Complejidad y optimización

Problemas computacionales, intratabilidad y problemas NP completos

Facultad de Ingeniería. Universidad del Valle

Mayo 2019

Universidad del Valle

Contenido

Universidad del Valle

Contenido

Universidad del Valle

Complejidad computacional

Introducción

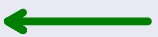
- En ciencias de la computación se utilizan algoritmos para resolver problemas y siempre se busca que esta solución sea la más eficiente.
- Una de las principales características que describe un algoritmo es su complejidad computacional que puede ser en términos de tiempo de ejecución o espacio necesario para hacerlo

Universidad del Valle

Complejidad computacional

Clases de complejidad

Existen varias clases de complejidad computacional

- Temporal 
- Espacial

Universidad del Valle

Complejidad computacional

Complejidad temporal

Es el tiempo requerido por un algoritmo para solucionar un problema.

- Varios algoritmos pueden solucionar el mismo problema
- Siempre buscamos la solución con menor complejidad temporal

Universidad del Valle

Complejidad computacional

Complejidad espacial

Es la memoria requerida por las estructuras de datos de un algoritmo para solucionar un problema.

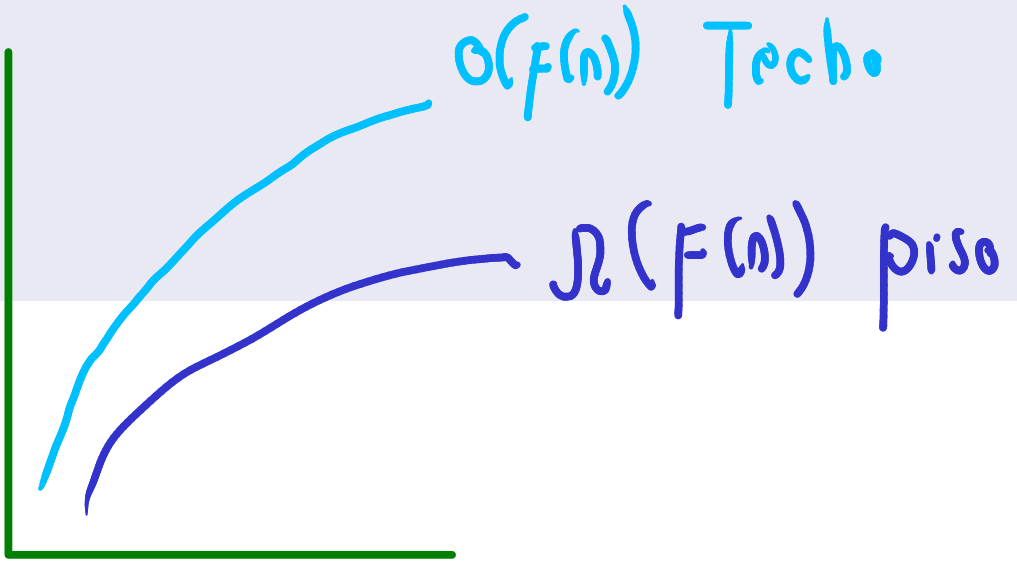
- Está directamente relacionada con la complejidad temporal, por ejemplo no es lo mismo recorrer un arreglo unidimensional que uno bidimensional
- Adquiere importancia la **selección de las estructuras de datos** utilizadas en la implementación, ejemplo: arreglos o listas enlazadas.
- Cada estructura de datos tiene su construcción en memoria, por ejemplo las listas enlazadas requieren almacenar los punteros del siguiente elemento y los arreglos sólo requieren el puntero del primer elemento.

Complejidad computacional

Notación

De acuerdo a la complejidad de la solución ante una entrada de tamaño n se utilizan las siguientes notaciones:

- $O(f(n))$
- $\Omega(f(n))$
- $\Theta(f(n))$



Universidad del Valle

Complejidad computacional

Notación

Recordemos ¿Que significa?

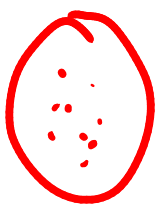
- $O(1)$ ← Constante
- $O(n^3)$ ← Polinomica
- $O(2^n)$ ← Exponencial / no polinomica
- $\Omega(1)$
- $\Omega(n^2)$
- $\Theta(1)$
- $\Theta(nlog(n))$

Universidad del Valle

Clasificación problemas

General

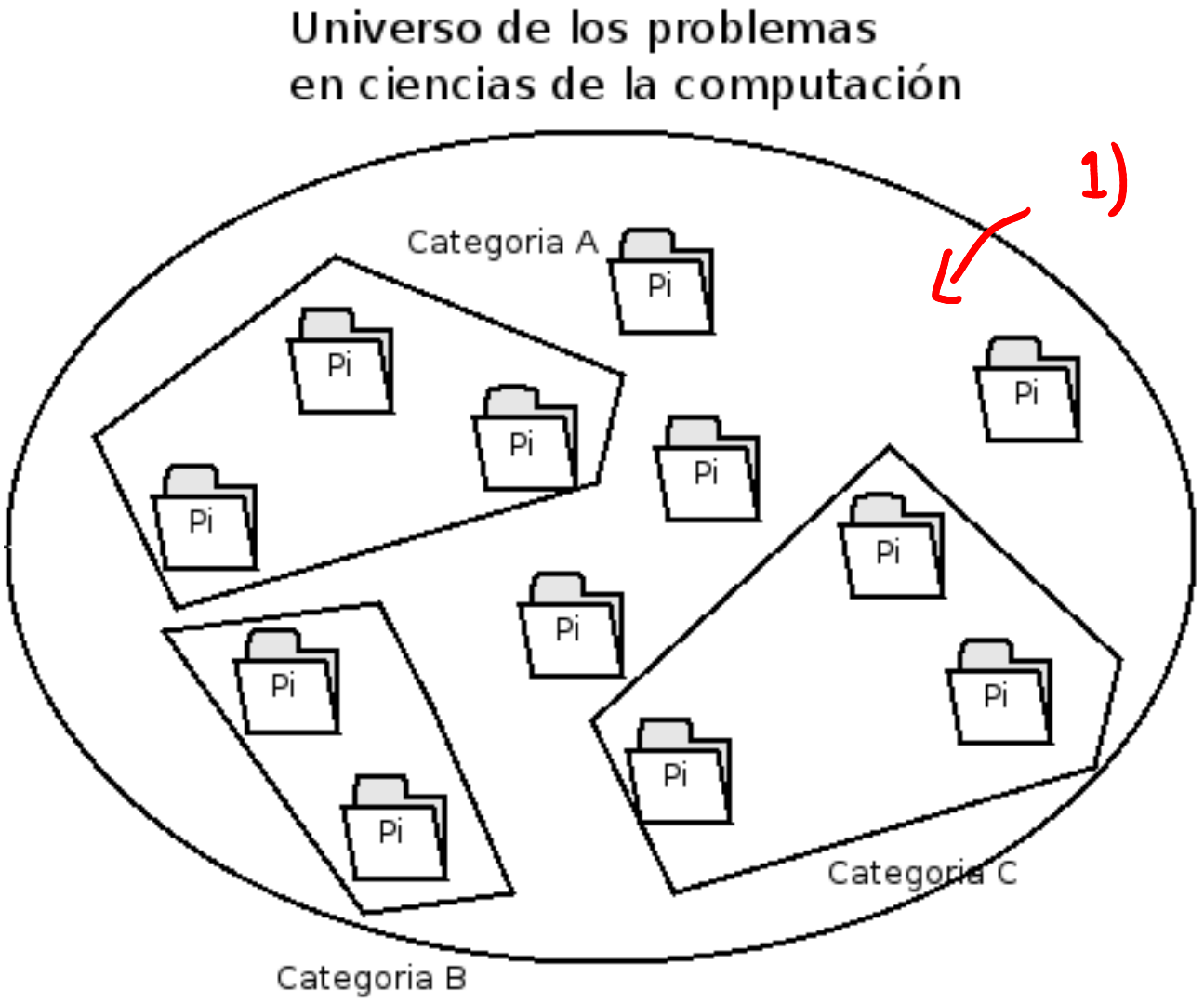
- En las ciencias de la computación los problemas tienen diferentes niveles de dificultad
- Algunos se pueden resolver y otros no, con la capacidad que contamos en la actualidad
- Se clasifican los problemas con el objetivo de agruparlos de acuerdo a un problema conocido



(a b c) = P (a, b, c, d)

Universidad del Valle

Clasificación problemas



Clasificación problemas

De acuerdo al tipo

Se pueden clasificar en: Problemas de solución y problemas de decisión

Problemas optimización

Estos consisten en encontrar la mejor solución a un problema de acuerdo a un criterio, estos pueden ser de decisión o solución.

Clasificación problemas

Ejemplo problemas de solución

1. Encontrar todos los caminos en un grafo desde un nodo A hasta un nodo B
2. Hallar las combinaciones de elementos en un conjunto S, cuya suma sea P

Clasificación problemas

Ejemplo problemas de decisión

1. El camino más corto entre A y B está conformado por los nodos $\{C, D, E\}$
2. Los elementos del subconjunto $S_1, S_2, .., S_N$ del conjunto S suman P

Respuesta: **SI** o **NO**

Clasificación problemas

Ejemplo problemas de optimización

1. Encontrar el camino más corto en un grafo desde un nodo A hasta un nodo B
2. Hallar las combinación con el menor número de elementos en un conjunto S, cuya suma sea P

Tratabilidad

Decibilidad

Los problemas puede ser decidibles o indecidibles.

Problemas decidibles

Pueden ser tratables o intratables

Problemas indecidibles

Puede ser indecidibles o altamente indecidibles.

Tratabilidad

Decibilidad

1. Los problemas decidibles son aquellos para los cuales se puede construir un algoritmo que lleve a una respuesta correcta
2. En sentido más formal, dado un problema P y una entrada s esta puede ser procesada por una máquina de Turing determinista
3. Estos problemas están clasificados en tratables e intratables.

Universidad del Valle

Tratabilidad

Decibilidad

Existen problemas indecidibles conocidos

- Problema de la parada. *Dado un algoritmo p y una entrada s, determinar si p puede procesar s en un número finito de pasos*
- Problema de la matriz mortal. *Determinar si existe un orden de multiplicaciones de matrices cuadradas de tal forma se pueda obtener la matriz cero*
- Conjetura de Collatz (3n + 1) *La sucesión dada por*

$$f(n) = \begin{pmatrix} 3n+1 & con\ n\ impar \\ \frac{n}{2} & con\ n\ par \end{pmatrix}$$

¿Es finita para todo n?

1 n: 1

F(5): 5, 16, 8, 2, 1

Universidad del Valle

Tratabilidad

Trabilidad

Asumiendo una máquina determinista como modelo de computación.

1. Los problemas tratables se pueden solucionar con algoritmos polinómicos $O(n^k)$
2. Los problemas intratables se resuelven en tiempos no polinómicos $\Omega(a^n)$

Universidad del Valle

Clasificación Máquinas de Turing

En las máquinas de Turing tenemos finitas e infinitas. Dentro de las finitas se tiene:

Determinista

Esta ejecuta una acción definida ante la lectura de un símbolo de entrada. **Este es el modelo computacional actual**

No determinista

Se replica para evaluar todas las ramificaciones que implica la lectura de un símbolo de entrada. **¡Afortunadamente, no existe ya que nos quitaría el trabajo!**

Universidad del Valle

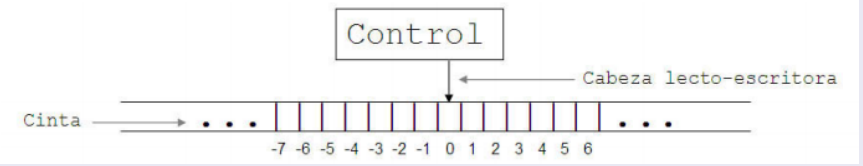
Contenido

Universidad del Valle

Máquina de Turing determinista

Definición

La máquina de Turing determinista es un dispositivo que manipula símbolos de una cinta de entrada, de acuerdo a una tabla de estados.



Los elementos de la máquina de Turing son:

- $\Lambda = \Sigma$ alfabeto $\cup$ B (Blanco)
- Q conjunto de estados (finito)
- $q_0 \in Q$ estado inicial.
- $Q_f \in Q$ conjunto de estados finales y de aceptación

Máquina de Turing determinista

Funcionamiento

- La entrada se lee en la cinta, un símbolo a la vez
- Se define un programa S como un conjunto de quíntuplas de $S \subset QxAxQxAxM$ donde $M = \{+1, -1\}$ que son los movimientos de la cabeza a derecha o izquierda
- Para cada par (q_i, s_i) existe una quíntupla que comienza ese par (máquina determinista)

$(q_i, S_t) \rightarrow (q_j, S_b, M_i)$

Estado actual

Estado siguiente

$(q_i, S_t, q_j, S_b, M_i)$

Símbolo de entrada

Símbolo

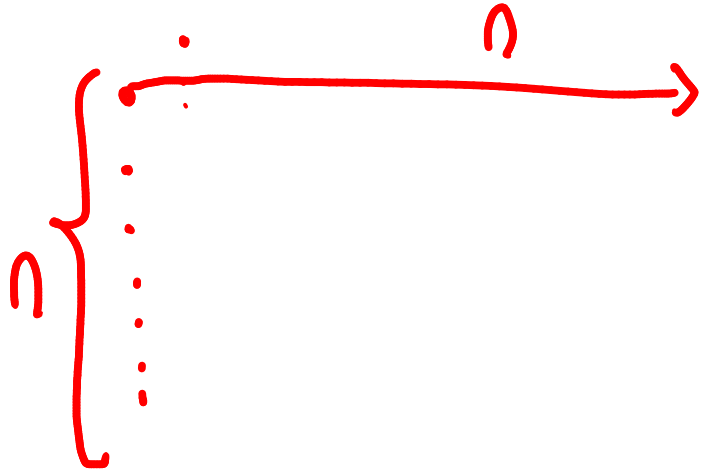
Universidad del Valle

$(q_1, q_2, \dots q_n)$ $(q_i < q_{i+1} < q_{i+2} < \dots < q_k)$

↓

DTM $n \times n-1 \times n-2 \times \dots 1 \to O(n!)$

NDTM



$O(n)$

Máquina de Turing determinista

Funcionamiento

- ¿Que sinifica la quintupla $(q_i, s_h, q_j, s_k, +1)$?. Si esta en el estado q_i la cabeza lee s_h, entonces escribe s_k, se mueve a la derecha y pasa al estado q_j.
- La complejidad de un algoritmo está dado por la cantidad de movimientos de la cabeza en función de la entrada
- Un problema está en P si existe una DTM de complejidad polinomial lo resuelve

Universidad del Valle

Máquina de Turing no determinista

Funcionamiento

- No pide unicidad de la quíntupla que comienza con cualquier par (q_i, s_j)
- En caso que hubiera más de una quíntupla, la máquina se replica continuando cada una una por una rama distinta. **De forma simultanea**
- Se dice que una máquina de Turing no determinista llega a un estado final por cualquiera de sus ramas, se da por solucionado el problema.

Máquina de Turing no determinista

Funcionamiento

- Una máquina de Turing no determinista es polinomial cuando existe una función polinomial que resuelve una instancia de un algoritmo dado en alguna de las ramas.
- Un problema es NP si existe una máquina de turing no determinista que lo solucione en tiempo polinomial
- Equivalentemente, un problema NP puede ser verificado en tiempo polinomial.

NP

Universidad del Valle

Contenido

Universidad del Valle

Clasificación P

Problema P

Es aquel problema de decisión que se puede solucionar en tiempo polinomial por una máquina determinista.

Problema NP

Son aquellos que **NO** pueden ser solucionados en tiempo polinomial por una máquina determinista. Pero **SI** pueden ser solucionado en tiempo polinomial por una máquina no determinista. Además, podemos verificarlos en tiempo polinomial por una máquina determinista.

Clasificación P

Problema NPC

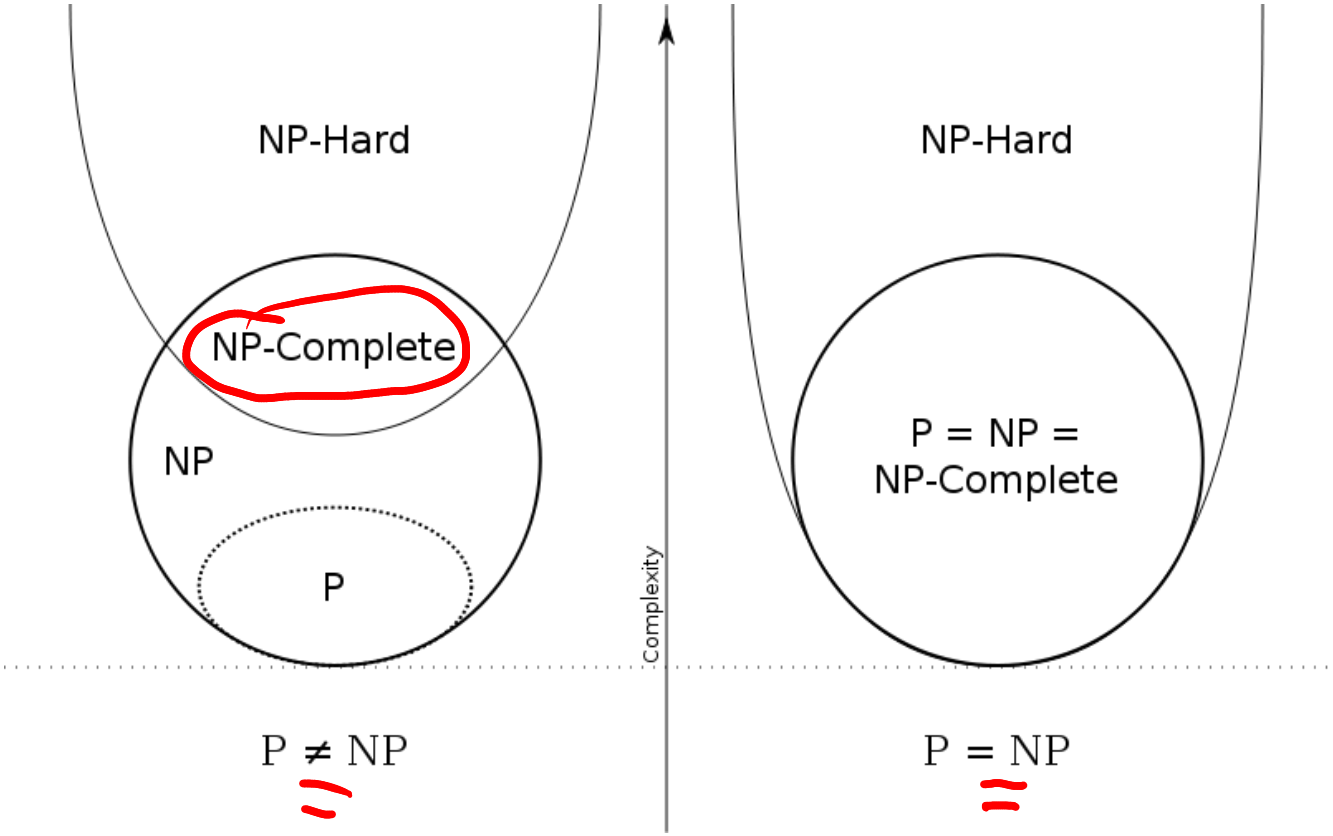
NP-Completo, son problemas que cumplen las siguientes características:

- No se ha demostrado que **NO** existe un algoritmo con complejidad polinomial que los pueda resolver
- No se ha encontrado una solución polinomial para ellos.

Problema NP-Hard

Son los problemas más difíciles de la calificación, estos problemas no pueden ser reducidos a uno NPC. Además, no podemos verificarlos en tiempo polinomial por una máquina determinista.

¿P=NP?



El problema ¿P=NP?

Universidad del Valle

Contenido

Universidad del Valle

SAT

Introducción

Este es el primer problema NPC completo demostrado. La demostración fue realizada por Stephen Cook en 1971.

Definición

Cualquier problema NPC puede ser reducido desde SAT en tiempo polinomial.

SAT

Definición formal

El problema consiste en un conjunto de variables booleanas $x_1, x_2.x_3, .., ..x_n$ y un conjunto de clausulas $s_1, s_2, s_3, .., s_n$. El objetivo es indicar si existen los valores de verdad de las variables en la cual la expresión es verdadera.

Example

$$(x_1 \vee x_2) \wedge (x_2 \vee x_3)$$
$$(x_1 \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee x_3)$$
$$(x_1 \vee x_2 \wedge x_3) \wedge \bar{x_1}$$
$$(x_1 \implies x_2 \vee x_3)$$

Universidad del Valle

SAT

Definición formal

Para facilitar el estudio del problema lo vamos a trabajar en su forma normal conjuntiva (conjunciones de disyunciones). Dado que SAT en su forma normal conjuntiva es un subconjunto del problema SAT sin restricciones, este también es NPC.

Example

$$(x_1 \vee x_2) \wedge (x_2 \vee x_3)$$
$$(x_1 \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee x_3)$$
$$(x_1 \vee x_2 \vee x_3) \wedge \bar{x_1}$$
$$(x_1 \vee x_3) \wedge x_1$$

Universidad del Valle

¿SAT es NPC?

Enunciados demostración

- Se utiliza una máquina de Turing no determinista para el cómputo de este algoritmo.
- Se codifica la entrada de la máquina de Turing de tal forma esta resulta en una fórmula booleana en FNC.
- Se la máquina de Turing acepta la entrada, entonces la fórmula es satisfactible.

¿SAT es NPC?

Explicación de la demostración

La máquina de Turing no determinista al poder procesar la fórmula booleana, dando los valores correctos a cada variable de cada clausula, conduce a un estado de aceptación, de acuerdo a los valores asignados a cada variable. Además se puede verificar en tiempo polinomial en máquina determinista que la salida es correcta, al evaluar las variables.

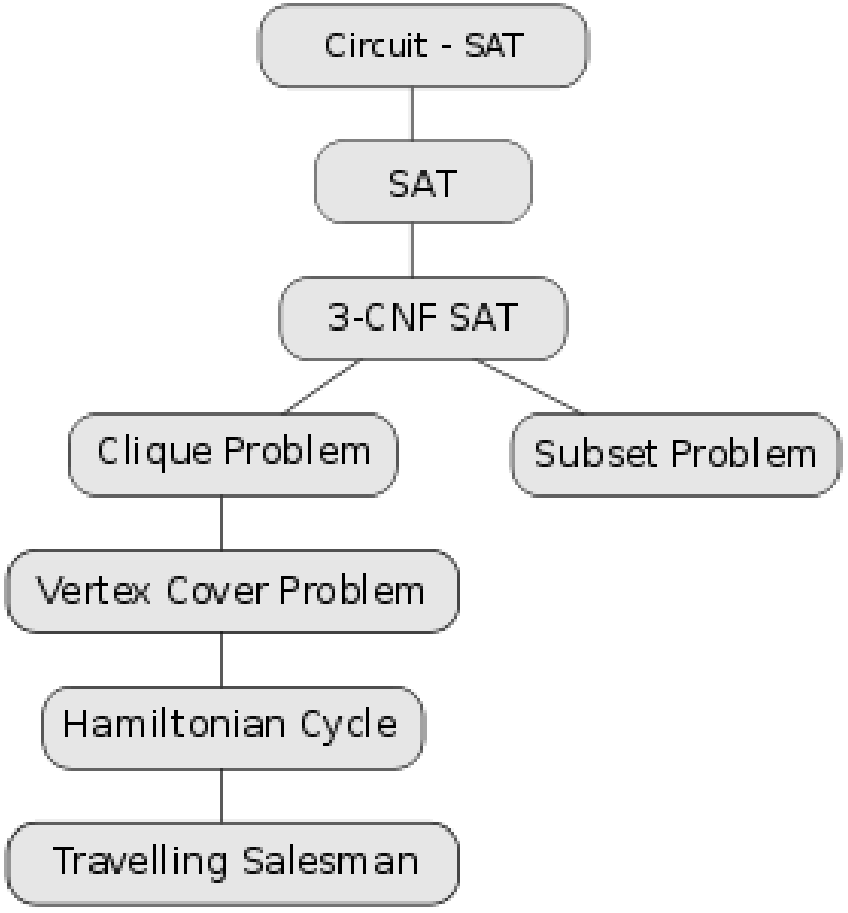
¿SAT es NPC?

Consecuencias

- Si se llegara a demostrar que SAT puede ser resuelto en tiempo polinomial entonces cualquier problema NPC puede ser resuelto en tiempo polinomial.
- Al reducir cualquier problema NP a SAT, se demuestra que es NPC.

Universidad del Valle

Problemas NPC



Universidad del Valle

Reducción

Definición

Sea A un problema de decisión NPC conocido y B un problema de decisión NP. Se puede transformar una instancia A en una instancia B en tiempo polinomial.

Requerimiento

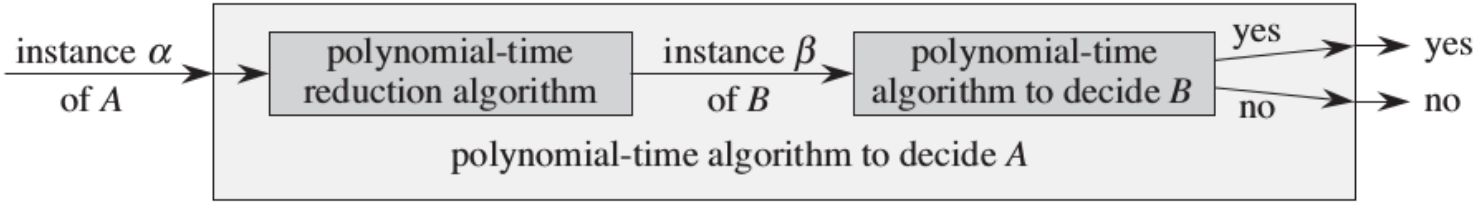
Se debe cumplir:

$$A \leq_p B$$

Es decir A tiene una complejidad menor o igual a B.

Universidad del Valle

Reducción



Definición

Una reducción es correcta si:

- Instancias negativas de A resultan en instancias negativas de B
- Instancias positivas de A resultan en instancias positivas de B

$x_1 \wedge \tilde{x}_1$

Universidad del Valle

Preguntas.