

# Polynomial-Time Reductions



# Contents

## Contents.

- Polynomial-time reductions.
- Reduction from special case to general case.
  - COMPOSITE reduces to FACTOR
  - VERTEX-COVER reduces to SET-COVER
- Reduction by simple equivalence.
  - PRIMALITY reduces to COMPOSITE, and vice versa
  - VERTEX COVER reduces to CLIQUE, and vice versa
- Reduction from general case to special case.
  - SAT reduces to 3-SAT
  - 3-COLOR reduces to PLANAR-3-COLOR
- Reduction by encoding with gadgets.
  - 3-CNF-SAT reduces to CLIQUE
  - 3-CNF-SAT reduces to HAM-CYCLE
  - 3-CNF-SAT reduces to 3-COLOR

# Polynomial-Time Reduction

Intuitively, problem X reduces to problem Y if:

- Any instance of X can be "rephrased" as an instance of Y.

Formally, problem X **polynomial reduces** to problem Y if arbitrary instances of problem X can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y.
  - computational model supplemented by special piece of hardware that solves instances of Y in a single step

**Remarks.**

- We pay for time to write down instances sent to black box  $\Rightarrow$  instances of Y are of polynomial size.
- Note: Cook-Turing reducibility (not Karp or many-to-one).
- Notation:  $X \leq_p Y$  (or more precisely  $X \leq_p^T Y$  ).

# Polynomial-Time Reduction

**Purpose.** Classify problems according to relative difficulty.

**Design algorithms.** If  $X \leq_p Y$  and  $Y$  can be solved in polynomial-time, then  $X$  can be solved in polynomial time.

**Establish intractability.** If  $X \leq_p Y$  and  $X$  cannot be solved in polynomial-time, then  $Y$  cannot be solved in polynomial time.

**Anti-symmetry.** If  $X \leq_p Y$  and  $Y \leq_p X$ , we use notation  $X \equiv_p Y$ .

**Transitivity.** If  $X \leq_p Y$  and  $Y \leq_p Z$ , then  $X \leq_p Z$ .

- Proof idea: compose the two algorithms.
- Given an oracle for  $Z$ , can solve instance of  $X$ :
  - run the algorithm for  $X$  using an oracle for  $Y$
  - each time oracle for  $Y$  is called, simulate it in a polynomial number of steps by using algorithm for  $Y$ , plus oracle calls to  $Z$

# Polynomial-Time Reduction

## Basic strategies.

- **Reduction by simple equivalence.**
- Reduction from special case to general case.
- Reduction from general case to special case.
- Reduction by encoding with gadgets.

# Compositeness and Primality

**COMPOSITE:** Given the decimal representation of an integer  $x$ , does  $x$  have a nontrivial factor?

**PRIME:** Given the decimal representation of an integer  $x$ , is  $x$  prime?

**Claim.**  $\text{COMPOSITE} \equiv_p \text{PRIME}$ .

- $\text{COMPOSITE} \leq_p \text{PRIME}$ .
- $\text{PRIME} \leq_p \text{COMPOSITE}$ .

## COMPOSITE (x)

```
IF (PRIME(x) = TRUE)
  RETURN FALSE
ELSE
  RETURN TRUE
```

## PRIME (x)

```
IF (COMPOSITE(x) = TRUE)
  RETURN FALSE
ELSE
  RETURN TRUE
```

# Vertex Cover

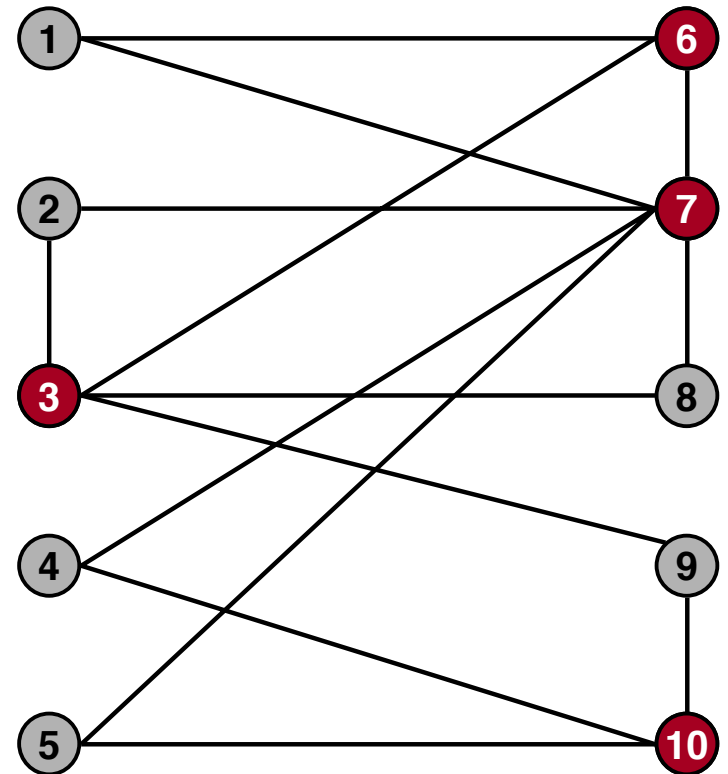
**VERTEX COVER:** Given an undirected graph  $G = (V, E)$  and an integer  $k$ , is there a subset of vertices  $S \subseteq V$  such that  $|S| \leq k$ , and if  $(v, w) \in E$  then either  $v \in S$ ,  $w \in S$  or both.

Básicamente, corresponde al conjunto de vértices que contienen todos los arcos

**Ex.**

- Is there a vertex cover of size 4?

**YES.**



# Vertex Cover

**VERTEX COVER:** Given an undirected graph  $G = (V, E)$  and an integer  $k$ , is there a subset of vertices  $S \subseteq V$  such that  $|S| \leq k$ , and if  $(v, w) \in E$  then either  $v \in S$ ,  $w \in S$  or both.

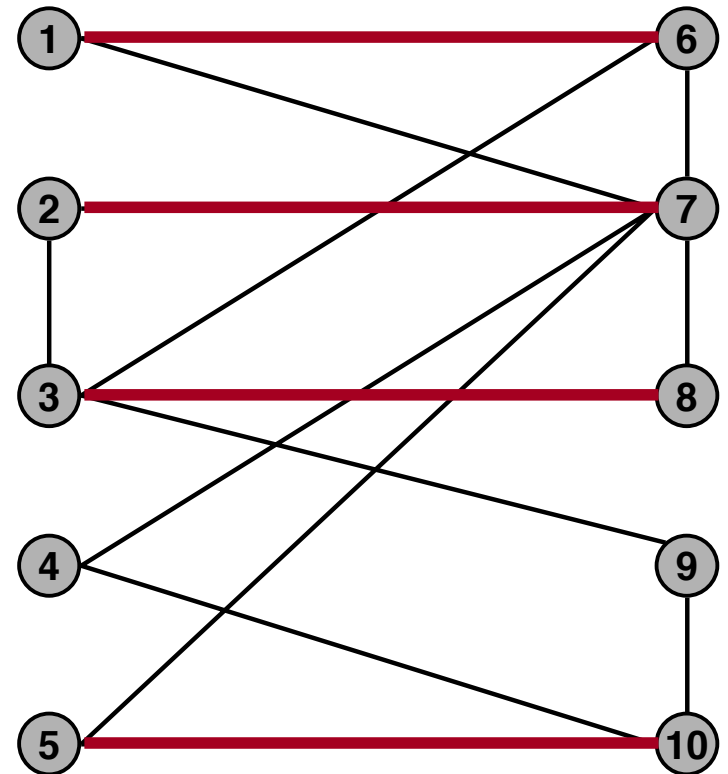
Ex.

- Is there a vertex cover of size 4?

**YES.**

- Is there a vertex cover of size 3?

**NO.**





# Clique

**CLIQUE:** Given  $N$  people and their pairwise relationships. Is there a group of  $S$  people such that every pair in the group knows each other.

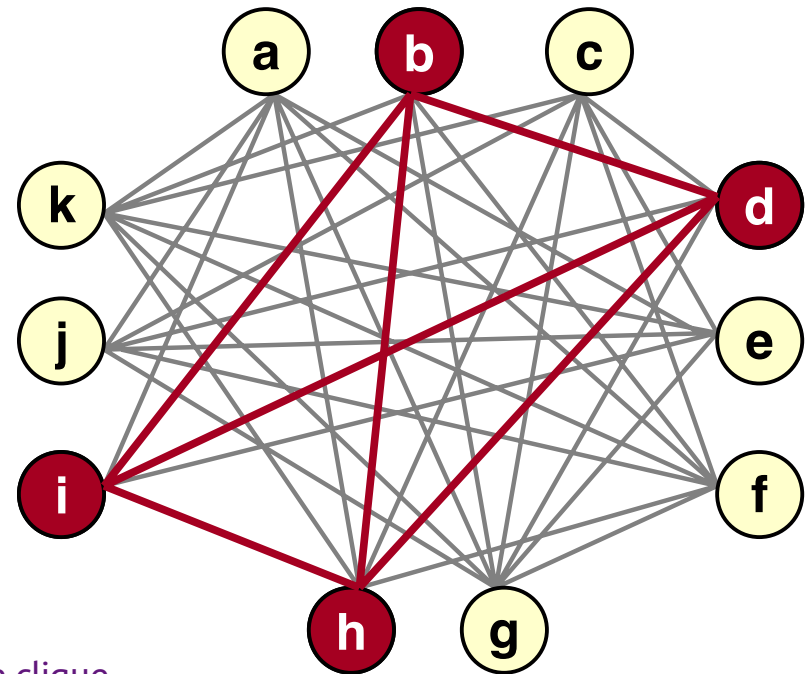
Ex.

- **People:**  $a, b, c, d, e, \dots, k$ .
- **Friendships:**  $(a, e), (a, f), (a, g), \dots, (h, k)$ .
- **Clique size:**  $S = 4$ .

YES Instance

Un clique es un subgrafo completo dentro de otro grafo (el grafo que contiene al clique no necesariamente debe ser completo). Recuerde que un grafo completo, todos los nodos están conectados, a través de arcos, con todos los demás nodos.

Nótese que si un grafo tiene un clique de tamaño  $k$ , entonces tiene un clique de tamaño  $k-1, k-2, \dots, 1$ . Para visualizarlo en el ejemplo, elimine el nodo "d" del clique que se destaca ¿El grafo resultante sigue siendo clique?



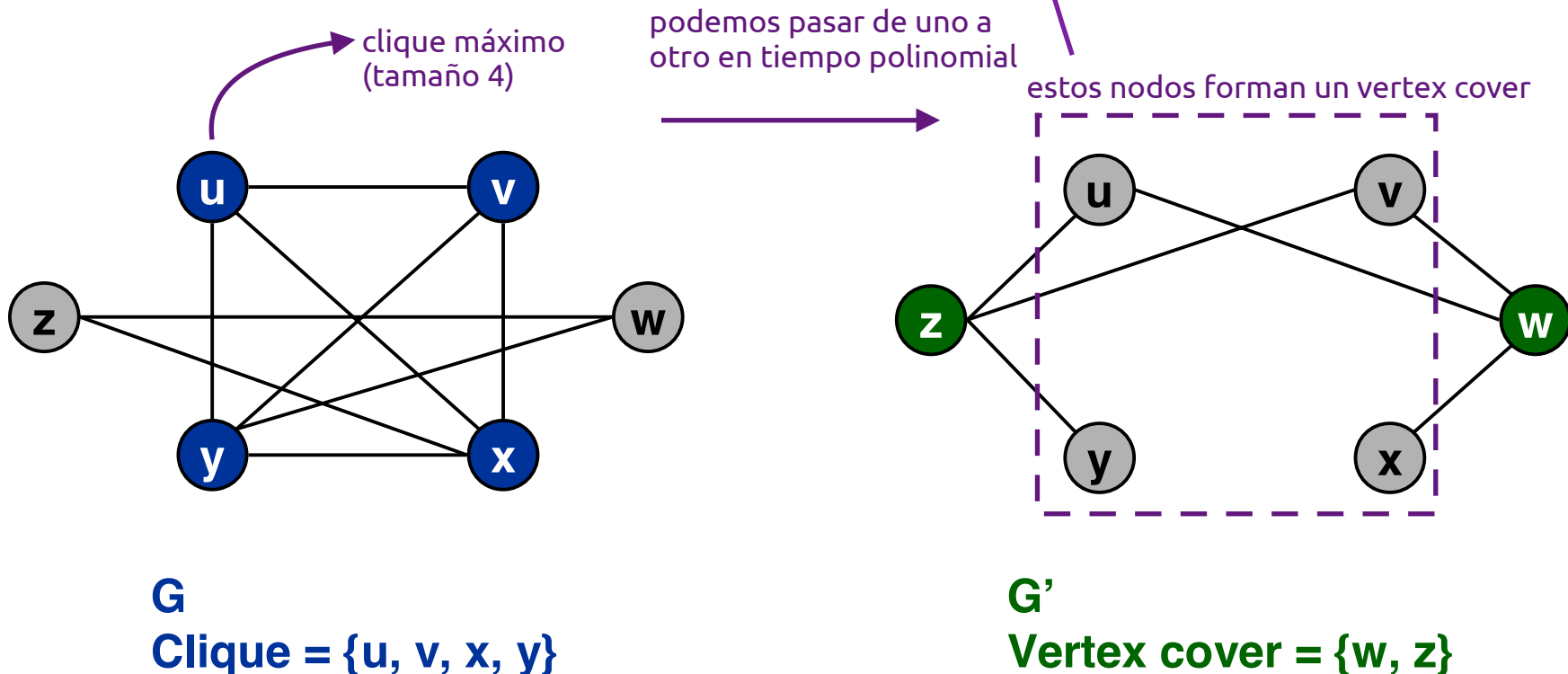
Friendship Graph

# Vertex Cover and Clique

**Claim.**  $\text{VERTEX COVER} \equiv_p \text{CLIQUE}$ .

- Given an undirected graph  $G = (V, E)$ , its complement is  $G' = (V, E')$ , where  $E' = \{ (v, w) : (v, w) \notin E \}$ .
- $G$  has a clique of size  $k$  if and only if  $G'$  has a vertex cover of size  $|V| - k$ .

Importante: el complemento de un clique resulta un vertex cover y viceversa (2 nodos forman parte del vertex cover)



Para buscar que un grafo tiene un clique usamos búsqueda binaria (esto no es tan alejado de responder si o no)

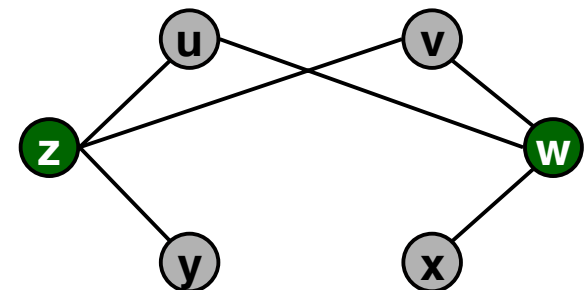
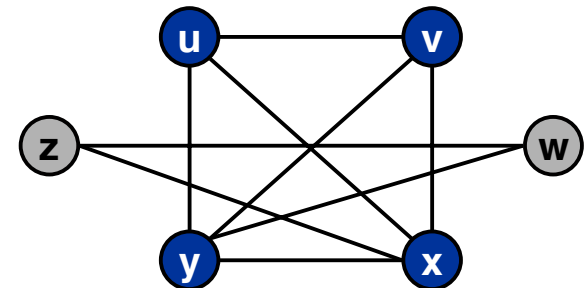
# Vertex Cover and Clique

**Claim.** VERTEX COVER  $\equiv_p$  CLIQUE.

- Given an undirected graph  $G = (V, E)$ , its complement is  $G' = (V, E')$ , where  $E' = \{ (v, w) : (v, w) \notin E \}$ .
- $G$  has a clique of size  $k$  if and only if  $G'$  has a vertex cover of size  $|V| - k$ .

**Proof.**  $\Rightarrow$

- Suppose  $G$  has a clique  $S$  with  $|S| = k$ .
- Consider  $S' = V - S$ .
- $|S'| = |V| - k$ .
- To show  $S'$  is a cover, consider any edge  $(v, w) \in E'$ .
  - then  $(v, w) \notin E$
  - at least one of  $v$  or  $w$  is not in  $S$  (since  $S$  forms a clique)
  - at least one of  $v$  or  $w$  is in  $S'$
  - hence  $(v, w)$  is covered by  $S'$



# Vertex Cover and Clique

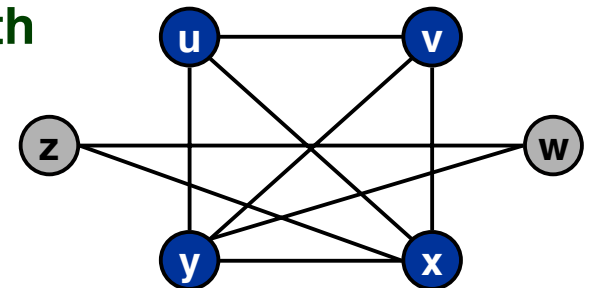
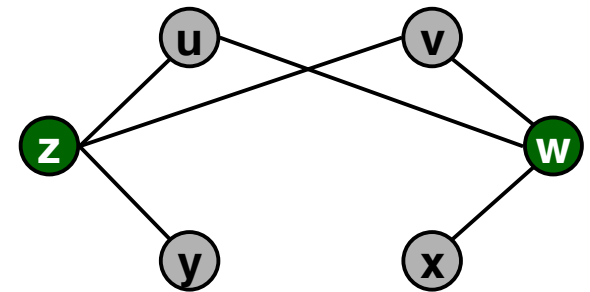
**Claim.** VERTEX COVER  $\equiv_p$  CLIQUE.

- Given an undirected graph  $G = (V, E)$ , its complement is  $G' = (V, E')$ , where  $E' = \{ (v, w) : (v, w) \notin E \}$ .
- $G$  has a clique of size  $k$  if and only if  $G'$  has a vertex cover of size  $|V| - k$ .

para demostrar que dos cosas son equivalentes tenemos que hacer la demostración en ambos sentidos

**Proof.**  $\Leftarrow$

- Suppose  $G'$  has a cover  $S'$  with  $|S'| = |V| - k$ .
- Consider  $S = V - S'$ .
- Clearly  $|S| = k$ .
- To show  $S$  is a clique, consider some edge  $(v, w) \in E'$ .
  - if  $(v, w) \in E'$ , then either  $v \in S'$ ,  $w \in S'$ , or both
  - by contrapositive, if  $v \notin S'$  and  $w \notin S'$ , then  $(v, w) \in E$
  - thus  $S$  is a clique in  $G$



# Polynomial-Time Reduction

## Basic strategies.

- Reduction by simple equivalence.
- **Reduction from special case to general case.**
- Reduction from general case to special case.
- Reduction by encoding with gadgets.

# Compositeness Reduces to Factoring

factor trivial de X es 1 y X (son 1 y el mismo número)

**COMPOSITE:** Given an integer  $x$ , does  $x$  have a nontrivial factor?

**FACTOR:** Given two integers  $x$  and  $y$ , does  $x$  have a nontrivial factor less than  $y$ ?

**Claim.**  $\text{COMPOSITE} \leq_p \text{FACTOR}$ .

**Proof.** Given an oracle for FACTOR, we solve COMPOSITE.

- Is 350 composite?
- Does 350 have a nontrivial factor less than 350?

COMPOSITE ( $x$ )
<pre>IF (FACTOR(<math>x</math>, <math>x</math>) = TRUE)   RETURN TRUE ELSE   RETURN FALSE</pre>

# Primality Testing and Factoring

Problemas NP intermedios: son problemas que no están en P y que no son NP-completo

## We established:

- $\text{PRIME} \leq_p \text{COMPOSITE} \leq_p \text{FACTOR}$ .

nadie sabe si está en NP-duro

## Natural question:

- Does  $\text{FACTOR} \leq_p \text{PRIME}$  ?
- Consensus opinion = NO.

## State-of-the-art.

- $\text{PRIME}$  in **randomized P** and conjectured to be in P.
- $\text{FACTOR}$  not believed to be in P.

tiene dos respuestas: "No" y "no se la respuesta" (esto da la posibilidad de responder "sí", pero no se sabe) (este se prueba varias veces) (la posibilidad de equivocarse va tendiendo a cero)

resulta mucho más barato correr un algoritmo randomizado muchas veces que el verdadero algoritmo que sabe la respuesta

## RSA cryptosystem.

- Based on dichotomy between two problems.
- To use, must generate large primes efficiently.
- Can break with efficient factoring algorithm.

# Set Cover

La unión de todos los  $S_i$  forman el universo completo

$S_i$  son conjuntos

**SET COVER:** Given a set  $U$  of elements, a collection  $S_1, S_2, \dots, S_m$  of subsets of  $U$ , and an integer  $k$ , does there exist a collection of at most  $k$  of these sets whose union is equal of  $U$ ?

## Sample application.

- $n$  available pieces of software.
- Set  $U$  of  $n$  capabilities that we would like our system to have.
- The  $i$ th piece of software provides the set  $S_i \subseteq U$  of capabilities.
- Goal: achieve all  $n$  capabilities using small number of pieces of software.

**Ex.**  $U = \{1, 2, 3, \dots, 12\}$ ,  $k = 3$ .

- $S_1 = \{1, 2, 3, 4, 5, 6\}$       $S_2 = \{5, 6, 8, 9\}$
- $S_3 = \{1, 4, 7, 10\}$       $S_4 = \{2, 5, 7, 8, 11\}$
- $S_5 = \{3, 6, 9, 12\}$       $S_6 = \{10, 11\}$

**YES:  $S_3, S_4, S_5$ .**

no está prohibido que conjuntos tengan elementos en común



# Vertex Cover Reduces to Set Cover

**SET COVER:** Given a set  $U$  of elements, a collection  $S_1, S_2, \dots, S_n$  of subsets of  $U$ , and an integer  $k$ , does there exist a collection of at most  $k$  of these sets whose union is equal to  $U$ ?

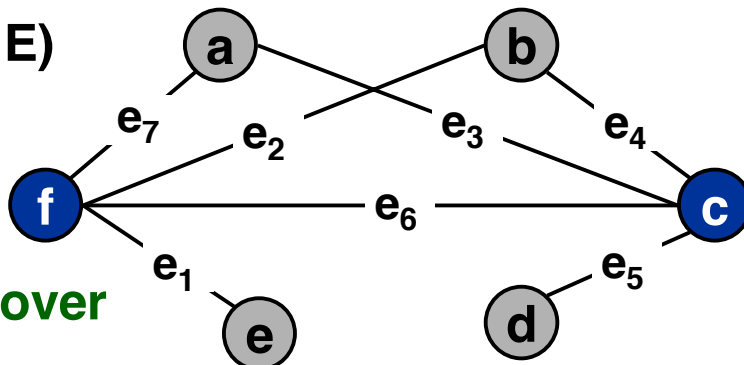
**VERTEX COVER:** Given an undirected graph  $G = (V, E)$  and an integer  $k$ , is there a subset of vertices  $S \subseteq V$  such that  $|S| \leq k$ , and if  $(v, w) \in E$  then either  $v \in S$ ,  $w \in S$  or both.

conjunto de nodos que tengan todos los arcos

**Claim.**  $\text{VERTEX-COVER} \leq_p \text{SET-COVER}$ .

**Proof.** Given black box that solves instances of SET-COVER.

$G = (V, E)$   
 $k = 2$



Vertex Cover

$U = \{1, 2, 3, 4, 5, 6, 7\}$

$k = 2$

$S_a = \{3, 7\}$

$S_b = \{2, 4\}$

$S_c = \{3, 4, 5, 6\}$

$S_d = \{5\}$

$S_e = \{1\}$

$S_f = \{1, 2, 6, 7\}$

Set Cover

# Vertex Cover Reduces to Set Cover

**SET COVER:** Given a set  $U$  of elements, a collection  $S_1, S_2, \dots, S_n$  of subsets of  $U$ , and an integer  $k$ , does there exist a collection of at most  $k$  of these sets whose union is equal to  $U$ ?

**VERTEX COVER:** Given an undirected graph  $G = (V, E)$  and an integer  $k$ , is there a subset of vertices  $S \subseteq V$  such that  $|S| \leq k$ , and if  $(v, w) \in E$  then either  $v \in S$ ,  $w \in S$  or both.

**Claim.**  $\text{VERTEX-COVER} \leq_p \text{SET-COVER}$ .

**Proof.** Given black box that solves instances of SET-COVER.

- Let  $G = (V, E)$ ,  $k$  be an instance of VERTEX-COVER.
- Create SET-COVER instance:
  - $k = k$ ,  $U = E$ ,  $S_v = \{e \in E : e \text{ incident to } v\}$
- Set-cover of size at most  $k$  if and only if vertex cover of size at most  $k$ .

# Polynomial-Time Reduction

## Basic strategies.

- Reduction by simple equivalence.
- Reduction from special case to general case.
- **Reduction from general case to special case.**
- Reduction by encoding with gadgets.

# Factoring and Finding Factors

**FACTOR:** Given two integers  $x$  and  $y$ , does  $x$  have a nontrivial factor less than  $y$ ?

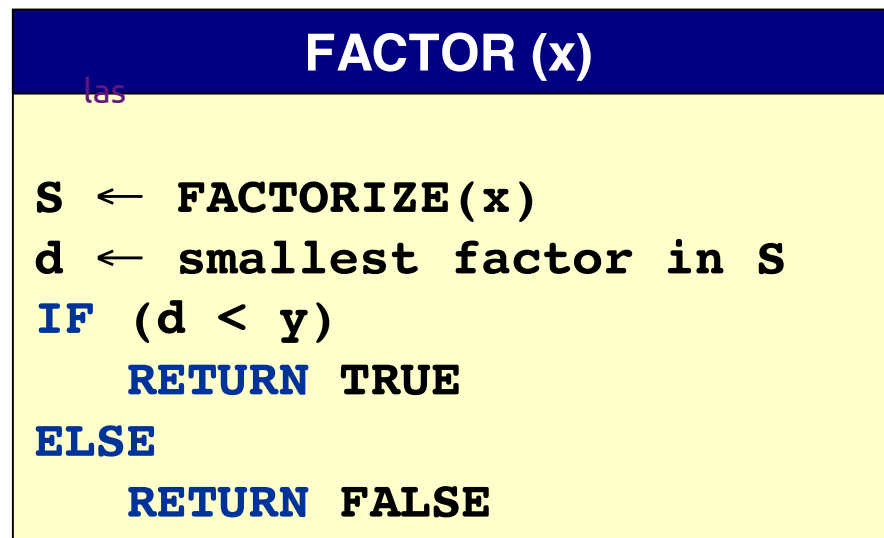
**FACTORIZE:** Given an integer  $x$ , find its prime factorization.

**Claim.**  $\text{FACTORIZE} \equiv_p \text{FACTOR}$ .

**Proof:**  $\text{FACTOR} \leq_p \text{FACTORIZE}$ .

- Reduction from special case to general case.

**S = prime factorization  
is of polynomial size**



Las variables que se mencionan en este algoritmo corresponden a los que se definen en esta diapositiva.

# Factoring and Finding Factors

**FACTOR:** Given two integers  $x$  and  $y$ , does  $x$  have a nontrivial factor less than  $y$ ?

**FACTORIZE:** Given an integer  $x$ , find its prime factorization.

**Claim.**  $\text{FACTORIZE} \equiv_p \text{FACTOR}$ .

**Proof:**  $\text{FACTORIZE} \leq_p \text{FACTOR}$ .

- Reduction from general case to special case.

si iteramos a lo "bruto", ya no será polinomial, por eso debemos hacer búsqueda binaria.

find smallest factor  
via binary search

$S$  = global variable  
containing set of factors

## FACTORIZE( $x$ )

**IF** ( $\text{FACTOR}(x, x) = \text{NO}$ )

$S \leftarrow S \cup \{x\}$

**RETURN**

En el fondo, lo que estamos haciendo es búsqueda binaria

$\text{left} = 1, \text{right} = x$

**WHILE** ( $\text{right} > \text{left} + 1$ )

$\text{mid} = (\text{left} + \text{right}) / 2$

**IF** ( $\text{FACTOR}(x, \text{mid}) = \text{TRUE}$ )

$\text{right} = \text{mid}$

**ELSE**

$\text{left} = \text{mid}$

$S \leftarrow S \cup \{\text{left}\}$

**FACTORIZE**( $x / \text{left}$ )

$\setminus \text{setminus minus}$

# Satisfiability

**Literal:** A Boolean variable or its negation.

$$x_i \text{ or } \overline{x_i}$$

**Clause:** A disjunction of literals.

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

**Conjunctive normal form:** A Boolean formula that is the conjunction of clauses.

$$B = C_1 \wedge C_2 \wedge C_3 \wedge C_4$$

**CNF-SAT:** Given propositional formula in conjunctive normal form, does it have a satisfying truth assignment?

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

**YES instance**

**$x_1 = \text{true}$**

**$x_2 = \text{true}$**

**$x_3 = \text{false}$**

# SAT Reduces to 3-SAT

**3-CNF-SAT:** CNF-SAT, where each clause has 3 distinct literals.

**Claim.**  $\text{CNF-SAT} \leq_p \text{3-CNF-SAT}$ .

- Case 3: clause  $C_j$  contains exactly 3 terms.
- Case 2: clause  $C_j$  contains exactly 2 terms.
  - add 1 new term, and replace  $C_j$  with 2 clauses

$$\begin{aligned} C_j = \overline{x_3} \vee x_7 &\Rightarrow \begin{aligned} C'_{j1} &= \overline{x_3} \vee x_7 \vee y \\ C'_{j2} &= \overline{x_3} \vee x_7 \vee \overline{y} \end{aligned} \end{aligned}$$

Corresponde a lo mismo que se vio en clases.

- Case 1: clause  $C_j$  contains exactly 1 term.
  - add 4 new terms, and replace  $C_j$  with 4 clauses

$$\begin{aligned} C_j = \overline{x_3} &\Rightarrow \begin{aligned} C'_{j1} &= \overline{x_3} \vee y_1 \vee y_2 \\ C'_{j2} &= \overline{x_3} \vee y_1 \vee \overline{y_2} \\ C'_{j3} &= \overline{x_3} \vee \overline{y_1} \vee y_2 \\ C'_{j4} &= \overline{x_3} \vee \overline{y_1} \vee \overline{y_2} \end{aligned} \end{aligned}$$

# SAT Reduces to 3-SAT

**3-CNF-SAT:** CNF-SAT, where each clause has 3 distinct literals.

**Claim.**  $\text{CNF-SAT} \leq_p \text{3-CNF-SAT}$ .

- **Case 4:** clause  $C_j$  contains  $\ell \geq 4$  terms.
  - introduce  $\ell - 1$  extra Boolean variables
  - replace  $C_j$  with  $\ell$  clauses

$$C_j = x_1 \vee \overline{x_3} \vee \overline{x_4} \vee x_5 \vee x_6 \vee \overline{x_9} \Rightarrow \begin{array}{lcl} C'_{j1} & = & x_1 \vee \overline{x_1} \vee y_1 \\ C'_{j2} & = & \overline{y_1} \vee \overline{x_3} \vee y_2 \\ C'_{j3} & = & \overline{y_2} \vee \overline{x_4} \vee y_3 \\ C'_{j4} & = & \overline{y_3} \vee x_5 \vee y_4 \\ C'_{j5} & = & \overline{y_4} \vee x_6 \vee y_5 \\ C'_{j6} & = & \overline{y_5} \vee \overline{x_9} \vee \overline{x_9} \end{array}$$



# SAT Reduces to 3-SAT

- Case 4: clause  $C_j$  contains  $\ell \geq 4$  terms.

$$C_j = t_{j1} \vee t_{j2} \vee t_{j3} \vee \cdots \vee t_{j\ell} \Rightarrow$$

$C'_{j1}$	$=$	$t_{j1}$	$\vee$	$t_{j1}$	$\vee$	$y_1$
$C'_{j2}$	$=$	$\overline{y_1}$	$\vee$	$t_{j2}$	$\vee$	$y_2$
$C'_{j3}$	$=$	$\overline{y_2}$	$\vee$	$t_{j3}$	$\vee$	$y_3$
$C'_{j4}$	$=$	$\overline{y_3}$	$\vee$	$t_{j4}$	$\vee$	$y_4$
$C'_{j5}$	$=$	$\overline{y_4}$	$\vee$	$t_{j5}$	$\vee$	$y_5$
$\vdots$		$\vdots$		$\vdots$		$\vdots$
$C'_{j\ell}$	$=$	$\overline{y_{\ell-1}}$	$\vee$	$t_{j\ell}$	$\vee$	$t_{j\ell}$

$k = 4$  points to the row  $C'_{j4}$ .

**Claim.** CNF-SAT instance is satisfiable if and only if 3-CNF-SAT instance is.

**Proof.**  $\Rightarrow$  Suppose SAT instance is satisfiable.

- If SAT assignment sets  $t_{jk} = 1$ , 3-SAT assignment sets:
  - $t_{jk} = 1$
  - $y_m = 1$  for all  $m < k$ ;  $y_m = 0$  for all  $m \geq k$

set TRUE

# SAT Reduces to 3-SAT

- Case 2: clause  $C_j$  contains  $\ell \geq 4$  terms.

$$C_j = t_{j1} \vee t_{j2} \vee t_{j3} \vee \cdots \vee t_{j\ell} \Rightarrow \begin{array}{llll} C'_{j1} & = & t_{j1} & \vee & t_{j1} & \vee & y_1 \\ C'_{j2} & = & \overline{y_1} & \vee & t_{j2} & \vee & y_2 \\ C'_{j3} & = & \overline{y_2} & \vee & t_{j3} & \vee & y_3 \\ C'_{j4} & = & \overline{y_3} & \vee & t_{j4} & \vee & y_4 \\ C'_{j5} & = & \overline{y_4} & \vee & t_{j5} & \vee & y_5 \\ & \vdots & & & \vdots & & \vdots \\ C'_{j\ell} & = & \overline{y_{\ell-1}} & \vee & t_{j\ell} & \vee & t_{j\ell} \end{array}$$

**Claim.** CNF-SAT instance is satisfiable if and only if 3-CNF-SAT instance is.

**Proof.**  $\Leftarrow$  Suppose 3-SAT instance is satisfiable.

- If 3-SAT assignment sets  $t_{jk} = 1$ , SAT assignment sets  $t_{jk} = 1$ .
- Consider clause  $C_j$ . We claim  $t_{jk} = 1$  for some  $k$ .
  - each of  $\ell - 1$  new Boolean variables  $y_j$  can only make one of  $\ell$  new clauses true
  - the remaining clause must be satisfied by an original term  $t_{jk}$

# Polynomial-Time Reduction

## Basic strategies.

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction from general case to special case.
- **Reduction by encoding with gadgets.**

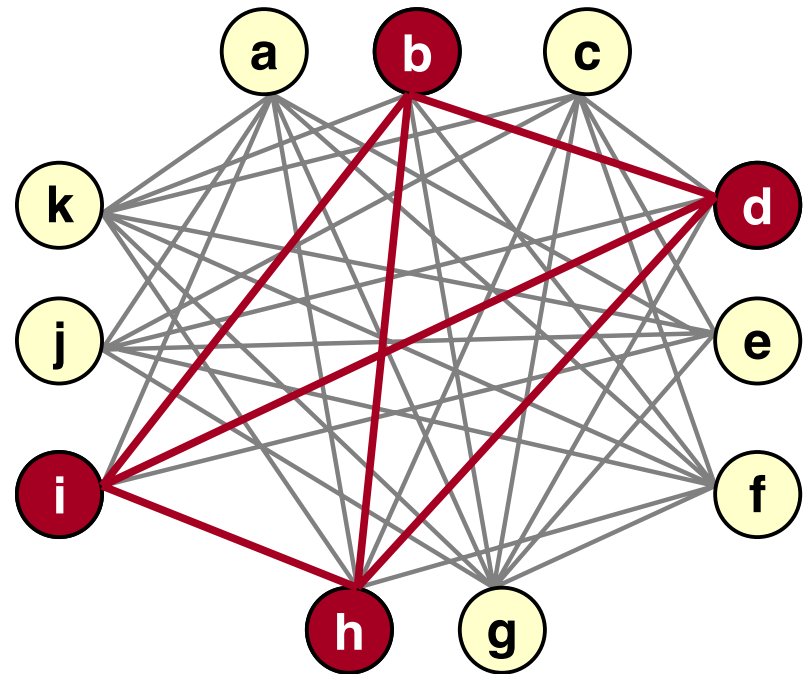
# Clique

**CLIQUE:** Given  $N$  people and their pairwise relationships. Is there a group of  $C$  people such that every pair in the group knows each other.

Ex.

- **People:**  $a, b, c, d, e, \dots, k$ .
- **Friendships:**  $(a, e), (a, f), (a, g), \dots, (h, k)$ .
- **Clique size:**  $C = 4$ .

YES Instance



Friendship Graph

# Satisfiability Reduces to Clique

**Claim.**  $\text{CNF-SAT} \leq_p \text{CLIQUE}$ .

- Given instance of CNF-SAT, create a person for each literal in each clause.

Importante, la notación en esta diapositiva es distinta:

- Tome  $x'$  como  $\neg x$  ( $x$  negado)
- Tome  $+$  con  $\vee$  (disyunción)
- Tome  $*$  como  $\wedge$  (conjunción)

first clause



$(x' + y + z) (x + y' + z) (y + z) (x' + y' + z')$   
 $C = 4 \text{ clauses}$

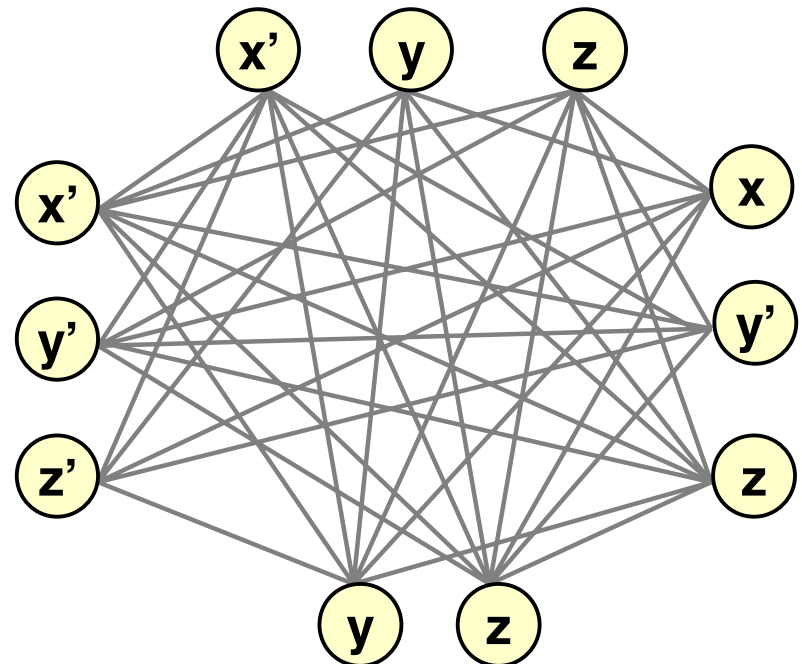
$k = 4$  (clique de tamaño 4)

# Satisfiability Reduces to Clique

**Claim.**  $\text{CNF-SAT} \leq_p \text{CLIQUE}$ .

- Given instance of CNF-SAT, create a person for each literal in each clause.
- Two people know each other except if:
  - they come from the same clause
  - they represent a literal and its negation

$(x' + y + z) (x + y' + z) (y + z) (x' + y' + z')$   
 $C = 4$  clauses



# Satisfiability Reduces to Clique

**Claim.**  $\text{CNF-SAT} \leq_p \text{CLIQUE}$ .

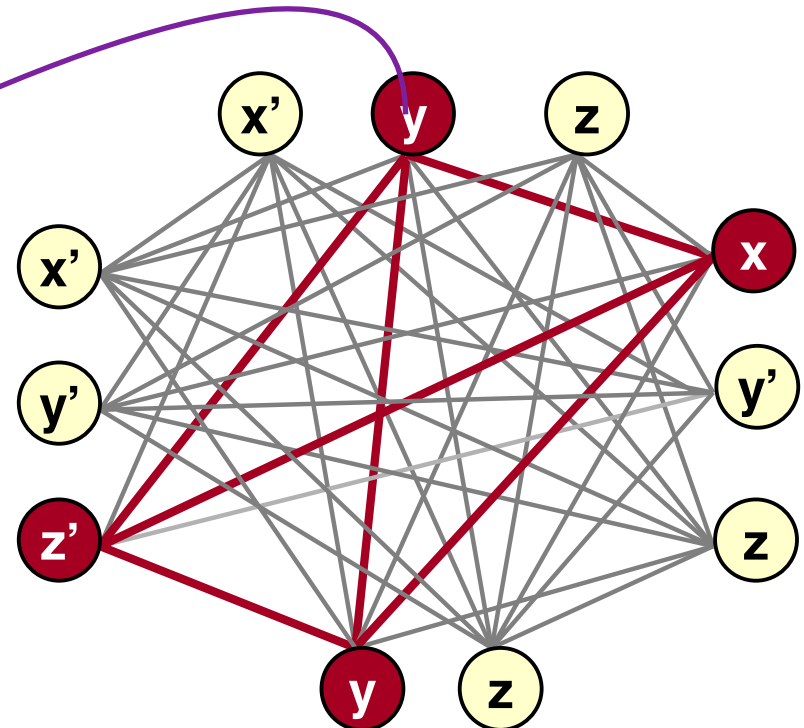
- Given instance of CNF-SAT, create a person for each literal in each clause.
- Two people know each other except if:
  - they come from the same clause
  - they represent a literal and its negation
- Clique of size  $C \Rightarrow$  satisfiable assignment.
  - set variable in clique to true
  - $(x, y, z) = (\text{true}, \text{true}, \text{false})$

Si tenemos un clique de tamaño  $k$ , entonces tendremos cliques de tamaño  $k-1, k-2, \dots, 1$

si todos estos literales son verdaderos, entonces la expresión booleana es verdadera

El tamaño del clique estará dado por la cantidad de cláusulas que tiene la expresión booleana

$(x' + y + z) (x + y' + z) (y + z) (x' + y' + z')$   
 $C = 4$  clauses

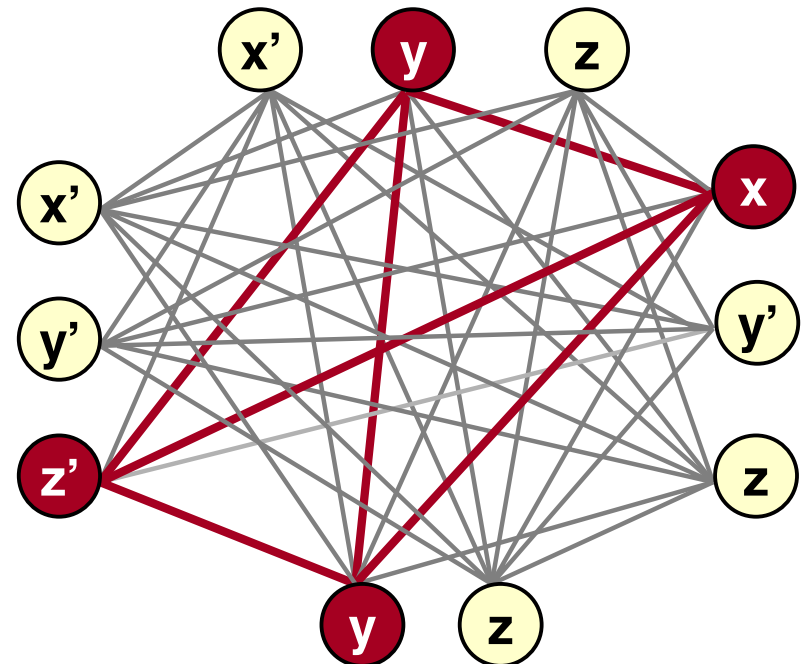


# Satisfiability Reduces to Clique

**Claim.**  $\text{CNF-SAT} \leq_p \text{CLIQUE}$ .

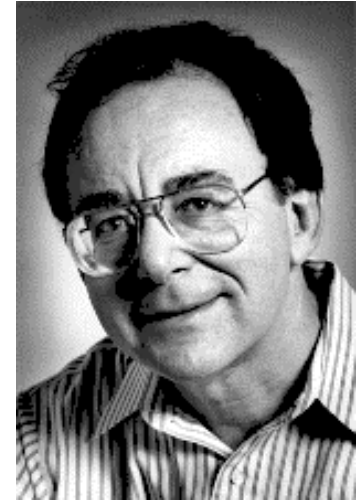
- Given instance of CNF-SAT, create a person for each literal in each clause.
- Two people know each other except if:
  - they come from the same clause
  - they represent a literal and its negation
- Clique of size  $C \Rightarrow$  satisfiable assignment.
- Satisfiable assignment  $\Rightarrow$  clique of size  $C$ .
  - $(x, y, z) = (\text{true}, \text{true}, \text{false})$
  - choose one true literal from each clause

$(x' + y + z) (x + y' + z) (y + z) (x' + y' + z')$   
 $C = 4$  clauses

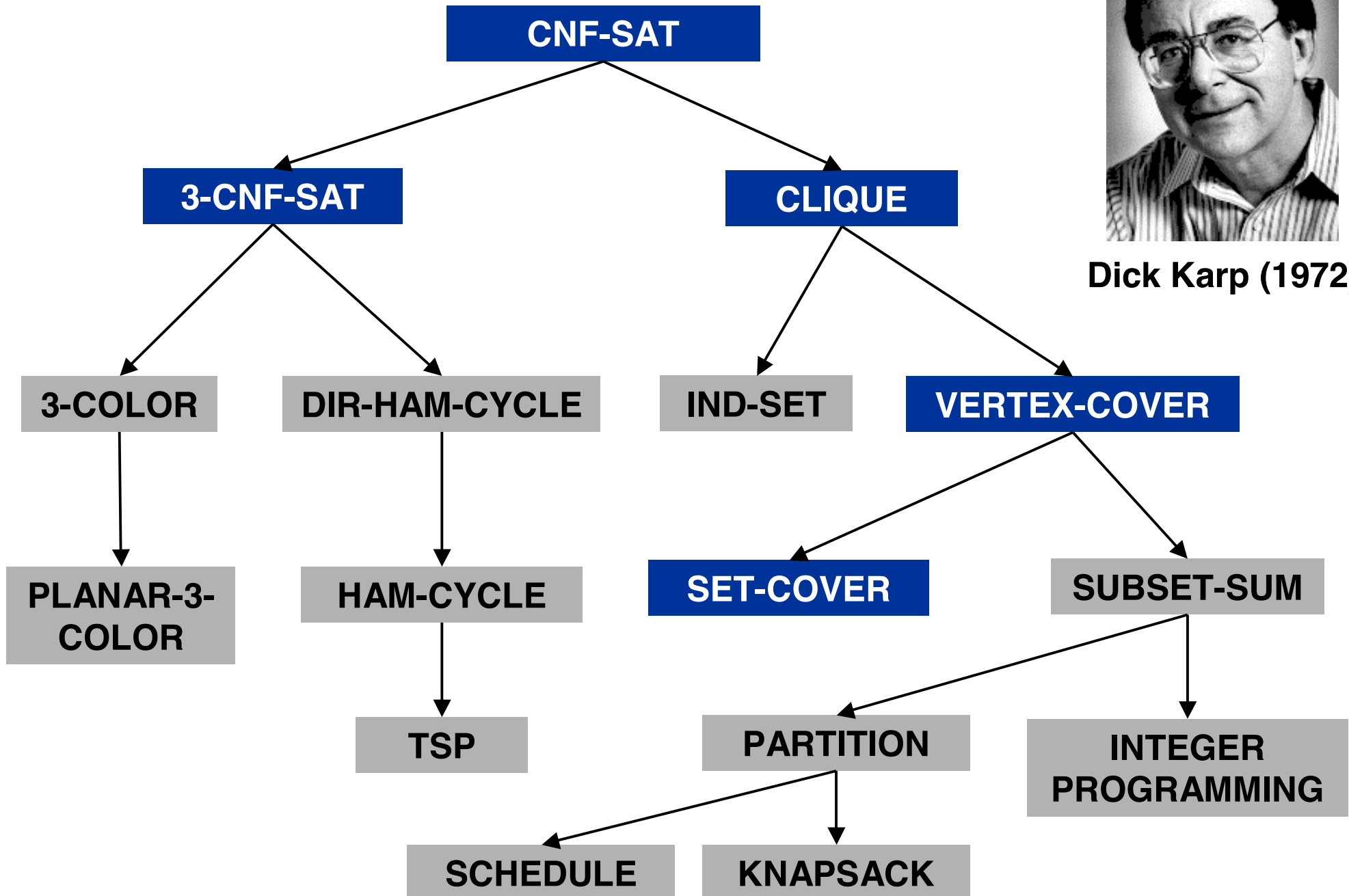




# Polynomial-Time Reductions



Dick Karp (1972)



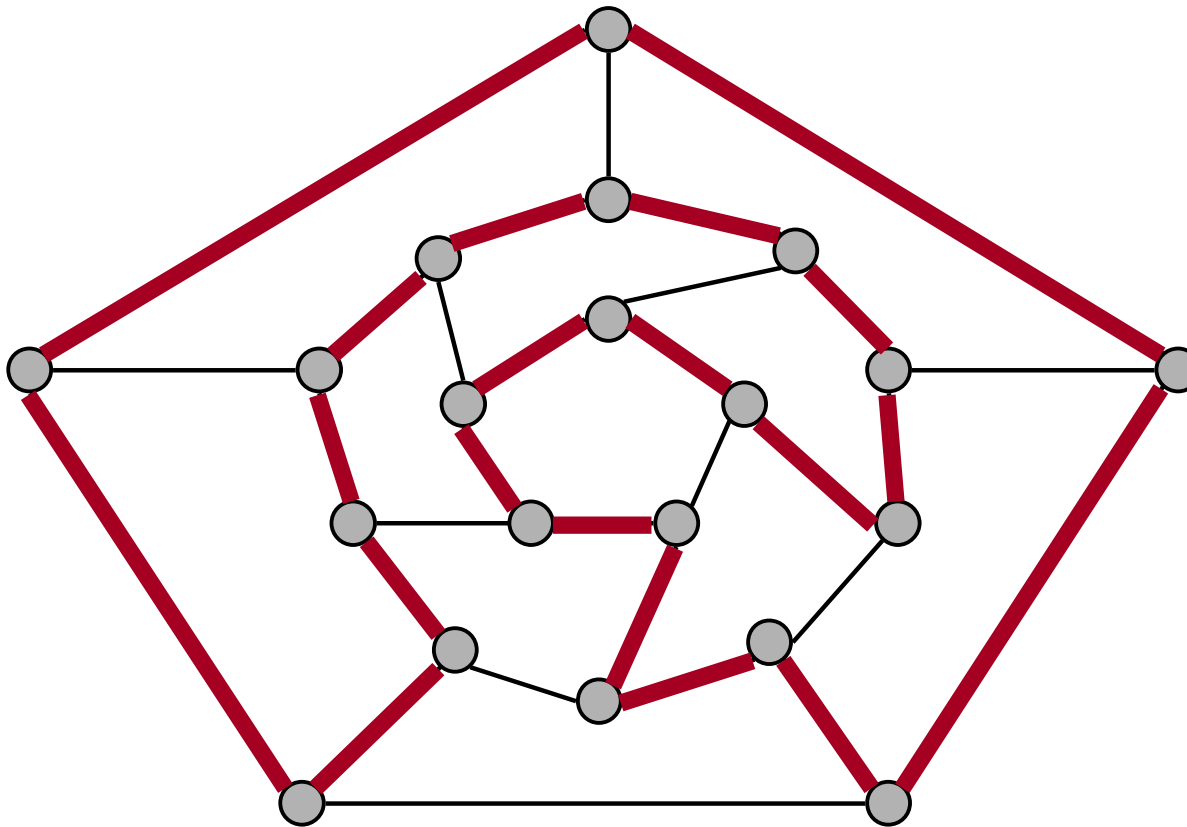
# Problem Genres

## Basic genres.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- **Sequencing problems: HAMILTONIAN-CYCLE, TSP.**
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK, FACTOR.

# Hamiltonian Cycle

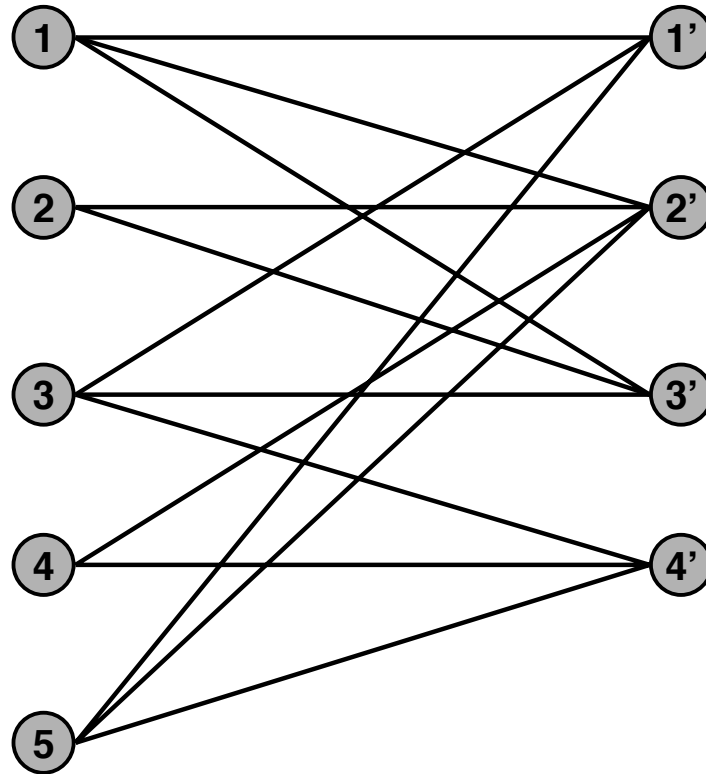
**HAMILTONIAN-CYCLE:** given an undirected graph  $G = (V, E)$ , does there exist a simple cycle  $C$  that contains every vertex in  $V$ .



**YES: vertices and faces of a dodecahedron.**

# Hamiltonian Cycle

**HAMILTONIAN-CYCLE:** given an undirected graph  $G = (V, E)$ , does there exist a simple cycle  $C$  that contains every vertex in  $V$ .



**NO:** bipartite graph with odd number of nodes.

# Finding a Hamiltonian Cycle

**HAM-CYCLE:** given an undirected graph  $G = (V, E)$ , does there exist a simple cycle  $C$  that contains every vertex in  $V$ .

**FIND-HAM-CYCLE:** given an undirected graph  $G = (V, E)$ , output a Hamiltonian cycle if one exists, otherwise output any cycle.

**Claim.**  $\text{HAM-CYCLE} \equiv_p \text{FIND-HAM-CYCLE}$ .

**Proof.**  $\leq_p$

HAM-CYCLE (G)
<pre>C ← FIND-HAM-CYCLE(G) IF (C is Hamiltonian)   RETURN TRUE ELSE   RETURN FALSE</pre>

# Finding a Hamiltonian Cycle

**HAM-CYCLE:** given an undirected graph  $G = (V, E)$ , does there exist a simple cycle  $C$  that contains every vertex in  $V$ .

**FIND-HAM-CYCLE:** given an undirected graph  $G = (V, E)$ , output a Hamiltonian cycle if one exists, otherwise output any cycle.

**Claim.**  $\text{HAM-CYCLE} \equiv_p \text{FIND-HAM-CYCLE}$ .

**Proof.**  $\geq_p$

A son todos los arcos del grafo

```
FIND-HAM-CYCLE (G)  
  
IF (HAM-CYCLE(G) = FALSE)  
  RETURN FALSE  
  
  A  $\leftarrow$  E  
  FOR EACH  $e \in E$   
    IF (HAM-CYCLE(V, A - {e}) = TRUE)  
      A  $\leftarrow$  A - {e}  
  
  RETURN unique cycle remaining in G
```

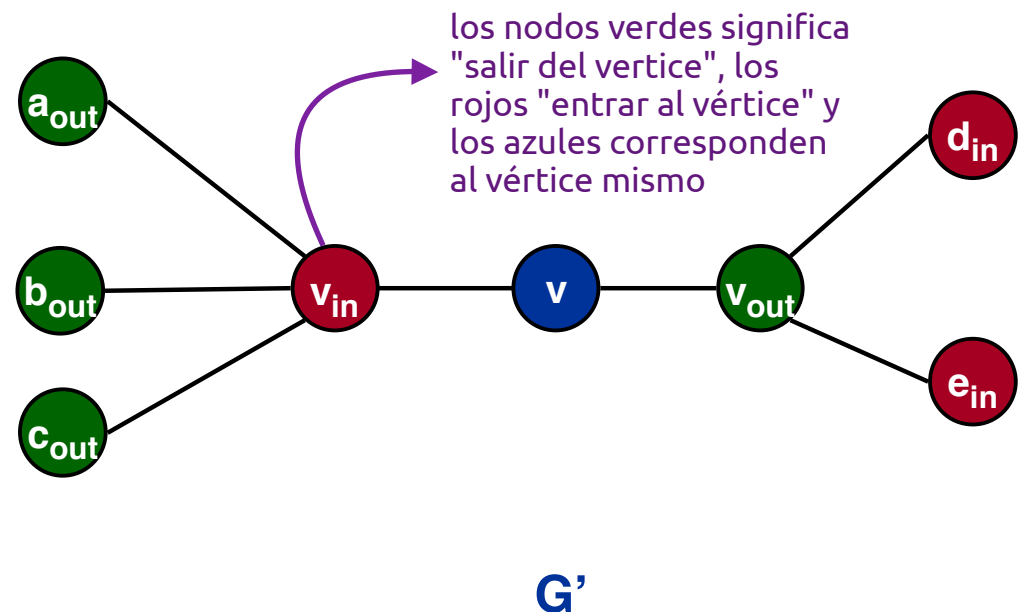
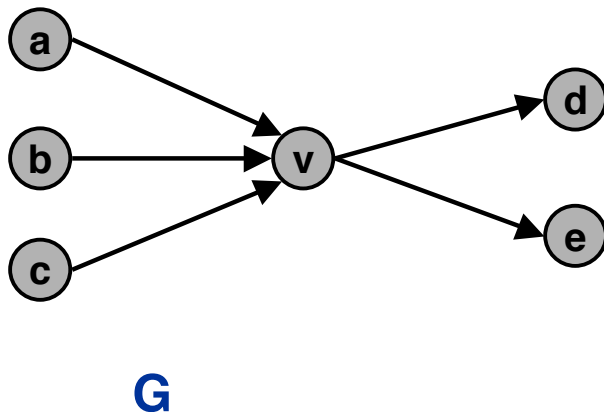
# Directed Hamiltonian Cycle

**DIR-HAM-CYCLE:** given a **directed** graph  $G = (V, E)$ , does there exists a simple directed cycle  $C$  that contains every vertex in  $V$ .

**Claim.**  $\text{DIR-HAM-CYCLE} \leq_p \text{HAM-CYCLE}$ .

**Proof.**

- Given a directed graph  $G = (V, E)$ , construct an undirected graph  $G'$  with  $3n$  vertices.



# Directed Hamiltonian Cycle

**Claim.**  $G$  has a Hamiltonian cycle if and only if  $G'$  does.

**Proof.**  $\Rightarrow$

- Suppose  $G$  has a directed Hamiltonian cycle  $C$ .
- Then  $G'$  has an undirected Hamiltonian cycle.




**Proof.**  $\Leftarrow$

- Suppose  $G'$  has an undirected Hamiltonian cycle  $C'$ .
- $C'$  must visit nodes in  $G'$  using one of following two orders:
  - ... ,  $G, R, B, G, R, B, G, R, B, \dots$
  - ... ,  $R, G, B, R, G, B, R, G, B, \dots$
- Blue nodes in  $C'$  make up directed Hamiltonian cycle  $C$  in  $G$ , or reverse of one.



# 3-SAT Reduces to Directed Hamiltonian Cycle

**Claim.**  $3\text{-CNF-SAT} \leq_p \text{DIR-HAM-CYCLE}$ .

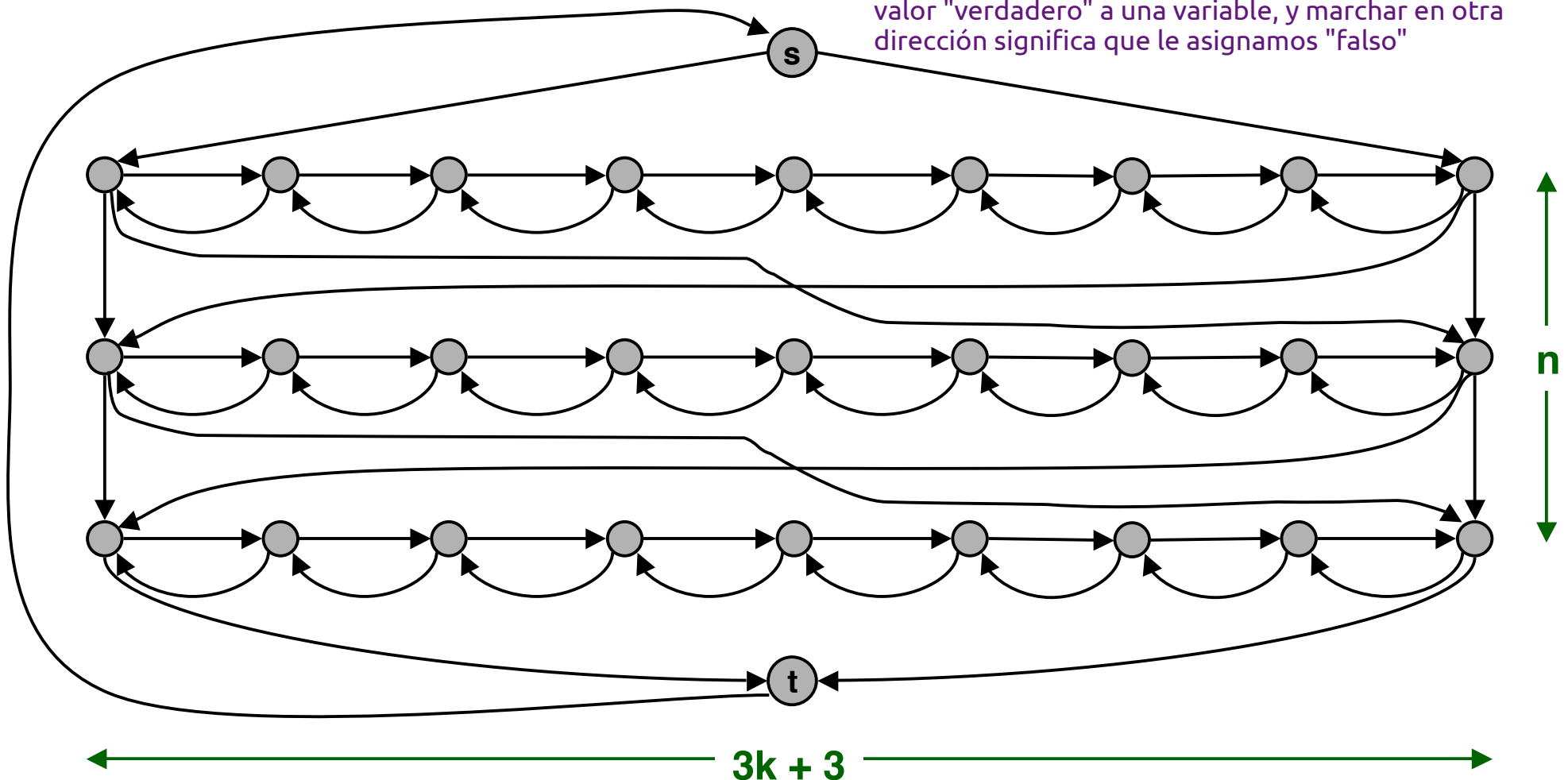
- Why not reduce from some other problem?
  -  Need to find another problem that is sufficiently close.  
(could reduce from VERTEX-COVER)
  -  If don't succeed, start from 3-CNF-SAT since its combinatorial structure is very basic.
  -  Downside: reduction will require certain level of complexity.

# 3-SAT Reduces to Directed Hamiltonian Cycle

**Proof:** Given 3-CNF-SAT instance with  $n$  variables  $x_i$  and  $k$  clauses  $C_j$ .

- Construct  $G$  to have  $2^n$  Hamiltonian cycles.
- Intuition: traverse path  $i$  from left to right  $\Leftrightarrow$  set variable  $x_i = 1$ .

marchar en una dirección significa que le asignamos valor "verdadero" a una variable, y marchar en otra dirección significa que le asignamos "falso"

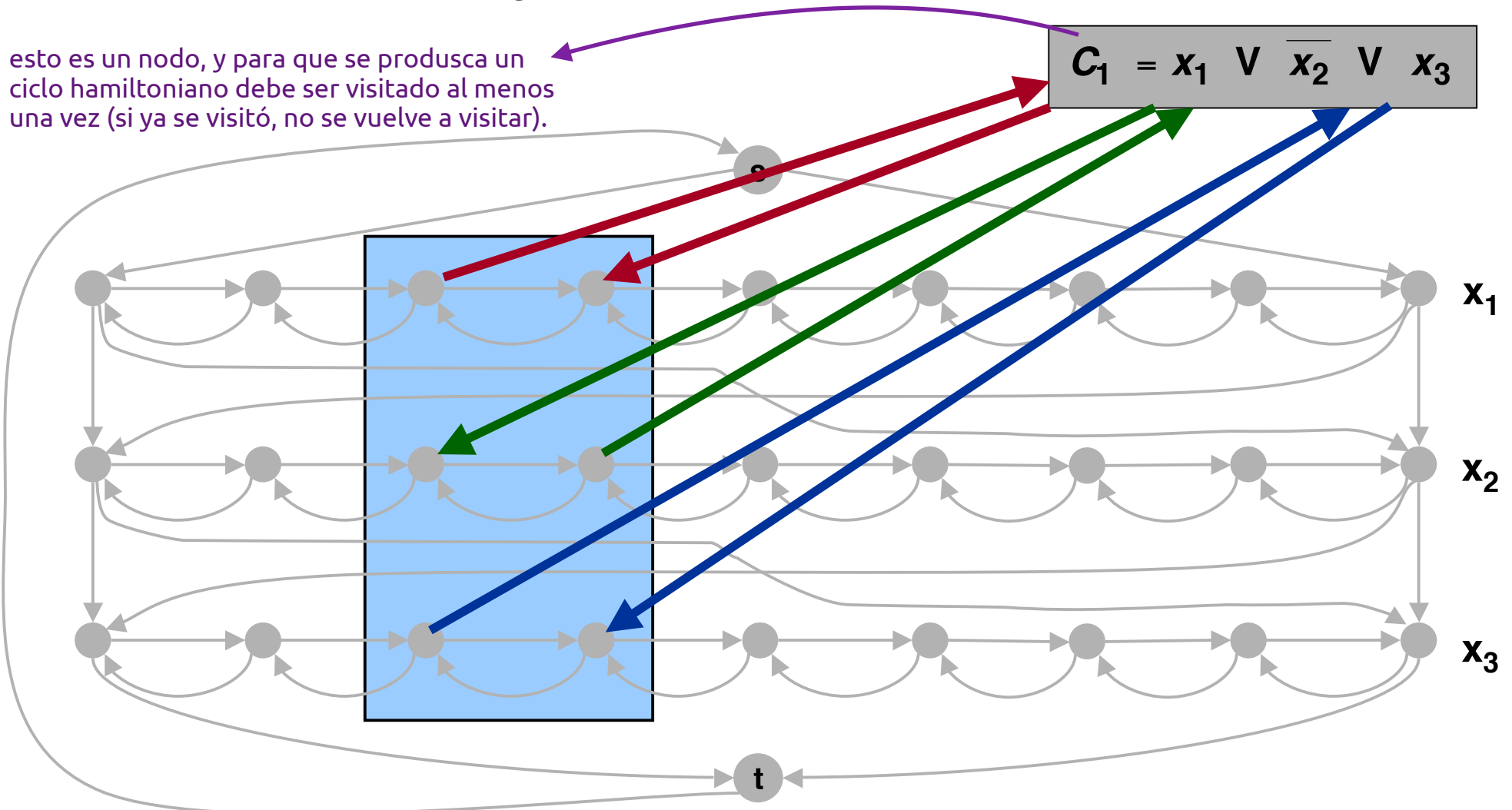


## 3-SAT Reduces to Directed Hamiltonian Cycle

**Proof:** Given 3-CNF-SAT instance with  $n$  variables  $x_i$  and  $k$  clauses  $C_j$ .

- **Add node and 6 edges for each clause.**

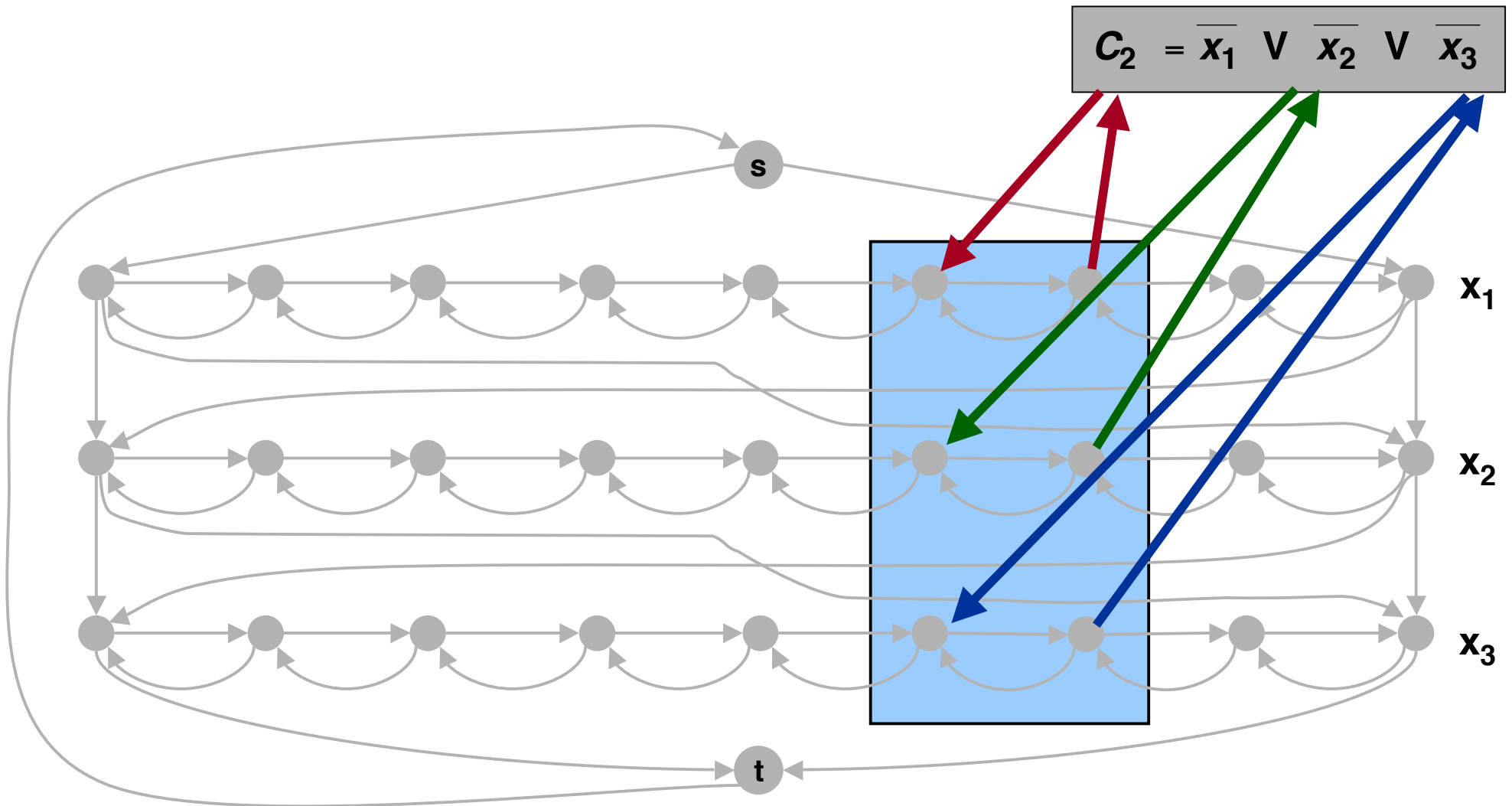
esto es un nodo, y para que se produzca un ciclo hamiltoniano debe ser visitado al menos una vez (si ya se visitó, no se vuelve a visitar).



# 3-SAT Reduces to Directed Hamiltonian Cycle

**Proof:** Given 3-CNF-SAT instance with  $n$  variables  $x_i$  and  $k$  clauses  $C_j$ .

- Add node and 6 edges for each clause.



# 3-SAT Reduces to Directed Hamiltonian Cycle

**Claim.** 3-CNF-SAT instance is satisfiable if and only if corresponding graph  $G$  has a Hamiltonian cycle.

**Proof.**  $\Rightarrow$

- Suppose 3-SAT instance has satisfying assignment  $x^*$ .
- Then, define Hamiltonian cycle in  $G$  as follows:
  - if  $x_i^* = 1$ , traverse path  $P_i$  from left to right
  - if  $x_i^* = 0$ , traverse path  $P_i$  from right to left
  - for each clause  $C_j$ , there will be at least one path  $P_i$  in which we are going in "correct" direction to splice node  $C_j$  into tour

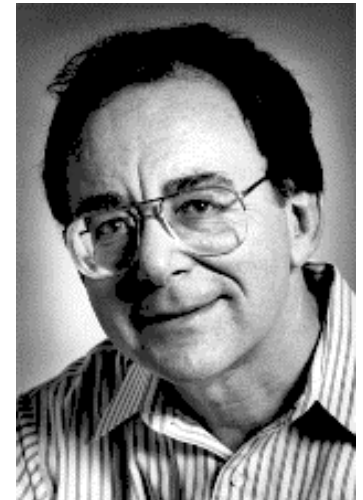
# 3-SAT Reduces to Directed Hamiltonian Cycle

**Claim.** 3-CNF-SAT instance is satisfiable if and only if corresponding graph  $G$  has a Hamiltonian cycle.

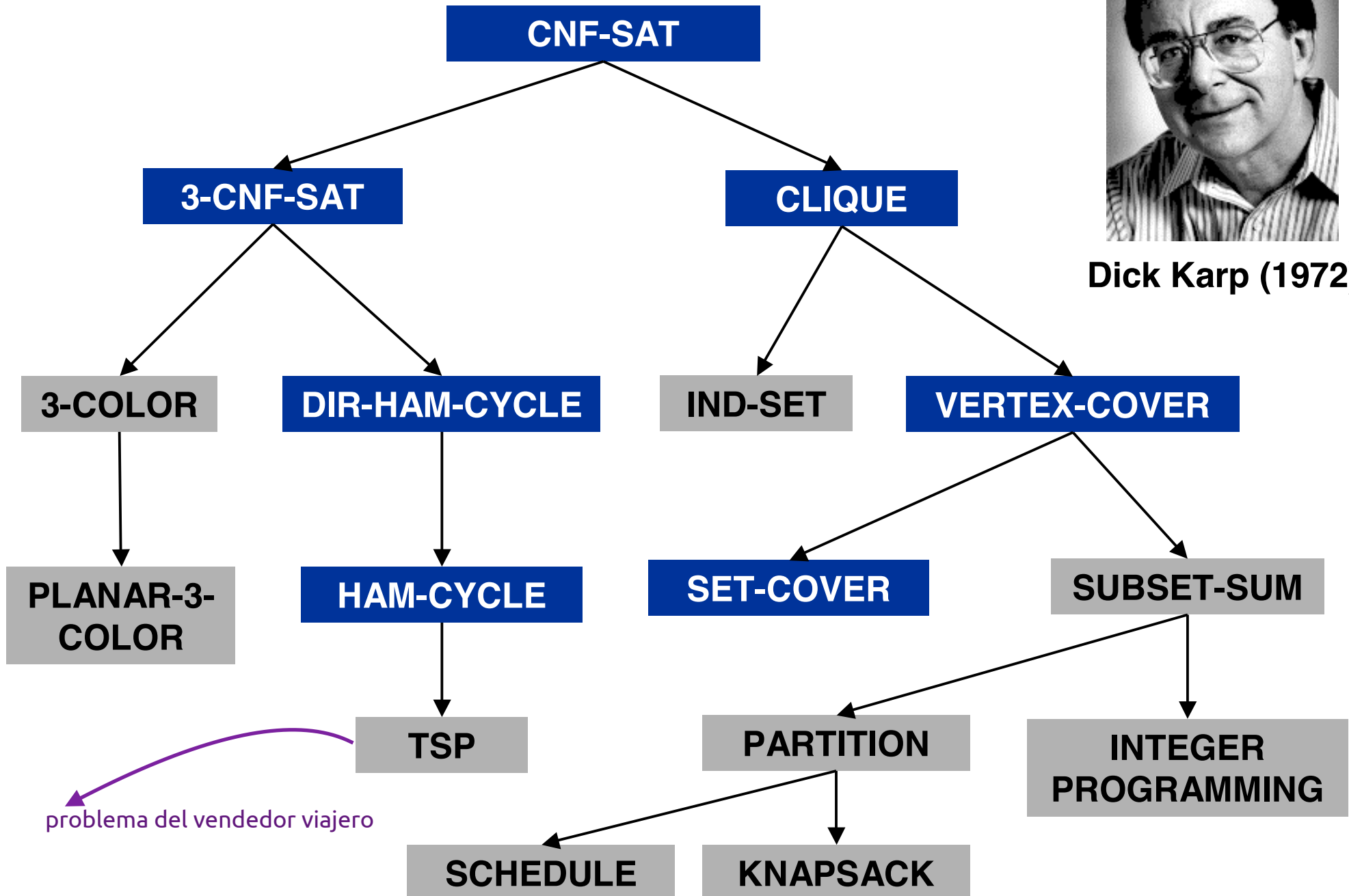
**Proof.**  $\Rightarrow$

- Suppose  $G$  has a Hamiltonian cycle  $C$ .
- If  $C$  enters clause node  $C_j$ , it must depart on mate edge.
  - thus, nodes immediately before and after  $C_j$  are connected by an edge  $e$  in  $G$
  - removing  $C_j$  from cycle, and replacing it with edge  $e$  yields Hamiltonian cycle on  $G - \{C_j\}$
- Continuing in this way, we are left with Hamiltonian cycle  $C'$  in  $G - \{C_1, C_2, \dots, C_k\}$ .
- Set  $x_i^* = 1$  if path  $P_i$  is traversed from left to right, and 0 otherwise.
- Since  $C$  visits each clause node  $C_j$ , at least one of the paths is traversed in "correct" direction, and each clause is satisfied.

# Polynomial-Time Reductions



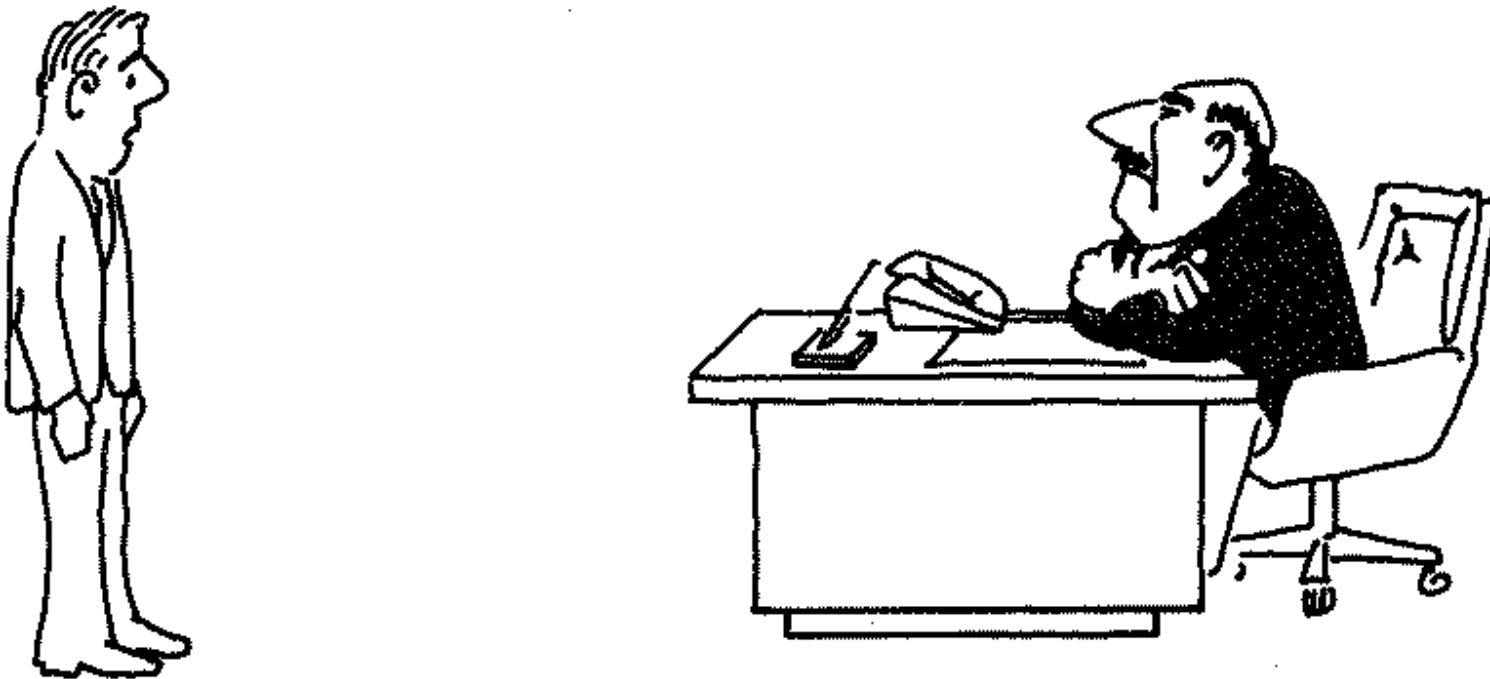
Dick Karp (1972)



problema del vendedor viajero

# Implications Reduction

**Proof that a problem is as hard as CNF-SAT is usually taken as signal to abandon hope of finding an efficient algorithm.**

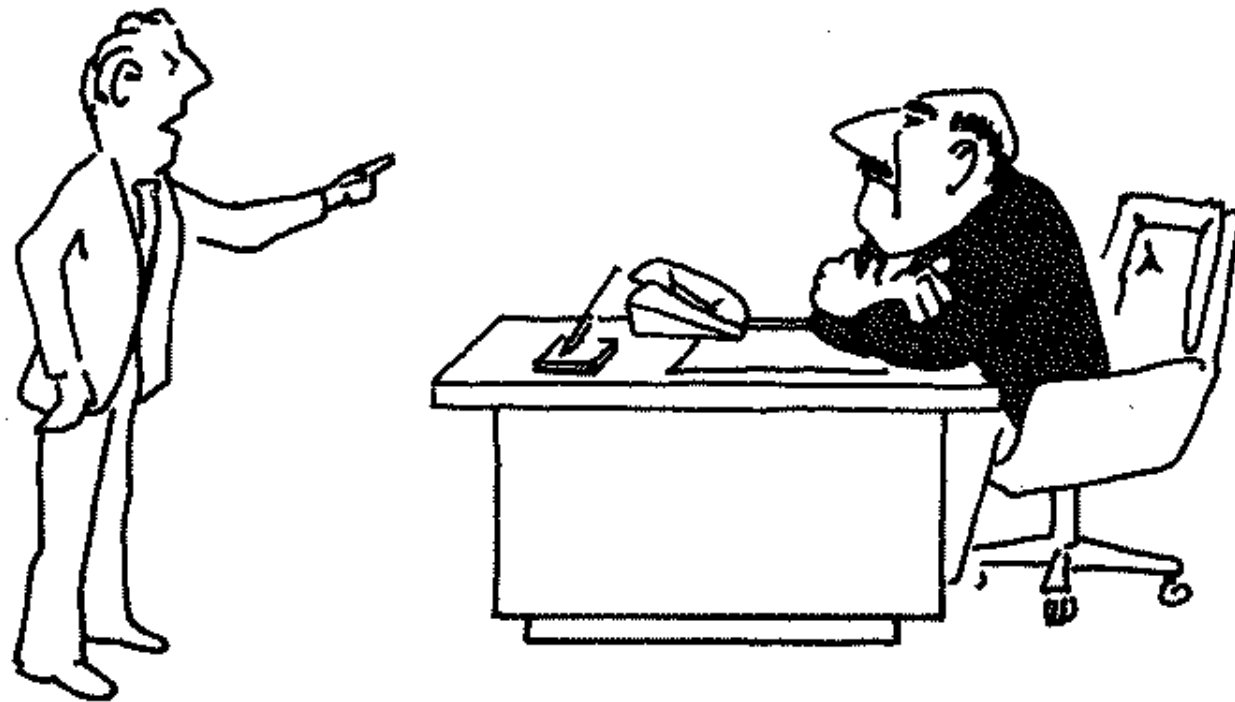


**“I can’t find an efficient algorithm, I guess I’m just too dumb.”**



# Implications Reduction

**Proof that a problem is as hard as CNF-SAT is usually taken as signal to abandon hope of finding an efficient algorithm.**



**“I can’t find an efficient algorithm, because no such algorithm is possible!”**

# Implications Reduction

Proof that a problem is as hard as CNF-SAT is usually taken as signal to abandon hope of finding an efficient algorithm.



“I can’t find an efficient algorithm, but neither can all these famous people.”

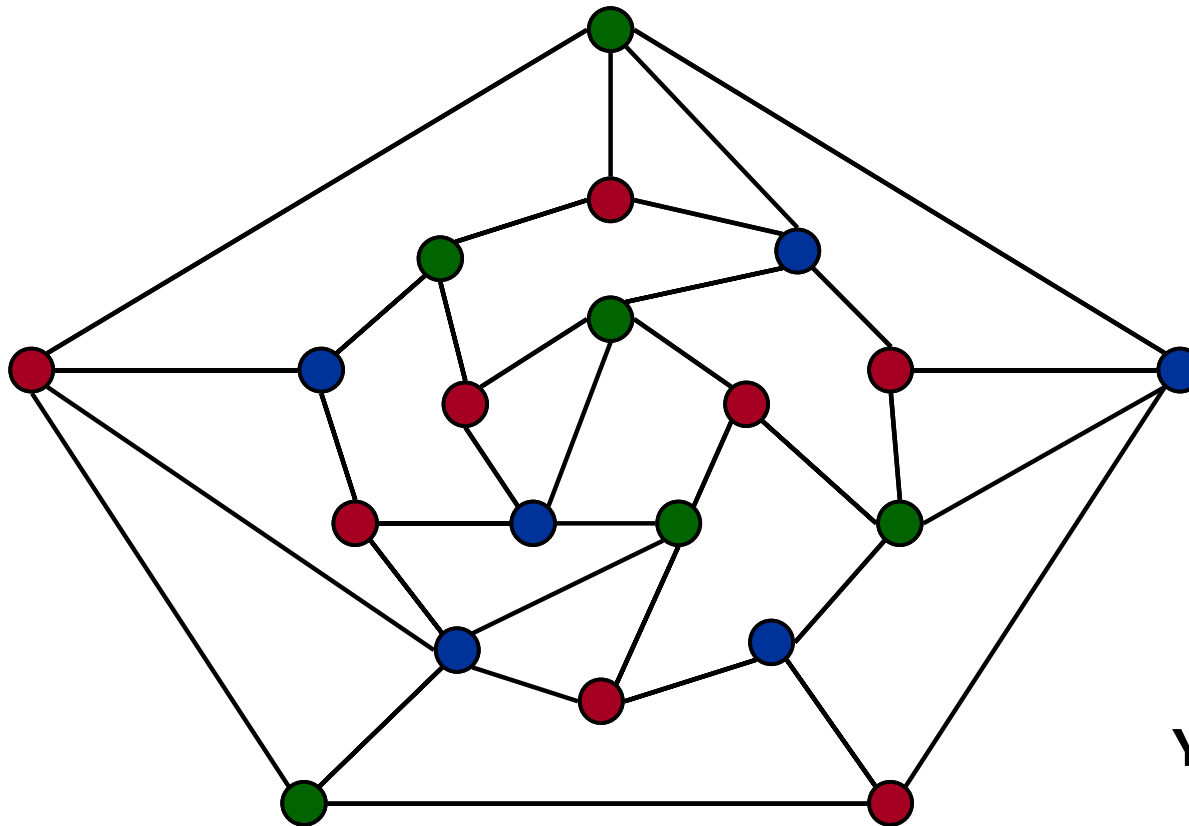
# Problem Genres

## Basic genres.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- **Partitioning problems: 3-COLOR.**
- Numerical problems: SUBSET-SUM, KNAPSACK, FACTOR.

# 3-Colorability

**3-COLOR:** Given an undirected graph does there exist a way to color the nodes R, G, and B such no adjacent nodes have the same color?



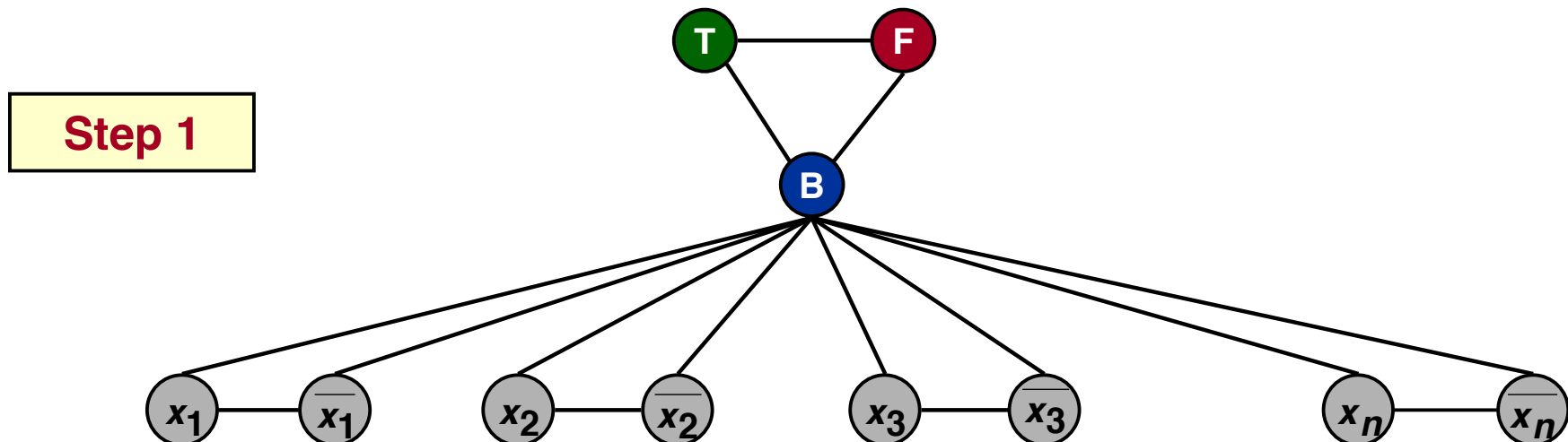
**YES instance**

# 3-Colorability

**Claim.**  $3\text{-CNF-SAT} \leq_p 3\text{-COLOR}$ .

**Proof:** Given 3-SAT instance with  $n$  variables  $x_i$  and  $k$  clauses  $C_j$ .

- Create instance of 3-COLOR  $G = (V, E)$  as follows.
- Step 1:
  - create triangle  $R$  (false),  $G$  (true), or  $B$
  - create nodes for each literal and connect to  $B$ 
    - ✎ Each literal colored  $R$  or  $G$ .
  - create nodes for each literal, and connect literal to its negation
    - ✎ Each literal colored opposite of its negation.

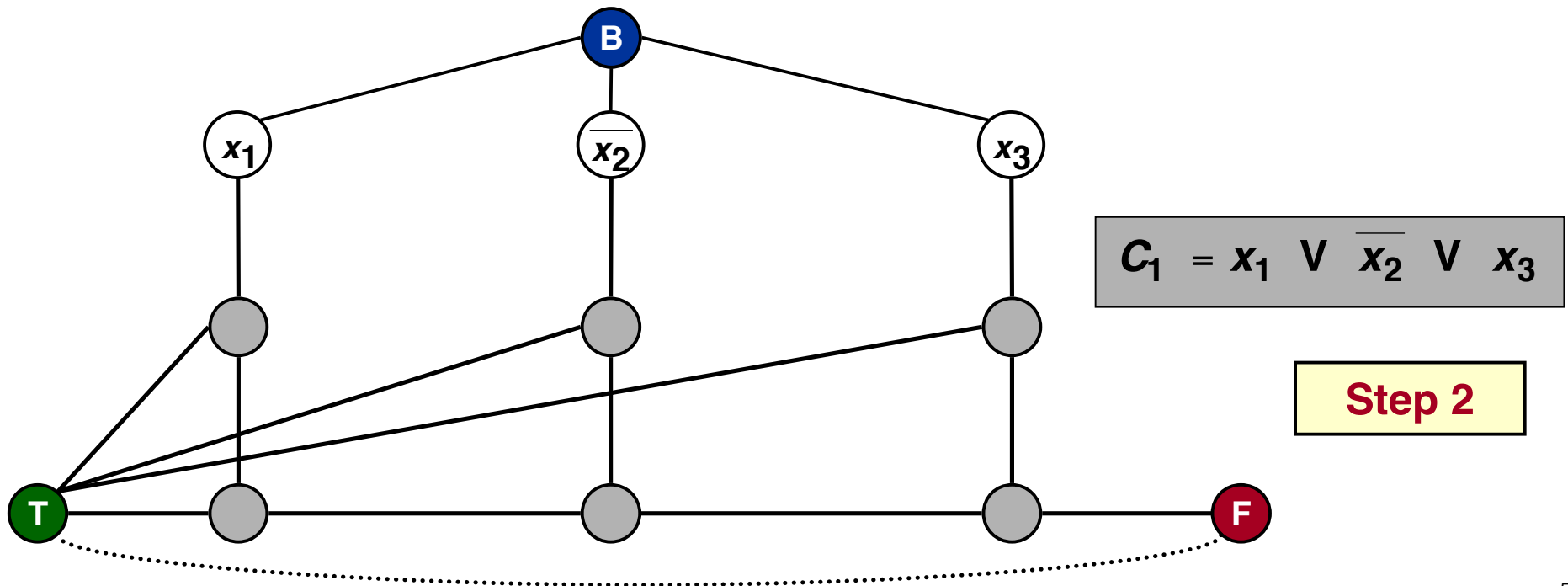


# 3-Colorability

**Claim.**  $3\text{-CNF-SAT} \leq_p 3\text{-COLOR}$ .

**Proof:** Given 3-SAT instance with  $n$  variables  $x_i$  and  $k$  clauses  $C_j$ .

- **Step 2:**
  - for each clause, add "gadget" of 6 new nodes and 13 new edges



# 3-Colorability

**Claim.**  $3\text{-CNF-SAT} \leq_p 3\text{-COLOR}$ .

**Proof:** Given 3-SAT instance with  $n$  variables  $x_i$  and  $k$  clauses  $C_j$ .

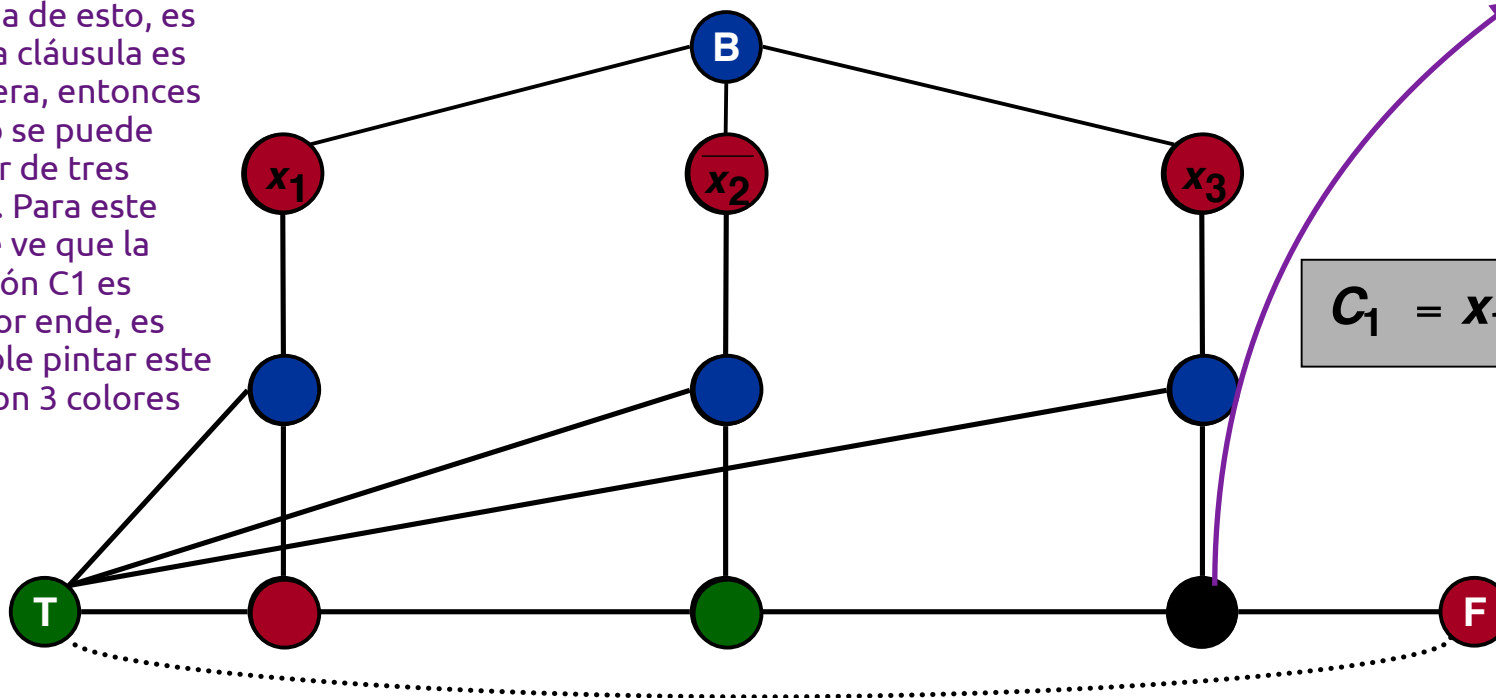
▪ **Step 2:**

- for each clause, add "gadget" of 6 new nodes and 13 new edges
- if 3-colorable, top row must have at least one green (true) node

 Otherwise, middle row all blue.

 Bottom row alternates between green and red  $\Rightarrow$  contradiction.

La gracia de esto, es que si la cláusula es verdadera, entonces el grafo se puede colorear de tres colores. Para este caso, se ve que la disjunción  $C_1$  es falsa, por ende, es imposible pintar este grafo con 3 colores



si los tres literales son falsos, entonces nos va a quedar un nodo con un 4to color, y por ende, no es un grafo de 3 colores

$$C_1 = x_1 \vee \overline{x_2} \vee x_3$$

**Step 2**

# 3-Colorability

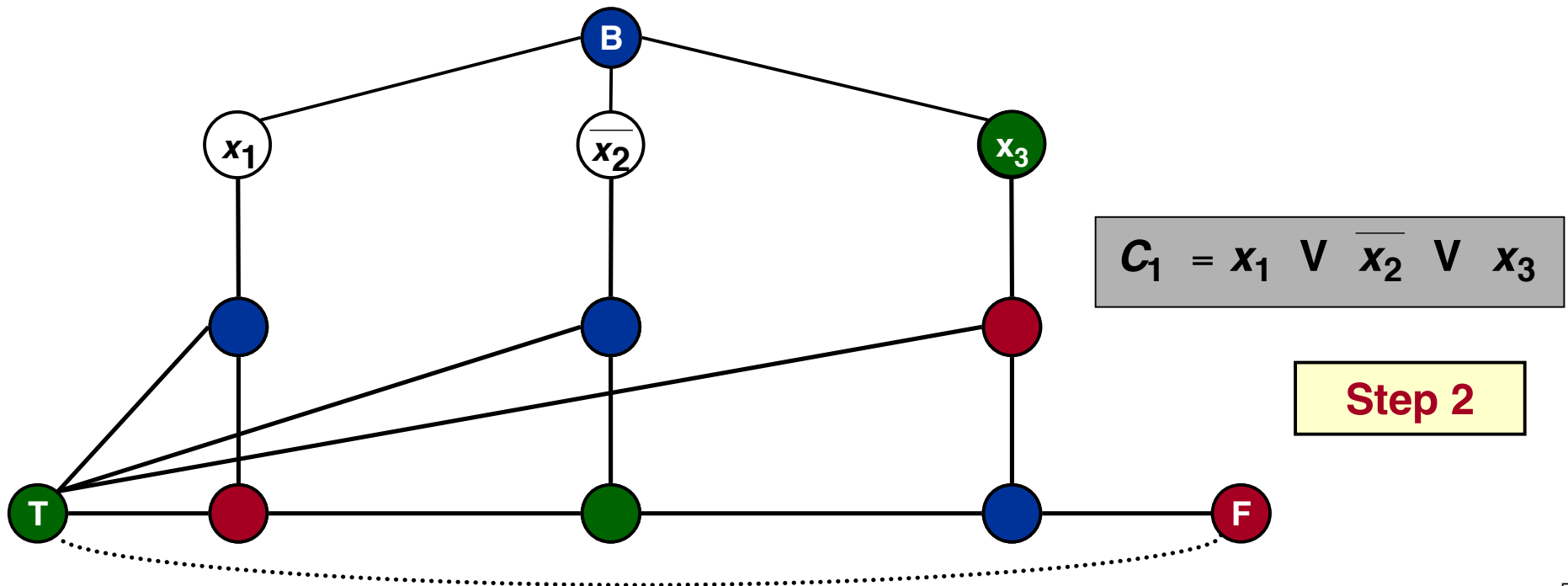
**Claim.**  $3\text{-CNF-SAT} \leq_p 3\text{-COLOR}$ .

**Proof:** Given 3-SAT instance with  $n$  variables  $x_i$  and  $k$  clauses  $C_j$ .

▪ **Step 2:**

- for each clause, add "gadget" of 6 new nodes and 13 new edges
- if top row has green (true) node, then 3-colorable

- ✎ Color vertex below green node red, and one below that blue.
- ✎ Color remaining middle row nodes blue.
- ✎ Color remaining bottom nodes red or green, as forced.

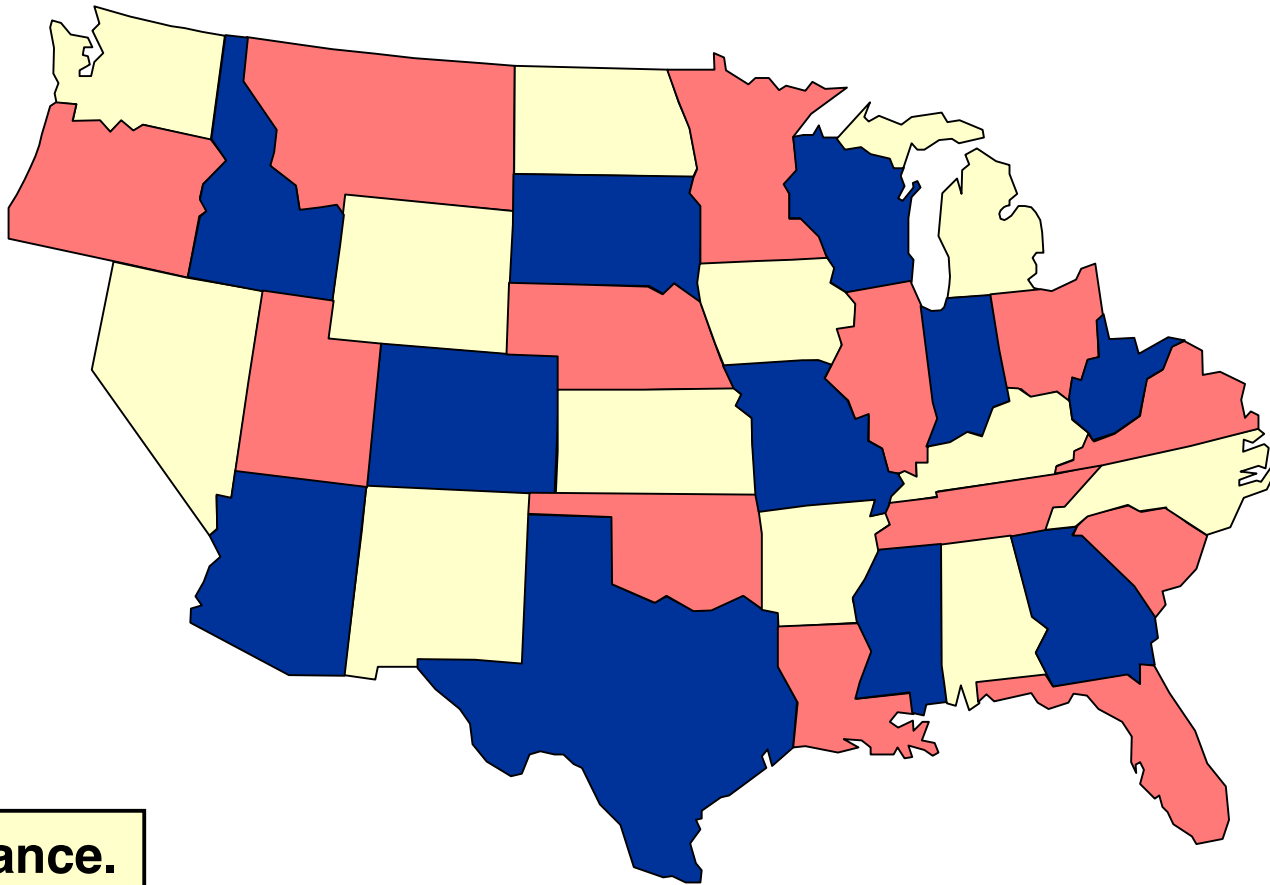




# Planar 3-Colorability

## PLANAR-3-COLOR.

- Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?

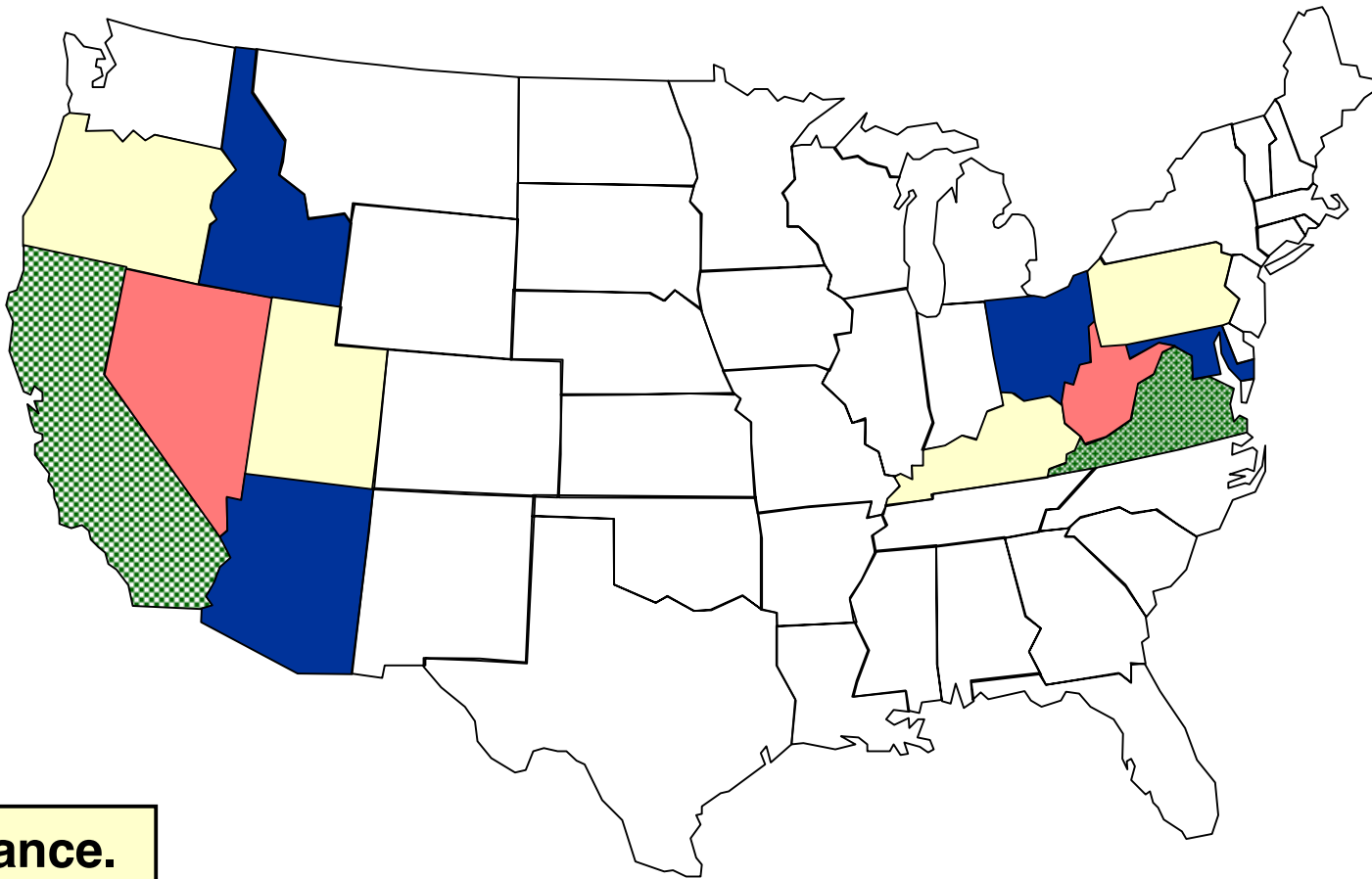


**YES instance.**

# Planar 3-Colorability

## PLANAR-3-COLOR.

- Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?

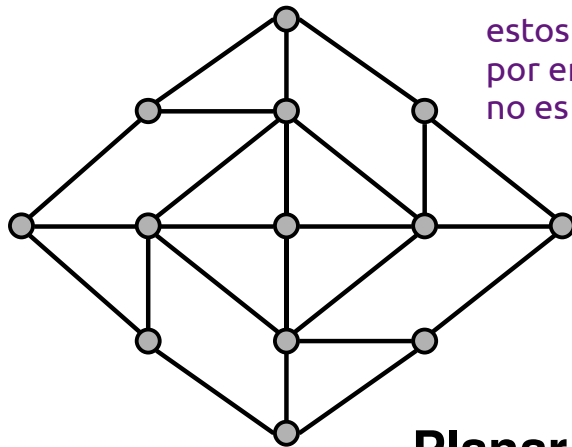


**NO instance.**

# Planarity

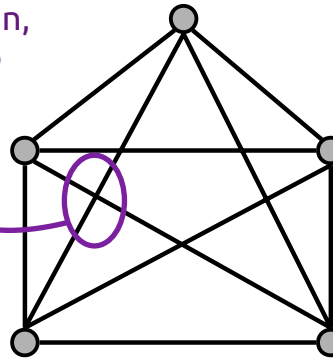
**Planarity.** A graph is planar if it can be embedded on the plane (or sphere) in such a way that no two edges graph.

- Applications: VLSI circuit design, computer graphics.

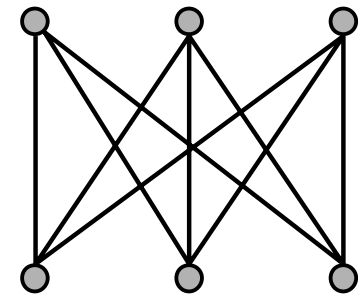


Planar

estos arcos se cruzan,  
por ende este grafo  
no es planar



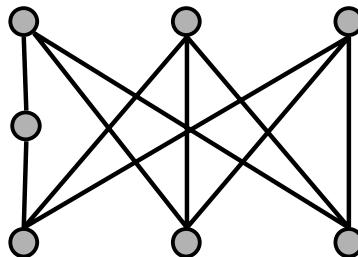
$K_5$ : non-planar



$K_{3,3}$ : non-planar

**Kuratowski's Theorem.** An undirected graph  $G$  is non-planar if and only if it contains a subgraph homeomorphic to  $K_5$  or  $K_{3,3}$ .

Recuerde que un grafo se considera planar, si al momento de dibujarlo (hablamos de un plano en 2 dimensiones), ninguno de sus vértices se cruzan



homeomorphic to  $K_{3,3}$

# Planarity Testing

**Kuratowski's Theorem.** An undirected graph  $G$  is non-planar if and only if it contains a subgraph homeomorphic to  $K_5$  or  $K_{3,3}$ .

**Brute force:**  $O(n^6)$ .

- Step 1. Contract all nodes of degree 2.
- Step 2. Check all subsets of 5 nodes to see if they form a  $K_5$ .
- Step 3. Check all subsets of 6 nodes to see if they form a  $K_{3,3}$ .

**Cleverness:**  $O(n)$ .

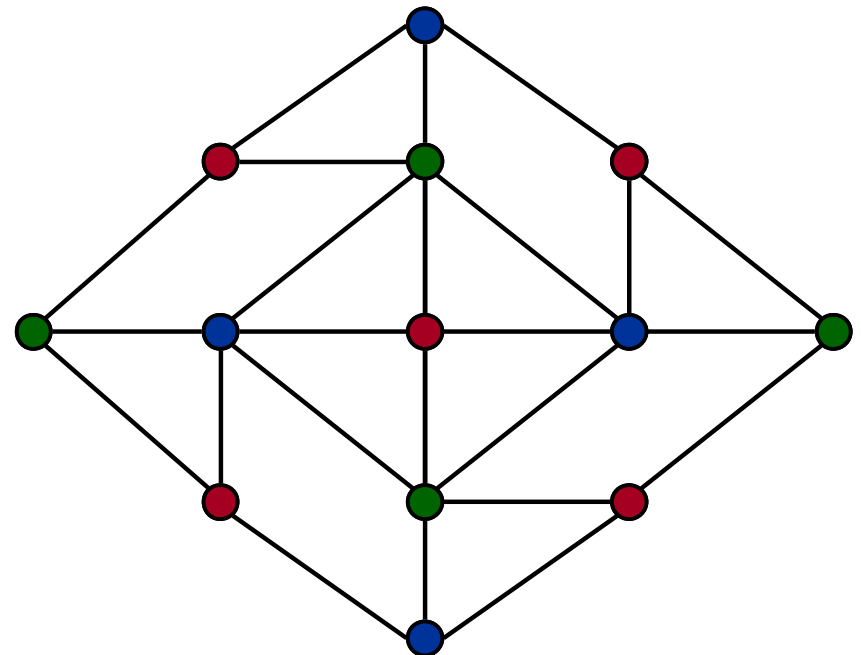
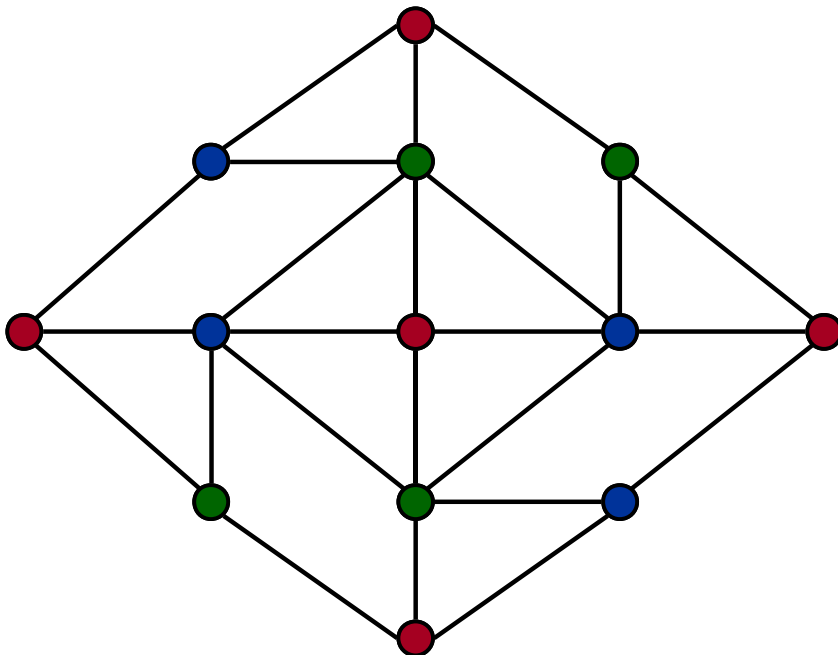
- Step 1. DFS.
- Step 2. Tarjan.

# Planar 3-Colorability

**Claim.**  $3\text{-COLOR} \leq_p \text{PLANAR-3-COLOR}$ .

**Proof sketch:** Given instance of 3-COLOR, draw graph in plane, letting edges cross if necessary.

- Replace each edge crossing with the following planar gadget  $W$ .
  - in any 3-coloring of  $W$ , opposite corners have the same color
  - any assignment of colors to the corners in which opposite corners have the same color extends to a 3-coloring of  $W$



# Planar 4-Colorability

**PLANAR-4-COLOR:** Given a planar map, can it be colored using 4 colors so that no adjacent regions have the same color?

**Intuition.**

- If PLANAR-3-COLOR is hard, then so is PLANAR-4-COLOR and PLANAR-5-COLOR.
- Don't always believe your intuition!

# Planar 4-Colorability

## PLANAR-2-COLOR.

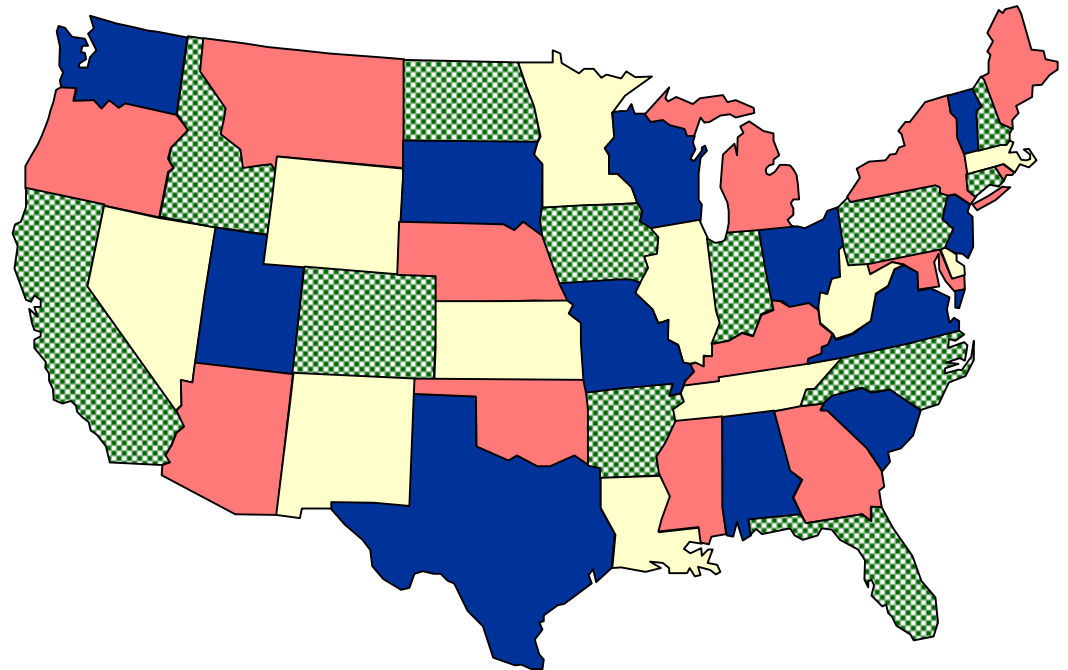
- Solvable in linear time.

## PLANAR-3-COLOR.

- NP-complete.

## PLANAR-4-COLOR.

- Solvable in  $O(1)$  time.



**Theorem (Appel-Haken, 1976).** Every planar map is 4-colorable.

- Resolved century-old open problem.
- Used 50 days of computer time to deal with many special cases.
- First major theorem to be proved using computer.

# Problem Genres

## Basic genres.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING.
- Numerical problems: SUBSET-SUM, KNAPSACK, FACTOR.



# Subset Sum

**SUBSET-SUM:** Given a set  $X$  of integers and a target integer  $t$ , is there a subset  $S \subseteq X$  whose elements sum to exactly  $t$ .

**Example:**  $X = \{1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344\}$ ,  $t = 3754$ .

- YES:  $S = \{1, 16, 64, 256, 1040, 1093, 1284\}$ .

## Remark.

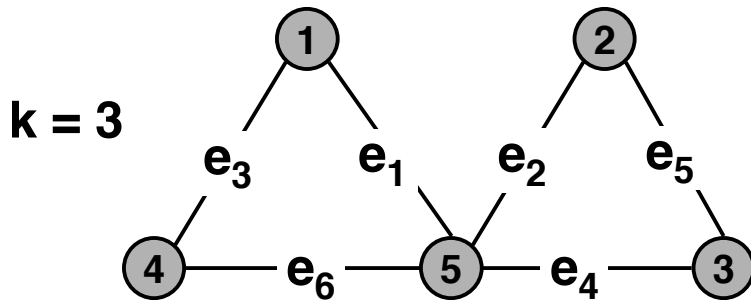
- With arithmetic problems, input integers are encoded in binary.
- Polynomial reduction must be polynomial in binary encoding.

# Subset Sum

Treat as base  $k+1$  integer

**Claim.** VERTEX-COVER  $\leq_p$  SUBSET-SUM.

**Proof.** Given instance  $G, k$  of VERTEX-COVER, create following instance of SUBSET-SUM.



	$e_1$	$e_2$	$e_3$	$e_4$	$e_5$	$e_6$
$v_1$	1	0	1	0	0	0
$v_2$	0	1	0	0	1	0
$v_3$	0	0	0	1	1	0
$v_4$	0	0	1	0	0	1
$v_5$	1	1	0	1	0	1

Node-arc incidence matrix

		$e_1$	$e_2$	$e_3$	$e_4$	$e_5$	$e_6$	decimal
$x_1$	1	1	0	1	0	0	0	5,184
$x_2$	1	0	1	0	0	1	0	4,356
$x_3$	1	0	0	0	1	1	0	4,116
$x_4$	1	0	0	1	0	0	1	4,161
$x_5$	1	1	1	0	1	0	1	5,393
$y_1$	0	1	0	0	0	0	0	1,024
$y_2$	0	0	1	0	0	0	0	256
$y_3$	0	0	0	1	0	0	0	64
$y_4$	0	0	0	0	1	0	0	16
$y_5$	0	0	0	0	0	1	0	4
$y_6$	0	0	0	0	0	0	1	1

$t$	3	2	2	2	2	2	2	15,018
-----	---	---	---	---	---	---	---	--------

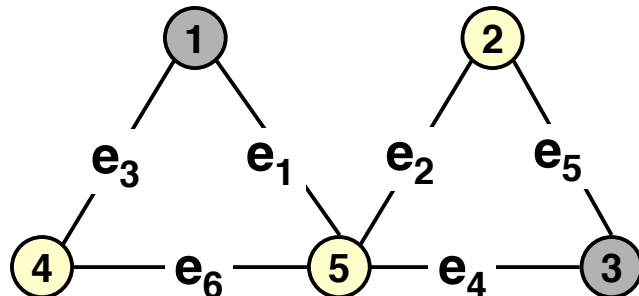
$k$

# Subset Sum

**Claim.**  $G$  has vertex cover of size  $k$  if and only if there is a subset  $S$  that sums to exactly  $t$ .

**Proof.**  $\Rightarrow$

- Suppose  $G$  has a vertex cover  $C$  of size  $k$ .
- Let  $S = C \cup \{y_j : |e_j \cap C| = 1\}$ 
  - most significant bits add up to  $k$
  - remaining bits add up to 2



		$e_1$	$e_2$	$e_3$	$e_4$	$e_5$	$e_6$	decimal
$x_1$	1	1	0	1	0	0	0	5,184
$x_2$	1	0	1	0	0	1	0	4,356
$x_3$	1	0	0	0	1	1	0	4,116
$x_4$	1	0	0	1	0	0	1	4,161
$x_5$	1	1	1	0	1	0	1	5,393
$y_1$	0	1	0	0	0	0	0	1,024
$y_2$	0	0	1	0	0	0	0	256
$y_3$	0	0	0	1	0	0	0	64
$y_4$	0	0	0	0	1	0	0	16
$y_5$	0	0	0	0	0	1	0	4
$y_6$	0	0	0	0	0	0	1	1

$t$	3	2	2	2	2	2	2	15,018
-----	---	---	---	---	---	---	---	--------

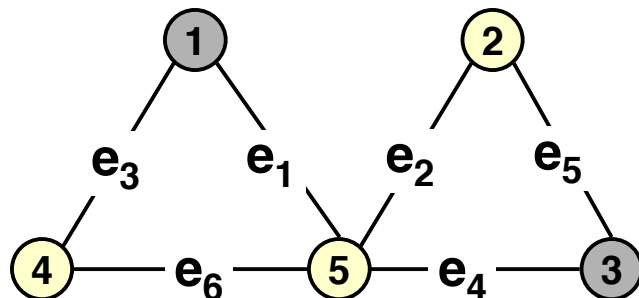
$k$

# Subset Sum

**Claim.**  $G$  has vertex cover of size  $k$  if and only if there is a subset  $S$  that sums to exactly  $t$ .

**Proof.**  $\Leftarrow$

- Suppose subset  $S$  sums to  $t$ .
- Let  $C = S \cap \{x_1, \dots, x_n\}$ .
  - each edge has three 1's, so no carries possible
  - $|C| = k$
  - at least one  $x_i$  must contribute to sum for  $e_j$



		$e_1$	$e_2$	$e_3$	$e_4$	$e_5$	$e_6$	decimal
$x_1$	1	1	0	1	0	0	0	5,184
$x_2$	1	0	1	0	0	1	0	4,356
$x_3$	1	0	0	0	1	1	0	4,116
$x_4$	1	0	0	1	0	0	1	4,161
$x_5$	1	1	1	0	1	0	1	5,393
$y_1$	0	1	0	0	0	0	0	1,024
$y_2$	0	0	1	0	0	0	0	256
$y_3$	0	0	0	1	0	0	0	64
$y_4$	0	0	0	0	1	0	0	16
$y_5$	0	0	0	0	0	1	0	4
$y_6$	0	0	0	0	0	0	1	1

$t$	3	2	2	2	2	2	2	15,018
-----	---	---	---	---	---	---	---	--------

$k$

# Partition




**SUBSET-SUM:** Given a set  $X$  of integers and a target integer  $t$ , is there a subset  $S \subseteq X$  whose elements sum to exactly  $t$ .

**PARTITION:** Given a set  $X$  of integers, is there a subset  $S \subseteq X$  such

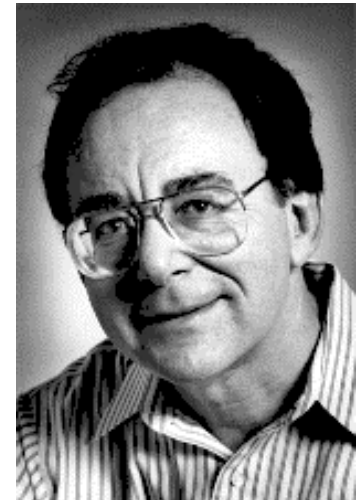
that  $\sum_{a \in S} a = \sum_{a \in X \setminus S} a$ . Una propiedad  $P$  de un programa (para el teorema de Rice) puede ser que  $P$  imprime "Hola mundo", suma dos variables, etc.

**Claim.**  $\text{SUBSET-SUM} \leq_p \text{PARTITION}$ .

**Proof.** Let  $(X, t)$  be an instance of SUBSET-SUM.

- Define  $W$  to be sum of integers in  $X$ :  $W = \sum_{a \in X} a$ .
- Create instance of PARTITION:  $X' = X \cup \{2W - t\} \cup \{W + t\}$ .
- SUBSET-SUM instance is yes if and only if PARTITION instance is.
  - in any partition of  $X'$ 
    -  Each half of partition sums to  $2W$ .
    -  Two new elements can't be in same partition.
    -  Discard new elements  $\Rightarrow$  subset of  $X$  that sums to  $t$ .

# Polynomial-Time Reductions



Dick Karp (1972)

