Reducciones para NP-completitud



UNIVERSIDAD DE CHILE

Arturo Merino

Plan

Recuerdo y plan

- **Recuerdo:**

Recuerdo y plan

- **Recuerdo:**
  - NP-completitud: NP-díficil + NP

Recuerdo y plan

- **Recuerdo:**
  - NP-completitud: NP-díficil + NP
    - Cook–Levin: SAT es NP-completo.

Recuerdo y plan

- **Recuerdo:**
  - NP-completitud: NP-díficil + NP
    - Cook–Levin: SAT es NP-completo.
  - Para probar que B es NP-completo:

Recuerdo y plan

- **Recuerdo:**
  - NP-completitud: NP-díficil $+$ NP
    - Cook–Levin: SAT es NP-completo.
  - Para probar que B es NP-completo:
    - $B \in$ NP y $A \leq_{\mathsf{P}} B$ para algún A NP-completo.

Recuerdo y plan

- **Recuerdo:**
  - NP-completitud: NP-díficil + NP
    - Cook–Levin: SAT es NP-completo.
  - Para probar que B es NP-completo:
    - $B \in$ NP y $A \leq_{\mathsf{P}} B$ para algún A NP-completo.
    - f polinomial y $w \in A \iff f(w) \in B$.

Recuerdo y plan

- **Recuerdo:**
  - NP-completitud: NP-díficil + NP
    - Cook–Levin: SAT es NP-completo.
  - Para probar que B es NP-completo:
    - $B \in$ NP y $A \leq_{\mathsf{P}} B$ para algún A NP-completo.
      - f polinomial y $w \in A \iff f(w) \in B$.
- Si A es NP-completo y creemos $P \neq NP$

Recuerdo y plan

- **Recuerdo:**
  - NP-completitud: NP-díficil + NP
    - Cook–Levin: SAT es NP-completo.
  - Para probar que B es NP-completo:
    - $B \in$ NP y $A \leq_{\mathsf{P}} B$ para algún A NP-completo.
    - f polinomial y $w \in A \iff f(w) \in B$.
- Si A es NP-completo y creemos $P \neq NP$
  - Creemos que no hay algoritmo polinomial para A.

Recuerdo y plan

- **Recuerdo:**
  - NP-completitud: NP-díficil + NP
    - Cook–Levin: SAT es NP-completo.
  - Para probar que B es NP-completo:
    - $B \in$ NP y $A \leq_{\mathsf{P}} B$ para algún A NP-completo.
      - f polinomial y $w \in A \iff f(w) \in B$.
  - Si A es NP-completo y creemos $P \neq NP$
    - Creemos que no hay algoritmo polinomial para A.
- **Plan:**

Recuerdo y plan

- **Recuerdo:**
  - NP-completitud: NP-díficil + NP
    - Cook–Levin: SAT es NP-completo.
  - Para probar que B es NP-completo:
    - $B \in$ NP y $A \leq_{\mathsf{P}} B$ para algún A NP-completo.
      - f polinomial y $w \in A \iff f(w) \in B$.
  - Si A es NP-completo y creemos $P \neq NP$
    - Creemos que no hay algoritmo polinomial para A.
- **Plan:**
  - NP-completitud via reducciones

NP-Completitud

SAT

- SAT con estructura adicional.

SAT

- SAT con estructura adicional.

Def.

Una fórmula está en CNF si es una conjunción de disyunciones de literales.

SAT

- SAT con estructura adicional.

Def.

Una fórmula está en CNF si es una conjunción de disyunciones de literales.

- Obtuvimos eso de Cook-Levin

SAT

- SAT con estructura adicional.

Def.

Una fórmula está en CNF si es una conjunción de disyunciones de literales.

- Obtuvimos eso de Cook-Levin

Def.

Input: Una fórmula en *CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$.

SAT

- SAT con estructura adicional.

Def.

Una fórmula está en CNF si es una conjunción de disyunciones de literales.

- Obtuvimos eso de Cook-Levin

Def.

Input: Una fórmula en *CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$.

Output: ¿Hay una asignación que satisfaga φ?

SAT

- SAT con estructura adicional.

Def.

Una fórmula está en CNF si es una conjunción de disyunciones de literales.

- Obtuvimos eso de Cook-Levin

Def.

Input: Una fórmula en *CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$.

Output: ¿Hay una asignación que satisfaga φ?

$$\text{SAT-CNF} = \{\langle \varphi \rangle : \varphi \text{ es una CNF satisfacible}\}.$$

SAT

- SAT con estructura adicional.

Def.

Una fórmula está en CNF si es una conjunción de disyunciones de literales.

- Obtuvimos eso de Cook-Levin

Def.

Input: Una fórmula en *CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$.

Output: ¿Hay una asignación que satisfaga φ?

$$\text{SAT-CNF} = \{\langle \varphi \rangle : \varphi \text{ es una CNF satisfacible}\}.$$

Teo.

SAT-CNF es NP-completo.

3SAT

- SAT con estructura adicional.

3SAT

- SAT con estructura adicional.

Def.

Input: Una fórmula en *3CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$, con tres literales de variables distintas por cláusula.

3SAT

- SAT con estructura adicional.

Def.

Input: Una fórmula en *3CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$, con tres literales de variables distintas por cláusula.

Output: ¿Hay una asignación que satisfaga φ?

3SAT

- SAT con estructura adicional.

Def.

Input: Una fórmula en *3CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$, con tres literales de variables distintas por cláusula.

Output: ¿Hay una asignación que satisfaga φ?

$$3\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es una 3CNF satisfacible}\}.$$

3SAT

- SAT con estructura adicional.

Def.

Input: Una fórmula en *3CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$, con tres literales de variables distintas por cláusula.

Output: ¿Hay una asignación que satisfaga φ?

$$3\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es una 3CNF satisfacible}\}.$$

Teo.

$3\mathrm{SAT}$ es NP-completo.

3SAT

- SAT con estructura adicional.

Def.

Input: Una fórmula en *3CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$, con tres literales de variables distintas por cláusula.

Output: ¿Hay una asignación que satisfaga φ?

$$3\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es una 3CNF satisfacible}\}.$$

Teo.

$3\mathrm{SAT}$ es NP-completo.

- ¿En NP?

3SAT

- SAT con estructura adicional.

Def.

Input: Una fórmula en *3CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$, con tres literales de variables distintas por cláusula.

Output: ¿Hay una asignación que satisfaga φ?

$$3\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es una 3CNF satisfacible}\}.$$

Teo.

$3\mathrm{SAT}$ es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

3SAT

- SAT con estructura adicional.

Def.

Input: Una fórmula en *3CNF*, $\varphi = C_1 \wedge \cdots \wedge C_m$, con tres literales de variables distintas por cláusula.

Output: ¿Hay una asignación que satisfaga φ?

$$3\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es una 3CNF satisfacible}\}.$$

Teo.

$3\mathrm{SAT}$ es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De SAT a $3\mathrm{SAT}$.

Independent Set

- Elegir sin entrar en conflicto

Independent Set

- Elegir sin entrar en conflicto

Def.

Input: Un grafo G y un número k.

Independent Set

- Elegir sin entrar en conflicto

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices sin aristas entre ellos?

Independent Set

- Elegir sin entrar en conflicto

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices sin aristas entre ellos?

$$\mathrm{IS} = \{\langle G, k\rangle : G \text{ tiene un conjunto independiente de tamaño } \geq k\}.$$

Independent Set

- Elegir sin entrar en conflicto

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices sin aristas entre ellos?

$$\mathrm{IS} = \{\langle G, k\rangle : G \text{ tiene un conjunto independiente de tamaño } \geq k\}.$$

Teo.

IS es NP-completo.

Independent Set

- Elegir sin entrar en conflicto

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices sin aristas entre ellos?

$$\mathrm{IS} = \{\langle G, k\rangle : G \text{ tiene un conjunto independiente de tamaño } \geq k\}.$$

Teo.

IS es NP-completo.

- ¿En NP?

Independent Set

- Elegir sin entrar en conflicto

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices sin aristas entre ellos?

$\mathrm{IS} = \{\langle G, k\rangle : G \text{ tiene un conjunto independiente de tamaño } \geq k\}$.

Teo.

IS es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

Independent Set

- Elegir sin entrar en conflicto

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices sin aristas entre ellos?

$$\mathrm{IS} = \{\langle G, k\rangle : G \text{ tiene un conjunto independiente de tamaño } \geq k\}.$$

Teo.

IS es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De $3\mathrm{SAT}$ a IS.

Clique

- Encontrar comunidades

Clique

- Encontrar comunidades

Def.

Input: Un grafo G y un número k.

Clique

- Encontrar comunidades

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices todos adyacentes entre sí?

Clique

- Encontrar comunidades

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices todos adyacentes entre sí?

$$\text{CLIQUE} = \{\langle G, k\rangle : G \text{ tiene un clique de tamaño } \geq k\}.$$

Clique

- Encontrar comunidades

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices todos adyacentes entre sí?

$$\mathrm{CLIQUE} = \{\langle G, k\rangle : G \text{ tiene un clique de tamaño } \geq k\}.$$

Teo.

CLIQUE es NP-completo.

Clique

- Encontrar comunidades

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices todos adyacentes entre sí?

$$\mathrm{CLIQUE} = \{\langle G, k\rangle : G \text{ tiene un clique de tamaño } \geq k\}.$$

Teo.

CLIQUE es NP-completo.

- ¿En NP?

Clique

- Encontrar comunidades

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices todos adyacentes entre sí?

$$\mathrm{CLIQUE} = \{\langle G, k\rangle : G \text{ tiene un clique de tamaño } \geq k\}.$$

Teo.

CLIQUE es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

Clique

- Encontrar comunidades

Def.

Input: Un grafo G y un número k.

Output: ¿Hay al menos k vértices todos adyacentes entre sí?

$$\mathrm{CLIQUE} = \{\langle G, k\rangle : G \text{ tiene un clique de tamaño } \geq k\}.$$

Teo.

CLIQUE es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De IS a CLIQUE.

Vertex Cover

- Poner controles para vigilar conexiones.

Vertex Cover

- Poner controles para vigilar conexiones.

Def.

Input: Un grafo G y un número k.

Vertex Cover

- Poner controles para vigilar conexiones.

Def.

Input: Un grafo G y un número k.

Output: ¿Hay a lo más k vértices que toquen todas las aristas?

Vertex Cover

- Poner controles para vigilar conexiones.

Def.

Input: Un grafo G y un número k.

Output: ¿Hay a lo más k vértices que toquen todas las aristas?

$\mathrm{VC} = \{\langle G, k\rangle : G \text{ tiene un cubrimiento por vértices de tamaño } \leq k\}$.

Vertex Cover

- Poner controles para vigilar conexiones.

Def.

Input: Un grafo G y un número k.

Output: ¿Hay a lo más k vértices que toquen todas las aristas?

$\mathrm{VC} = \{\langle G, k\rangle : G \text{ tiene un cubrimiento por vértices de tamaño } \leq k\}$.

Teo.

VC es NP-completo.

Vertex Cover

- Poner controles para vigilar conexiones.

Def.

Input: Un grafo G y un número k.

Output: ¿Hay a lo más k vértices que toquen todas las aristas?

$\mathrm{VC} = \{\langle G, k\rangle : G \text{ tiene un cubrimiento por vértices de tamaño } \leq k\}$.

Teo.

VC es NP-completo.

- ¿En NP?

Vertex Cover

- Poner controles para vigilar conexiones.

Def.

Input: Un grafo G y un número k.

Output: ¿Hay a lo más k vértices que toquen todas las aristas?

$\mathrm{VC} = \{\langle G, k\rangle : G \text{ tiene un cubrimiento por vértices de tamaño } \leq k\}$.

Teo.

VC es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

Vertex Cover

- Poner controles para vigilar conexiones.

Def.

Input: Un grafo G y un número k.

Output: ¿Hay a lo más k vértices que toquen todas las aristas?

$\mathrm{VC} = \{\langle G, k\rangle : G \text{ tiene un cubrimiento por vértices de tamaño } \leq k\}$.

Teo.

VC es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De IS a VC.

Set Cover

- Satisfacer requerimientos con tareas

Set Cover

- Satisfacer requerimientos con tareas

Def.

Input: Un conjunto U, una colección $\mathcal{S}$ de subconjuntos de U y un número k.

Set Cover

- Satisfacer requerimientos con tareas

Def.

Input: Un conjunto U, una colección $\mathcal{S}$ de subconjuntos de U y un número k.

Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

Set Cover

- Satisfacer requerimientos con tareas

Def.

Input: Un conjunto U, una colección $\mathcal{S}$ de subconjuntos de U y un número k.

Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

$$\mathrm{SC} = \{\langle U, \mathcal{S}, k\rangle : \exists \mathcal{C} \subseteq \mathcal{S},\ |\mathcal{C}| \leq k,\ \bigcup \mathcal{C} = U\}.$$

Set Cover

- Satisfacer requerimientos con tareas

Def.

Input: Un conjunto U, una colección $\mathcal{S}$ de subconjuntos de U y un número k.

Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

$$\mathrm{SC} = \{\langle U, \mathcal{S}, k\rangle : \exists \mathcal{C} \subseteq \mathcal{S},\ |\mathcal{C}| \leq k,\ \bigcup \mathcal{C} = U\}.$$

Teo.

SC es NP-completo.

Set Cover

- Satisfacer requerimientos con tareas

Def.

Input: Un conjunto U, una colección $\mathcal{S}$ de subconjuntos de U y un número k.

Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

$$\mathrm{SC} = \{\langle U, \mathcal{S}, k\rangle : \exists \mathcal{C} \subseteq \mathcal{S},\ |\mathcal{C}| \leq k,\ \bigcup \mathcal{C} = U\}.$$

Teo.

SC es NP-completo.

- ¿En NP?

Set Cover

- Satisfacer requerimientos con tareas

Def.

Input: Un conjunto U, una colección $\mathcal{S}$ de subconjuntos de U y un número k.

Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

$$\mathrm{SC} = \{\langle U, \mathcal{S}, k\rangle : \exists \mathcal{C} \subseteq \mathcal{S},\ |\mathcal{C}| \leq k,\ \bigcup \mathcal{C} = U\}.$$

Teo.

SC es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

Set Cover

- Satisfacer requerimientos con tareas

Def.

Input: Un conjunto U, una colección $\mathcal{S}$ de subconjuntos de U y un número k.

Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

$$\mathrm{SC} = \{\langle U, \mathcal{S}, k\rangle : \exists \mathcal{C} \subseteq \mathcal{S},\ |\mathcal{C}| \leq k,\ \bigcup \mathcal{C} = U\}.$$

Teo.

SC es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De VC a SC.

Hamiltonian Cycle

- Planificar ruta de reparto

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

$$\text{HAMCYCLE} = \{\langle G \rangle : G \text{ tiene un ciclo hamiltoniano}\}.$$

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

$$\text{HAMCYCLE} = \{\langle G \rangle : G \text{ tiene un ciclo hamiltoniano}\}.$$

Teo.

HAMCYCLE es NP-completo.

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

$$\text{HAMCYCLE} = \{\langle G \rangle : G \text{ tiene un ciclo hamiltoniano}\}.$$

Teo.

HAMCYCLE es NP-completo.

- ¿En NP?

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

$$\text{HAMCYCLE} = \{\langle G \rangle : G \text{ tiene un ciclo hamiltoniano}\}.$$

Teo.

HAMCYCLE es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

$$\text{HAMCYCLE} = \{\langle G \rangle : G \text{ tiene un ciclo hamiltoniano}\}.$$

Teo.

HAMCYCLE es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De 3SAT a HAMCYCLE.

Traveling Salesman

- Planificar ruta de reparto a menor costo

Traveling Salesman

- Planificar ruta de reparto a menor costo

Def.

Input: Las distancias d entre n ciudades y una cota D.

Traveling Salesman

- Planificar ruta de reparto a menor costo

Def.

Input: Las distancias d entre n ciudades y una cota D.

Output: ¿Hay un tour de largo total a lo más D?
Un tour visita cada ciudad una vez y vuelve al inicio.

Traveling Salesman

- Planificar ruta de reparto a menor costo

Def.

Input: Las distancias d entre n ciudades y una cota D.

Output: ¿Hay un tour de largo total a lo más D?
Un tour visita cada ciudad una vez y vuelve al inicio.

$$\mathrm{TSP} = \{\langle d, D\rangle : \text{hay un tour de largo total } \leq D\}.$$

Traveling Salesman

- Planificar ruta de reparto a menor costo

Def.

Input: Las distancias d entre n ciudades y una cota D.

Output: ¿Hay un tour de largo total a lo más D?
Un tour visita cada ciudad una vez y vuelve al inicio.

$$\mathrm{TSP} = \{\langle d, D\rangle : \text{hay un tour de largo total } \leq D\}.$$

Teo.

TSP es NP-completo.

Traveling Salesman

- Planificar ruta de reparto a menor costo

Def.

Input: Las distancias d entre n ciudades y una cota D.

Output: ¿Hay un tour de largo total a lo más D?
Un tour visita cada ciudad una vez y vuelve al inicio.

$$\mathrm{TSP} = \{\langle d, D\rangle : \text{hay un tour de largo total } \leq D\}.$$

Teo.

TSP es NP-completo.

- ¿En NP?

Traveling Salesman

- Planificar ruta de reparto a menor costo

Def.

Input: Las distancias d entre n ciudades y una cota D.

Output: ¿Hay un tour de largo total a lo más D?
Un tour visita cada ciudad una vez y vuelve al inicio.

$$\mathrm{TSP} = \{\langle d, D\rangle : \text{hay un tour de largo total } \leq D\}.$$

Teo.

TSP es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

Traveling Salesman

- Planificar ruta de reparto a menor costo

Def.

Input: Las distancias d entre n ciudades y una cota D.

Output: ¿Hay un tour de largo total a lo más D?
Un tour visita cada ciudad una vez y vuelve al inicio.

$$\mathrm{TSP} = \{\langle d, D\rangle : \text{hay un tour de largo total } \leq D\}.$$

Teo.

TSP es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De $\mathrm{HAMCYCLE}$ a TSP.

Hamiltonian Path

- Ruta de reparto con partida y final fijo

Hamiltonian Path

- Ruta de reparto con partida y final fijo

Def.

Input: Un grafo dirigido G.

Hamiltonian Path

- Ruta de reparto con partida y final fijo

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un camino que visite cada vértice exactamente una vez?

Hamiltonian Path

- Ruta de reparto con partida y final fijo

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un camino que visite cada vértice exactamente una vez?

$$\text{HAMPATH} = \{\langle G \rangle : G \text{ tiene un camino hamiltoniano}\}.$$

Hamiltonian Path

- Ruta de reparto con partida y final fijo

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un camino que visite cada vértice exactamente una vez?

$$\text{HAMPATH} = \{\langle G \rangle : G \text{ tiene un camino hamiltoniano}\}.$$

Teo.

HAMPATH es NP-completo.

Hamiltonian Path

- Ruta de reparto con partida y final fijo

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un camino que visite cada vértice exactamente una vez?

$$\text{HAMPATH} = \{\langle G \rangle : G \text{ tiene un camino hamiltoniano}\}.$$

Teo.

HAMPATH es NP-completo.

- ¿En NP?

Hamiltonian Path

- Ruta de reparto con partida y final fijo

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un camino que visite cada vértice exactamente una vez?

$$\text{HAMPATH} = \{\langle G \rangle : G \text{ tiene un camino hamiltoniano}\}.$$

Teo.

HAMPATH es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

Hamiltonian Path

- Ruta de reparto con partida y final fijo

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un camino que visite cada vértice exactamente una vez?

$$\text{HAMPATH} = \{\langle G \rangle : G \text{ tiene un camino hamiltoniano}\}.$$

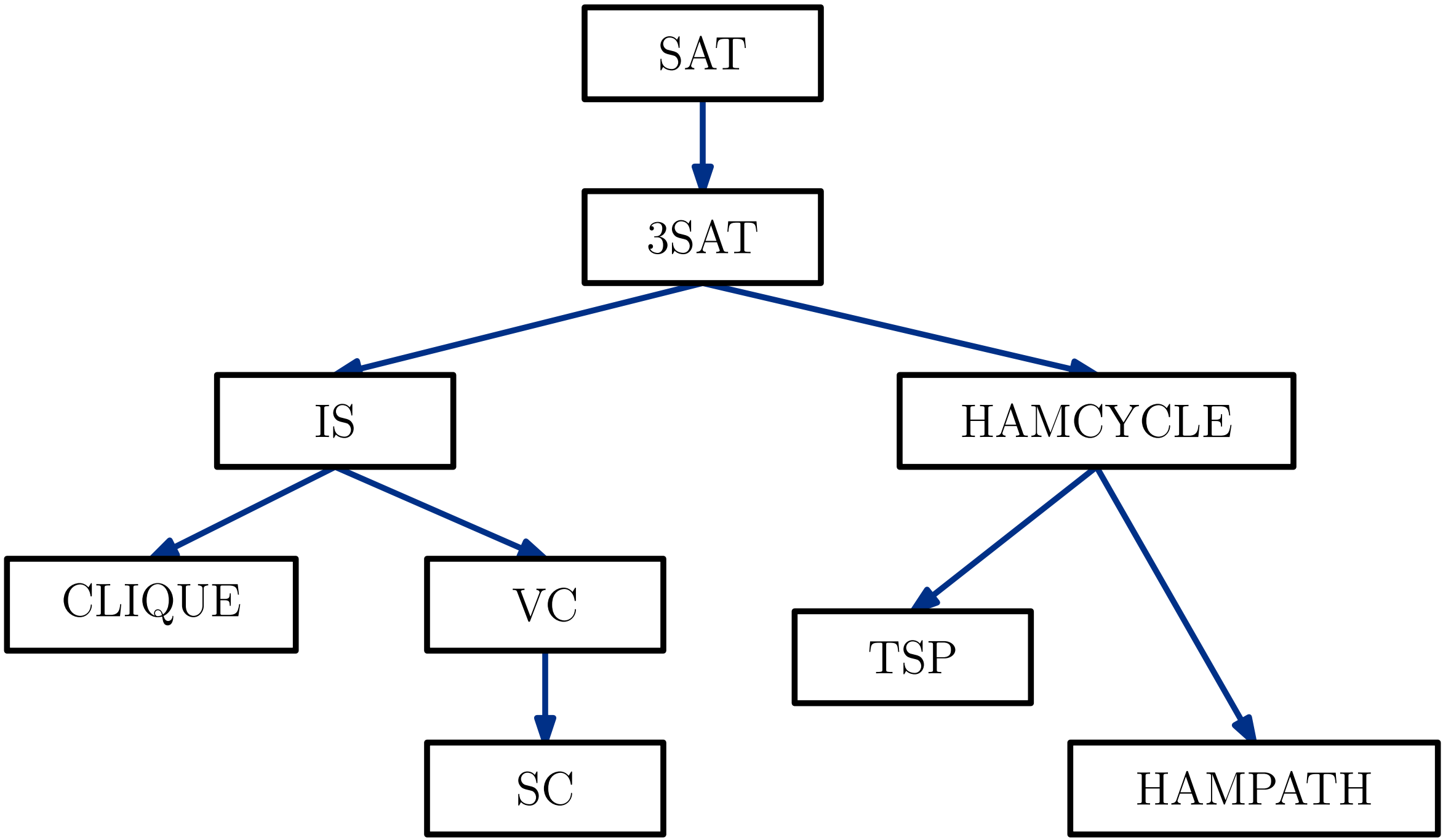
Teo.

HAMPATH es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De HAMCYCLE a HAMPATH.

Red de reducciones

La red de reducciones



Dudas/Preguntas