Diseño de Algoritmos
Sesión 09

Profesores:
Tomás Lara Valdovinos – t.lara@uandresbello.edu
Jessica Meza-Jaque – je.meza@uandresbello.edu

OBJETIVOS DE LA SESIÓN

- Conocer el concepto de reducción polinomial
- Conocer el concepto de NP-Completo
- Conocer el concepto de NP-Duro

CONTENIDOS DE LA SESIÓN

- Concepto de reducción polinomial
- Concepto NP-Completo
- Concepto NP-Duro
- Ejemplos

NP-Completo – Visión global

- El término NP-Completo se utiliza para describir los problemas de decisión más difíciles en NP.
- Si existiera un algoritmo polinomialmente acotado para un problema NP-Completo, existiría un algoritmo polinomialmente acotado para todos los problemas en NP.
- Los problemas incluidos en la clase NP-Completo son equivalentes. Se refiere a que si cualquiera de ellos está en P, entonces todos lo están!

Reducción Polinomial

- Supongamos que tenemos un problema que denotaremos **R**, y otro problema **R’**.
- Decimos que R es reducible a R’ si:
  - Existe una función oráculo **f** que resuelve R’ si y solo si la salida de f(x) para todo x en R, también resuelve R.
- R puede ser tanto un problema de **Búsqueda** o de **Decisión**

Reducción polinomial

- Si existe un algoritmo A que resuelve de manera eficiente el problema de R’ entonces decimos que R es eficientemente resoluble.

- Una función oráculo es una subrutina funcionalmente especificada (sabemos lo que hace) Pero su operación permanece sin especificar.

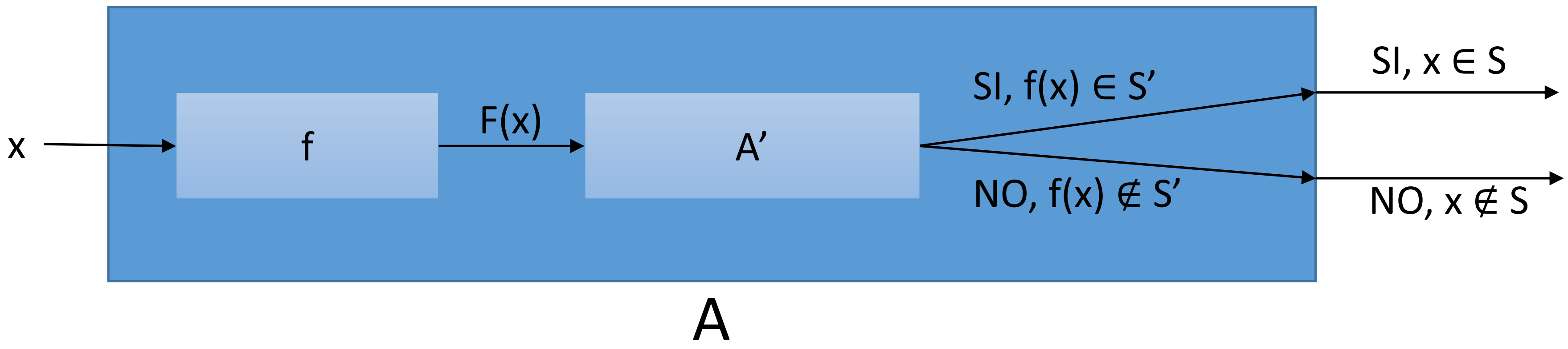
Reducción polinomial - Ejemplo

- Todo problema en **PC** es reducible a algún problema de decisión en **NP**

Reducción de Karp

- Sea **S** y **S'** dos problemas de decisión

- Decimos que f es una reducción polinomial de Karp de S a S' si:
  - Para cada x, tenemos que x pertenece a S si y solo si f(x) pertenece a S'
  - A es el algoritmo que resuelve el problema de S
  - A' es el algoritmo que resuelve el problema de S'



NP-Completo

- Formalizando el concepto NP-Completo tenemos que:

- Un problema **C** es NP-Completo si:
  - C pertenece a **NP**
  - **Todo** problema en **NP** es reducible polinomialmente a **C**

- En consecuencia, de tenerse un algoritmo en **P** que resuelva C, se tendría una solución en P para todos los problemas en NP, por lo tanto:

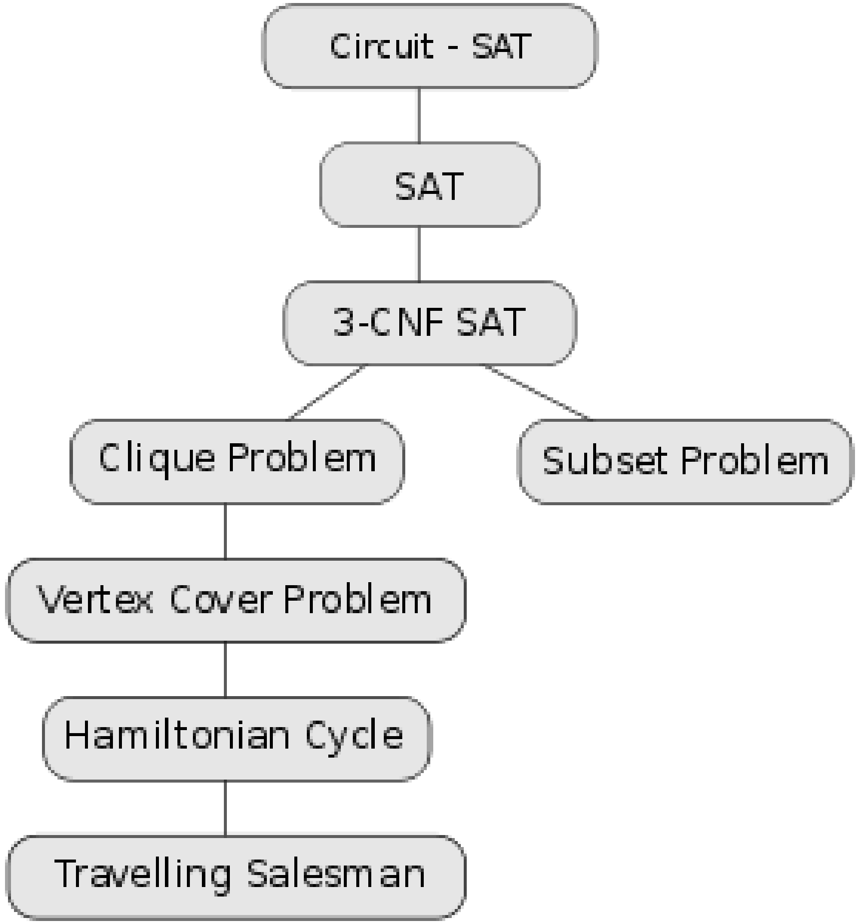
$$P = NP$$

NP-Completo

- En cambio, si pudiera **demostrarse** que no existe **algoritmo eficiente** que resuelva cualquier problema NP-Completo, entonces:

$$P \neq NP$$

NP-Completo – ejemplo reducciones



NP-Duro (NP-Hard)

- Son todos los problemas los cuales pueden ser **reducidos a otros NP-Completo**, pero no se puede identificar su pertenencia a **NP**

- En palabras simples, son problemas difíciles de resolver y no se puede verificar en tiempo polinómico la validez de su solución.

Soluciones aproximadas

- Las soluciones para NP-Completo tienen solución en tiempo exponencial
- Podemos utilizar alguno de estos enfoques para resolver un problema NP-Completo de tamaño arbitrario:
  - Aproximación
  - Probabilístico
  - Restricciones
  - Casos particulares
  - Algoritmo genético
  - Heurísticas

CHECK - OBJETIVOS DE LA SESIÓN

- Conocer el concepto de reducción polinomial
- Conocer el concepto de NP-Completo
- Conocer el concepto de NP-Duro

CHECK

Diseño de Algoritmos

Sesión 09

Profesores:
Tomás Lara Valdovinos – t.lara@uandresbello.edu
Jessica Meza-Jaque – je.meza@uandresbello.edu