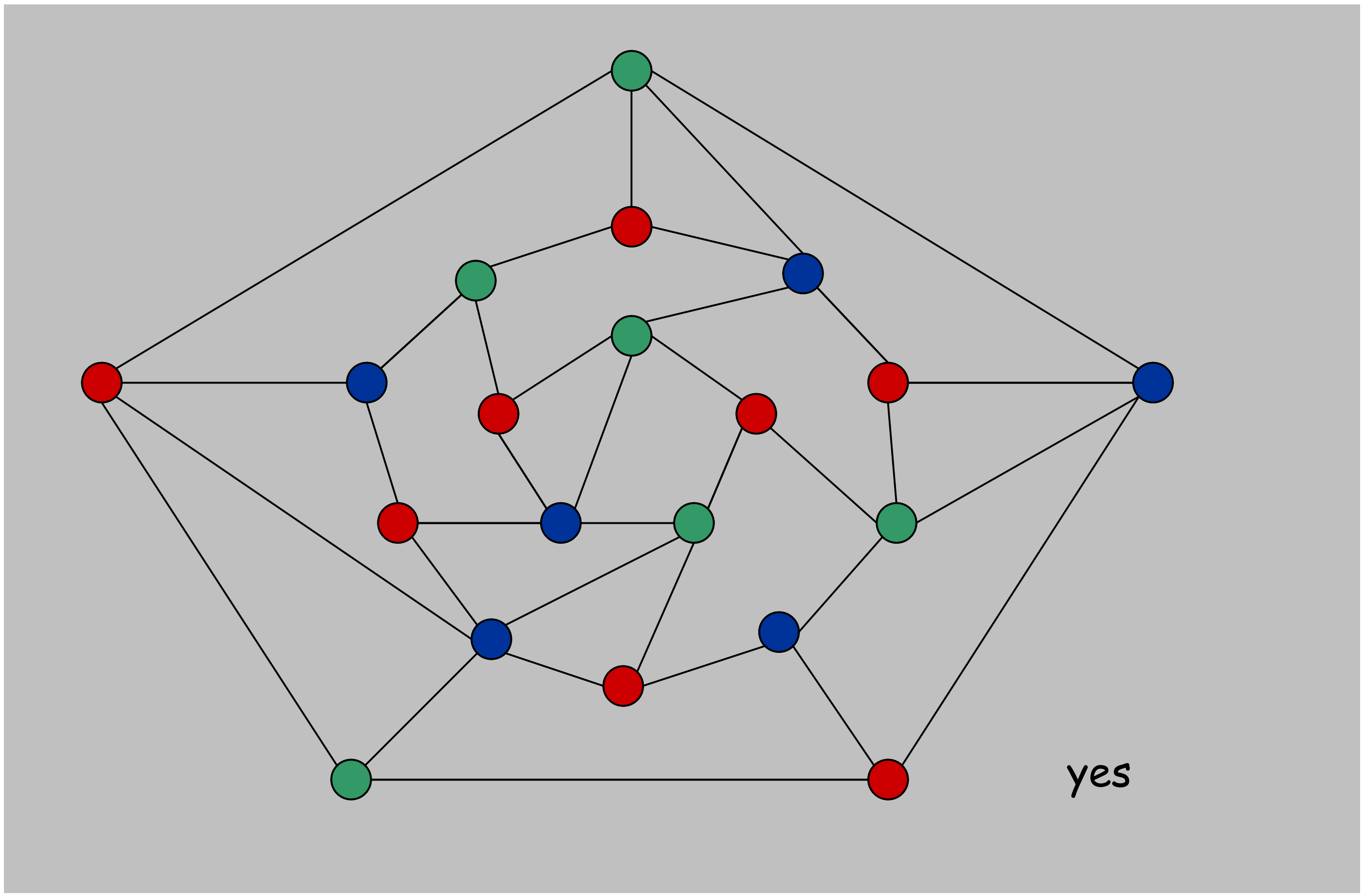
$$d(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 2 & \text{if } (u, v) \notin E \end{cases}$$

- Istanza di TSP ha tour di lunghezza $\leq$ n sse G è Hamiltoniano ▪

8.6 Problemi di Partitioning

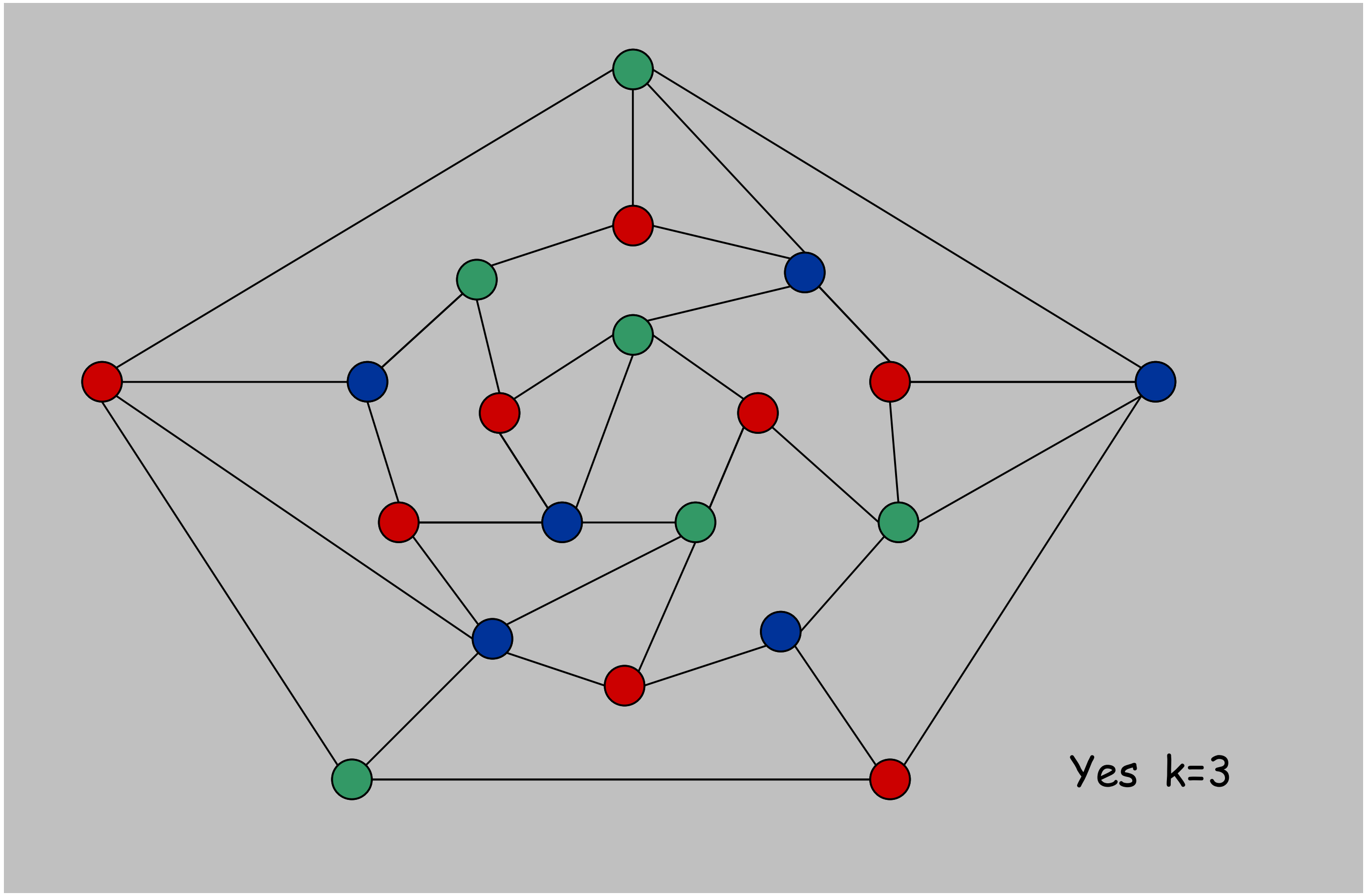
3-Colorabilità

3-COLOR: Dato un grafo non orientato G esiste un modo di colorare i nodi in rosso, blu, o verde in modo che nodi adiacenti NON hanno lo stesso colore?



k-Colorabilità

k-COLOR: Dato un grafo non orientato G esiste un modo di colorare I nodi usando k colori in modo che nodi adiacenti NON hanno lo stesso colore?



Allocazione di Registri

Allocazione di registri. Assegna le variabili di un programma ai registri della macchina in modo tale che non più di k registri sono utilizzati e non esistono 2 variabili che sono utilizzate contemporaneamente e sono assegnate allo stesso registro

Fact. k-COLOR $\leq_P$ k-REGISTER-ALLOCATION per una qualsiasi costante $k \geq 3$.

Allocazione di Registri

Allocazione di registri. Assegna le variabili di un programma ai registri della macchina in modo tale che non più di k registri sono utilizzati e non esistono 2 variabili che sono utilizzate contemporaneamente e sono assegnate allo stesso registro

Fact. 3-COLOR $\leq_P$ k-REGISTER-ALLOCATION per una qualsiasi costante $k \geq 3$.

Data istanza di k-color $G=(V,E)$
Creiamo Grafo dei conflitti=$G=(V, E)$
- V= {variabili }
- (u,v) in E se esiste un'operazione dove u e v sono variabili "live" allo stesso tempo

Osservazione. Possiamo risolvere problema dell'allocazione di k registri (k-REGISTER-ALLOCATION) sse il grafo dei conflitti è k-colorable.

Nota. Identifichiamo i colori con i registri

Fatto. 3-SAT $\leq_P$ 3-COLOR.

Dim. Dato istanza 3-SAT Φ, costruiamo istanza di 3-COLOR che è 3-colorabile sse Φ è soddisfacibile .

Costruzione .

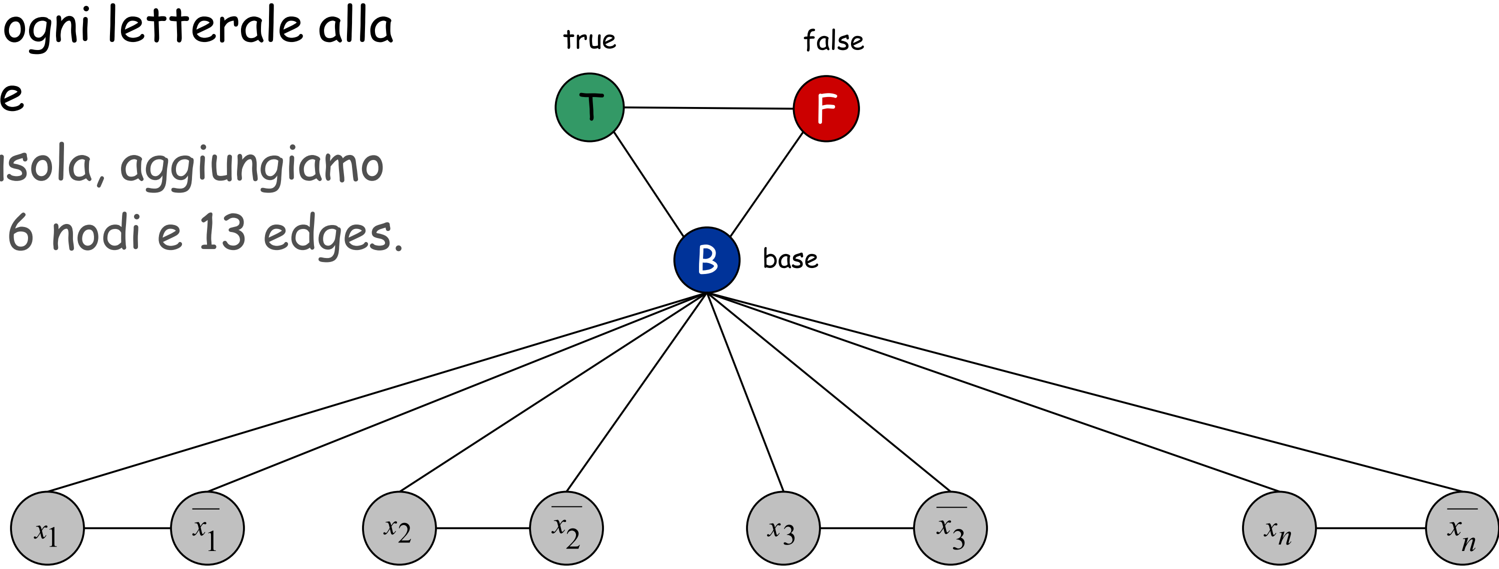
i. per ogni letterale, creiamo un nodo .

ii. Creiamo 3 nuovi nodi T, F, B; e li connettiamo in un triangolo, connettiamo ogni letterale a B.

iii. Connettiamo ogni letterale alla sua negazione

i. per ogni clausola, aggiungiamo un gadget di 6 nodi e 13 edges.

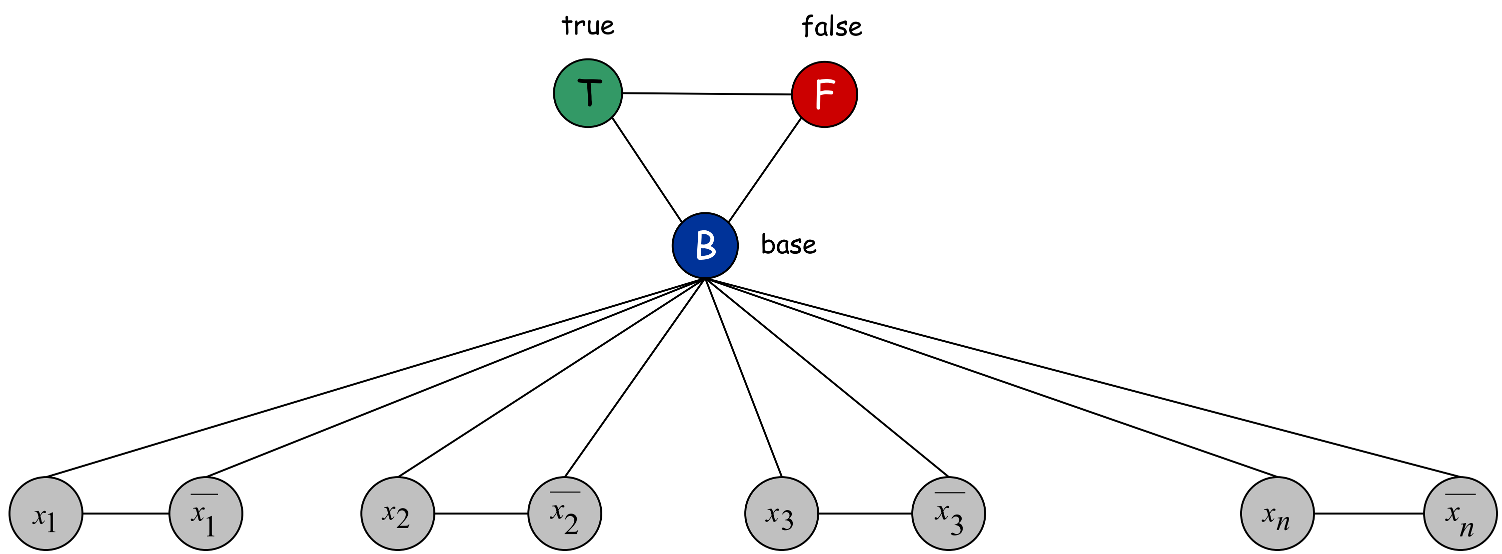
Vedremo dopo



3-Colorability

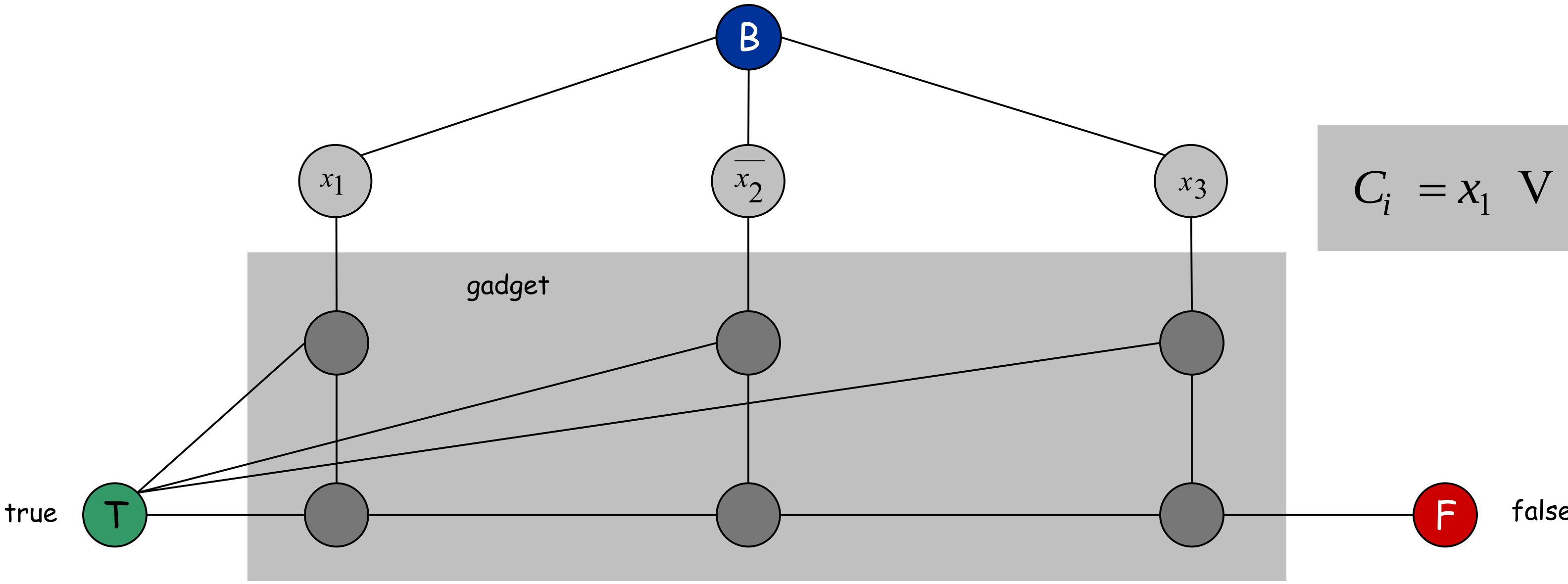
- (ii) assicura che T, F e B hanno 3 colori diversi
- (iii) assicura un letterale e la sua negazione hanno colori diversi e diversi da B

Quindi usiamo colori: T, F e B (colore base)
Ogni letterale ha valore T of F



- (ii) assicura che ogni letterale è T o F
- (iii) assicura un letterale e la sua negazione hanno valori opposti.
- (iv) Gadget assicura che almeno un letterale in ogni clausola è T.

Supponiamo che siano tutti F, …



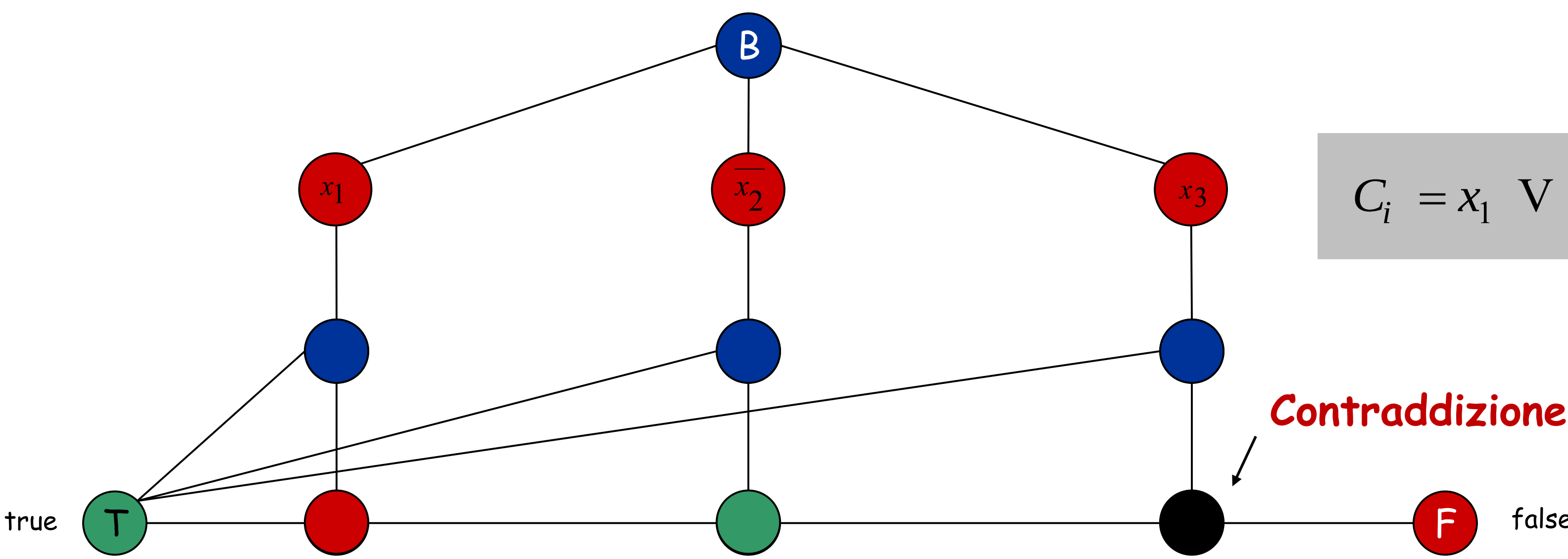
$$C_i = x_1 \vee \overline{x_2} \vee x_3$$

3-Colorability

- (ii) assicura che ogni letterale è T o F
- (iii) assicura un letterale e la sua negazione hanno valori opposti.
- (iv) Gadget assicura che almeno un letterale in ogni clausola è T.

Supponiamo che siano tutti F, …

… non 3-colorabile



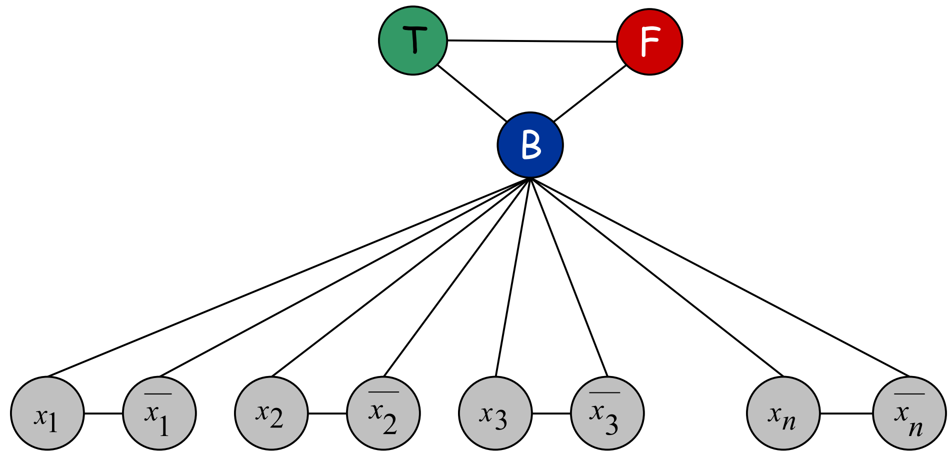
$$C_i = x_1 \ \vee \ \overline{x_2} \ \vee \ x_3$$

Fatto. 3-SAT $\leq_P$ 3-COLOR.

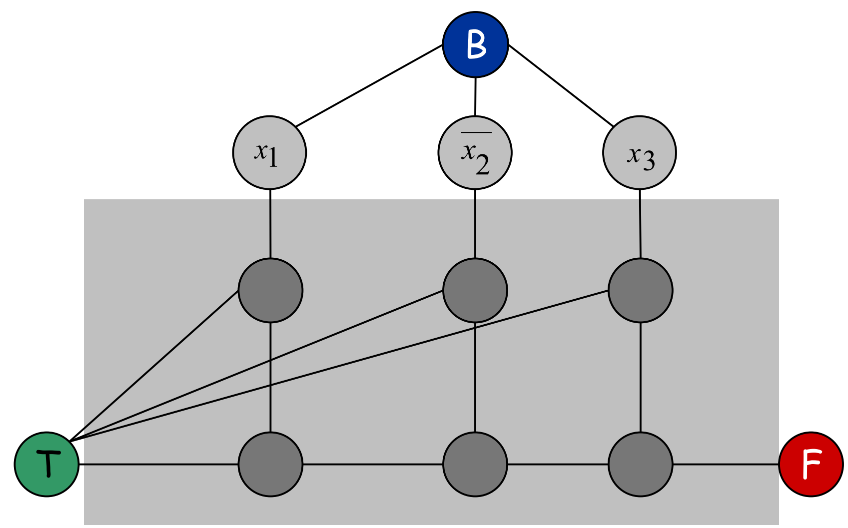
Dim. Data Φ, costruiamo G che è 3-colorabile sse Φ è soddisfacibile .

Per ottenere G,

a) costruiamo



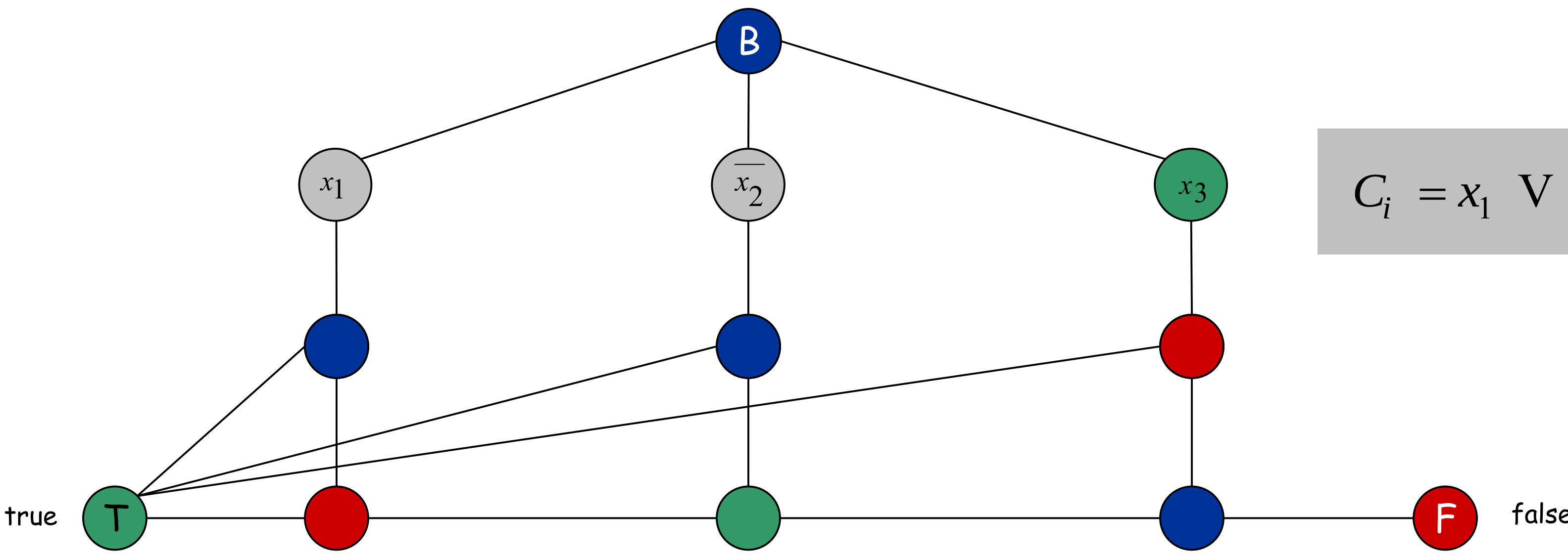
b) Per ogni clausola aggiungiamo
I 6 nodi del gadget ed
i 13 edge incidenti su di essi



Possiamo 3-colorare G implica che
esiste colorazione T,F dei letterali
(da a) ogni letterale e suo opposto colori diversi, da b) ogni clausola ha un T)
quindi formula soddisfacibile

Assumiamo che la formula Φ è soddisfacibile .

- Coloriamo base con B, nodo T con T, nodo F con F .
- Coloriamo tutti letterali true con T e false con F.
- Coloriamo nodo sotto letterale true con F, e
  nodo sotto letterale false con B.
- Coloriamo nodi restanti come imposto dai passi precedenti. ▪



$$C_i = x_1 \lor \overline{x_2} \lor x_3$$

8.8 Problemi Numerici

Subset Sum

SUBSET-SUM. Dati numeri naturali $w_1, \ldots, w_n$ e intero W, esiste sottoinsieme la cui somma è esattamente W?

Es: { 1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344 }, W = 3754.
Yes. 1 + 16 + 64 + 256 + 1040 + 1093 + 1284 = 3754.

Nota **con problemi, aritmentici input interi sono codificati in binario. Riduzione polinomiale deve esserlo nella lunghezza della codifica.**

Fatto. 3-SAT $\leq_P$ SUBSET-SUM.
Dim. Data istanza Φ di 3-SAT, costruiamo istanza di SUBSET-SUM che ha soluzione sse Φ è soddisfacibile .

Subset Sum

costruzione . Dato istanza Φ con n variabili e k clausole, formiamo 2n + 2k interi, ognuno di n+k digit

Fatto. Φ è soddisfacibile sse esiste sottoinsieme la cui somma è W.
Dim. No si hanno riporti.

n= k = 3

12 interi

$$C_1 = \bar{x} \vee y \vee z$$
$$C_2 = x \vee \bar{y} \vee z$$
$$C_3 = \bar{x} \vee \bar{y} \vee \bar{z}$$

Righe ulteriori per ottenere 12 righe
Per ogni C abbiamo due righe con solo 1 (risp. 2) in corrispondenza di C e tutti 0 altrove

	x	y	z	C_1	C_2	C_3	
x	1	0	0	0	1	0	100,010
¬ x	1	0	0	1	0	1	100,101
y	0	1	0	1	0	0	10,100
¬ y	0	1	0	0	1	1	10,011
z	0	0	1	1	1	0	1,110
¬ z	0	0	1	0	0	1	1,001
	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
W	1	1	1	4	4	4	111,444

Se formula soddisfacibile

prendiamo numeri corrispondenti a letterali True

$$C_1 = \bar{x} \vee y \vee z$$
$$C_2 = x \vee \bar{y} \vee z$$
$$C_3 = \bar{x} \vee \bar{y} \vee \bar{z}$$

(es: y,z true e x false)

Aggiustiamo con righe supplementari
per far sommare a 4 le colonne
relative alle clausole

Viceversa, se ho **sottoinsieme**

Consideriamo le **righe corrispondenti**

W= 111,444

4 assicura che ogni C ha un letterale true

1 evita di prendere letterale e suo negato

Poniamo true i letterali "**verdi**"

	x	y	z	C_1	C_2	C_3	
x	1	0	0	0	1	0	100,010
¬ x	1	0	0	1	0	1	100,101
y	0	1	0	1	0	0	10,100
¬ y	0	1	0	0	1	1	10,011
z	0	0	1	1	1	0	1,110
¬ z	0	0	1	0	0	1	1,001
	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
W	1	1	1	4	4	4	111,444

Scheduling con Release Times

SCHEDULE-RELEASE-TIMES. Dato un insieme di n jobs con processing time t_i, release time r_i, e deadline d_i, è possibile schedulare tutti i job su una sola macchina in modo che che job i è processato in uno slot di t_i time units *contigue* nell'intervallo $[r_i, d_i]$?

Fatto. SUBSET-SUM $\leq_P$ SCHEDULE-RELEASE-TIMES.

Dim. Dato un istanza di SUBSET-SUM $w_1, \ldots, w_n$, e target W,

- Crea n jobs con processing time $t_i = w_i$, release time $r_i = 0$, e no deadline ($d_i = 1 + \Sigma_j w_j$).
- Crea job 0 con $t_0 = 1$, release time $r_0 = W$, e deadline $d_0 = W+1$.

E possibile schedulare job da 1 a n ovunque tranne che in [W, W+1]