Máquinas de Turing
MTU, Problemas Indecidibles y problemas P-NP

Fabio Martínez Carrillo

Autómatas
Escuela de Ingeniería de Sistemas e Informatica
Universidad Industrial de Santander - UIS

28 de noviembre de 2017

Agenda

1. Simulación de AF, PDA con MT
2. Máquinas de Turing Universal
   - Codificación y enumeración de MT
   - Maquina de Turing Universal (M_u)
3. Problemas indecidibles
4. Problemas P-NP
   - Problemas P
   - Problemas NP
   - NP completos

Simulación de Autómatas con MT

- **AFD** se simula con una **MT** estandar
- **Autómata con pila** con una **MT** de dos cintas
- Los lenguajes regulares y los independientes del contexto son **recursivos**

Simulación de Autómatas

AFD, AFN, AFN-ε

- Se convierte a un AFD $M = (Q, q_0, F, \Sigma, \delta)$
- Se contruye la MT M' tal que $L(M) = L(M')$ añadiendo un estado q_f
- Se adicionan transiciones de F hacia q_f cuando el simbolo es blanco
- $M' = (Q', q_0, F', \Sigma, \Gamma, B, \delta')$

Simulación de Autómatas

- $Q = Q \cup \{q_f\}$, q_f es el estado nuevo
- $\Gamma = \Sigma \cup \{B\}$
- $F' = \{q_f\}$
- $\delta'(q, s) = (\delta(q, s), s, \rightarrow)$
- $\delta'(q, B) = \delta(q_f, b, -)$

Teorema

Todo lenguaje regular es recursivo

Simulación de Autómatas con Pila

- Autómata con Pila $M = (Q, q_0, F, \Sigma, \Gamma, z_0, \delta)$
- La MT M' actua sobre las dos cintas:
  - La primera los simbolos de entrada
  - La segunda simula la pila

$M' = (Q', q_0, F', \Gamma', B, \delta)$

- $Q' = Q \cup \{q_f\}$
- $\Gamma' = \Sigma \cup \Gamma \cup B$
- $F' = \{q_f\}$

Simulación de Autómatas con Pila

Configuración inicial

$$\delta(q_0, (s, B)) = (q_0, (s, z_0), (-, -))$$

Transición: $\delta(q, a, s) = (p, \gamma)$

Por ejemplo: $\delta(q, a, s) = (p, a_1 a_2 a_3)$ se simula:

- $\delta(q, (a, s)) = (q_e, (a, -), (a_3, \rightarrow))$
- $\delta(q_e, (a, B)) = (q_e, (a, -), (a_2, \rightarrow))$
- $\delta(q_e, (a, B)) = (p, (a, \rightarrow), (a_1, -))$

Transición final

$$\delta(q, (B, s)) = (q_f, (B, s), (-, -))$$

Teorema

Todo LIC es un lenguaje recursivo.

Agenda

1. Simulación de AF, PDA con MT
2. Máquinas de Turing Universal
   - Codificación y enumeración de MT
   - Maquina de Turing Universal (M_u)
3. Problemas indecidibles
4. Problemas P-NP
   - Problemas P
   - Problemas NP
   - NP completos

Máquinas de Turing y algoritmos

La MT es un modelo conveniente para representar ***"lo que es computable"***

Tesis de Church-Turing

Todo algoritmo puede ser descrito por medio de una máquina de Turing.

- Tanto los algoritmos que producen una salida para cada entrada como aquéllos que no terminan (ingresan en bucles infinitos)

Máquinas de Turing y algoritmos

Tesis de Church-Turing

Todo algoritmo puede ser descrito por medio de una máquina de Turing.

1. La adición de recursos computacionales a las MT (múltiples pistas o cintas, no determinismo, etc) no incrementa el poder computacional del modelo básico: **representa el límite de lo que un dispositivo de computación secuencial puede hacer**

Codificación y enumeración de MT

Toda MT se puede codificar como una secuencia binaria finita

Codificación

- Codificación para las MT que actuan sobre un alfabeto de entrada pre-establecido
- Suponemos que toda **MT** tiene un único estado inicial q_1 , y un único estado final q_2
- El alfabeto de la cinta:

$$\Gamma = \{s_1, s_2, \ldots, s_m, \ldots, s_p\}$$

  - s_1 símbolo en blanco B
  - $\Sigma = \{s_2, \ldots, s_m\}$
  - $s_{m+1}, \ldots, sp$ símbolos auxiliares

Los símbolos $\{s_1, s_2, \ldots, s_m, \ldots, s_p\}$ se codifican como secuencias de unos

Símbolo	Codificación
s_1 (símbolo b̶)	1
s_2	11
s_3	111
$\vdots$	$\vdots$
s_m	$\underbrace{11\cdots1}_{m \text{ veces}}$
$\vdots$	$\vdots$
s_p	$\underbrace{11\cdots1}_{p \text{ veces}}$

Los estados de la **MT** $q_1, q_2, \ldots, q_n$ se codifica como una secuencia de unos.

Estado	Codificación
q_1 (inicial)	1
q_2 (final)	11
$\vdots$	$\vdots$
q_n	$\underbrace{11\cdots1}_{n \text{ veces}}$

Los desplazamientos $\rightarrow$, $\leftarrow$, $-$ se codifican como 1, 11, 111.

Codificación de funciones de transición

$\delta(q, a) = (p, b, D)$ se codifica utilizando ceros como separadores.

Ejemplo: $\delta(q_3, s_2) = (q_5, s_3, \rightarrow)$

0111011011111011101010

$\delta(q_i, s_k) = (q_j, s_l.D)$

$01^i01^k01^j01^l01^t0$

* hay 6 ceros separados por una secuencia de unos

Codificación de una MT

Una MT se codifica escribien consecutivamente las codificaciones de todas sus transiciones.

$$C_1 C_2 \ldots C_r$$

Como el orden no importa, una MT puede tener varias configuraciones.

MT que acepta el lenguaje a^+b

$$\delta(q_1, a) = (q_3, a, \rightarrow)$$
$$\delta(q_3, a) = (q_3, a, \rightarrow)$$
$$\delta(q_3, b) = (q_4, b, \rightarrow)$$
$$\delta(q_4, \mathtt{b}) = (q_2, \mathtt{b}, -)$$

Cual es la secuencia de simbolos que codifica la MT?

MT que acepta el lenguaje a^+b

$$\delta(q_1, a) = (q_3, a, \rightarrow)$$
$$\delta(q_3, a) = (q_3, a, \rightarrow)$$
$$\delta(q_3, b) = (q_4, b, \rightarrow)$$
$$\delta(q_4, \text{b}) = (q_2, \text{b}, -)$$

Cual es la secuencia de simbolos que codifica la MT?

01011011101101001110110111011010011101110111011101001111010110101110

MT que acepta el lenguaje a^+b

$$\delta(q_1, a) = (q_3, a, \rightarrow)$$
$$\delta(q_3, a) = (q_3, a, \rightarrow)$$
$$\delta(q_3, b) = (q_4, b, \rightarrow)$$
$$\delta(q_4, \text{ъ}) = (q_2, \text{ъ}, -)$$

Cual es la secuencia de simbolos que codifica la MT?

$$010110111011010011101101110110100111011101111011101001111010110101110$$

$$0101^2 01^3 01^2 01001^3 01^2 01^3 01^2 01001^3 01^3 01^4 01^3 01001^4 0101^2 0101^3 0.$$

Cambiando el orden de las transiciones obtenemos $4! = 24$ diferentes configuraciones

Consideraciones

- la codificación de una MT no puede comenzar en 1
- No pueden aparecer tres ceros consecutivos.
- Las secuencias 0101000110, 010001110 y 1011010110111010 no codifican MT
- Una cadena que codifica una MT se llama **Código valido de MT**

Las cadenas se pueden organizar Lexicográficamente

$$0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, 0001, 0010, \ldots$$

Entonces las cadenas se pueden enumerar !

- Lo que se puede enumerar se puede programar (eventos discretos)

Consideraciones

- Si las cadenas se pueden enumerar
- y cada cadena representa una maquina de turin M_1 actuando sobre un Σ
- Podemos hablar de la *i*-esima MT : $\{M_1, M_2, M_3, \ldots\}$

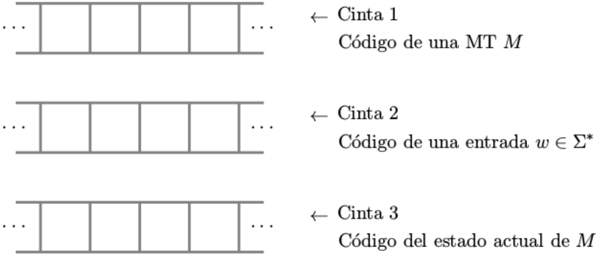
Codificación de las cadenas de entrada: $\Sigma = \{s_2, \ldots, s_m\}$

- Por ejemplo la palabra: *aabBab*
- 0110110111010110110
- para $w \in \Sigma$ no aparecen dos ceros consecutivos.
- para $s_{i_1} s_{i_2} \ldots s_{i_k}$

$$01^{i_1}01^{i_2}0 \ldots 01^{i_k}0$$

- Las cadenas de Σ^* se pueden ordenar lexicográficamente
- Esto induce un orden y obtenemos una numeración.
- $w_1, w_2, w_3 \ldots$
- Tenemos dos enumeraciones diferentes: $w_1, w_2, w_3 \ldots$ y $M_1, M_2, M_3, \ldots$

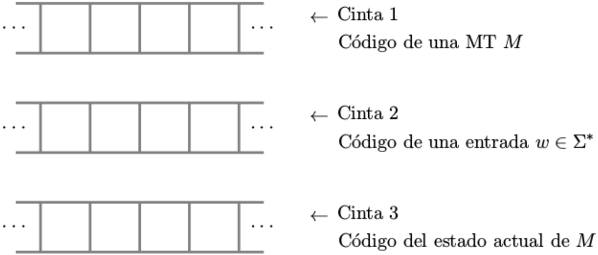
Maquina de Turing Universal (M_u)



La MT M_u simula el comportamiento de todas las MT sobre una entrada dada.

- Procesa pares (M, w). M y w son la codificación en cadenas.

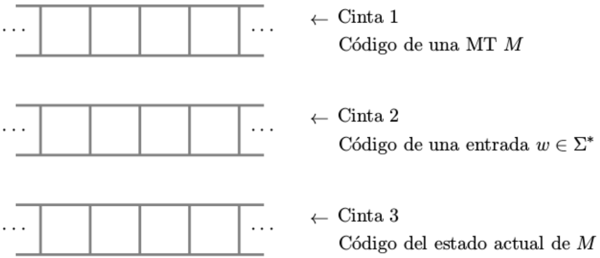
Maquina de Turing Universal (M_u)



La MT M_u simula el comportamiento de todas las MT sobre una entrada dada.

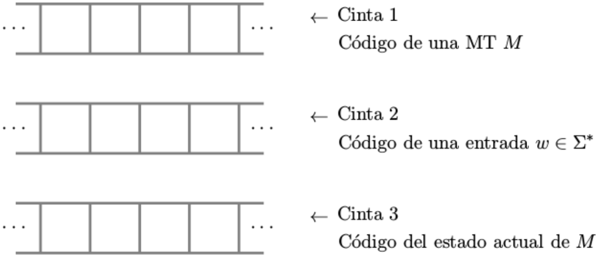
- La pareja (M, w) se puede representar como una cadena binaria $M0w$
  - Caso donde aparecen tres ceros consecutivos

Maquina de Turing Universal (M_u)



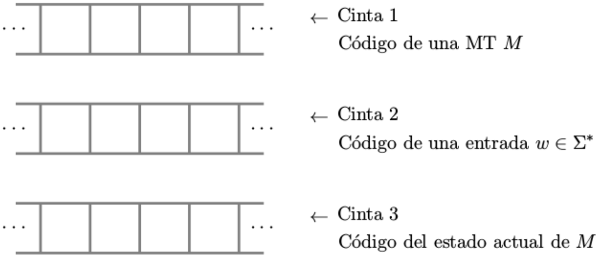
- Primera cinta contiene código MT
- Segunda cinta contiene código de entrada w
- Tercera cinta se almacenan los estados actuales de M de forma cofificada.

Funcionamiento de M_u



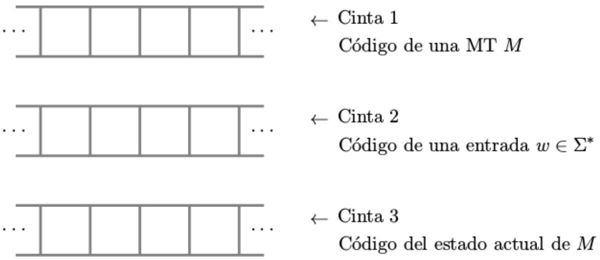
- La entrada $M0w$ se escribe en la primera cinta y las demás permanecen en blanco.
- Se Deja el código de M en la primera cinta y se copia w en la segunda

Funcionamiento de M_u



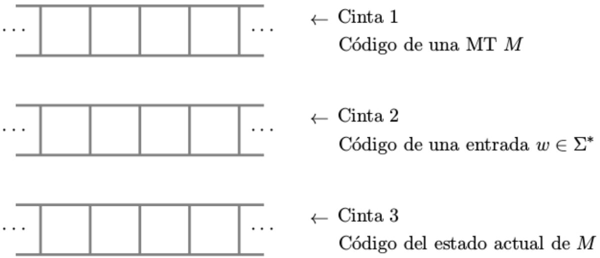
- La cadena 1 que representa el estado inicial se escribe en la tercera cinta
- La unidad de control escanea el primer símbolo de cada cadena binaria en las tres cintas

Funcionamiento de M_u



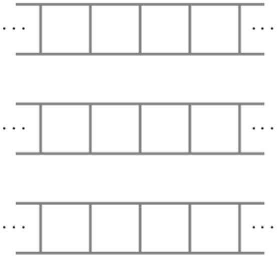
- Examina el código de M para determinar si representa un MT. Sino M_u se detiene.
- M_u utiliza la información en la cinta 2 y 3 para buscar en la cinta 1 una transición aplicable.

Funcionamiento de M_u



- Si encuentra una transición aplicable, M_u simula en la cinta dos lo que haria M
- Se cambia el estado en la cinta tres.
- Se re-escribe los simbolos en la cinta dos utilizando subrutinas.

Funcionamiento de M_u



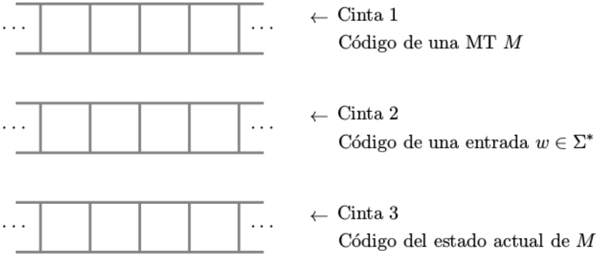
$\leftarrow$ Cinta 1
Código de una MT M

$\leftarrow$ Cinta 2
Código de una entrada $w \in \Sigma^*$

$\leftarrow$ Cinta 3
Código del estado actual de M

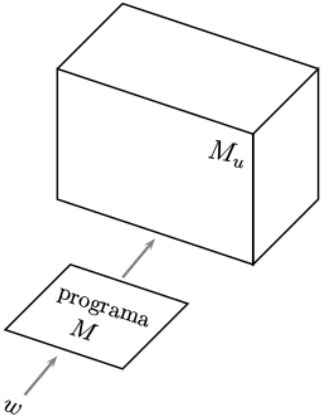
- La simulación continúa si hay transiciones aplicables.
- Si al procesar una entrada w, M_u se detiene en el único estado de aceptación. Entonces la cadena es aceptada
- Si M_u no encuentra una transición aplicable, M_u se detiene sin aceptar como M

Lenguaje Universal M_u



$$L_u = \{M0w : \text{la MT M acepta la cadena } w \in \Sigma^*\}$$

- M_u acepta si M acepta



Una MT como un programa computacional y M_u resulta ser un mecanismo en el que podemos ejecutar todos los programas.

Máquinas de Turing y computadores

MT está determinada por un conjunto finito de instrucciones (su función de transición) y está definida con referencia a alfabetos finitos.

Computador

- Capacidad de almacenamiento (memoria, discos, etc) potencialmente infinita
- El modelo multi-cintas de máquina de Turing
  - una cinta para simular la memoria principal del computador
  - Otra para las direcciones de memoria
  - Un número adicional (pero finito) de cintas para simular los discos de almacenamiento presentes

Esto se denomina **Máquina de Turing universal**

Agenda

Problemas intratables

Discusión sobre los problemas que según su eficiencia se pueden o no tratar computacionalmente.

Problemas decidibles: calculados con MT y tienen una complejidad polinómica

Problemas indecidibles: Complejidad mayor o igual a la exponencial. Resoluble en casos sencillos.

Un lenguaje es recursivamente enumerable (computable) si exhibe **un algoritmo de aceptación**

- para una entrada *u* el algoritmo finaliza en aceptación si y solo si $u \in L$
  - Para entradas no aceptadas la Mt puede para en un estado de no aceptación o seguir ejecutandose infinitamente

Lenguajes RE no recursivos

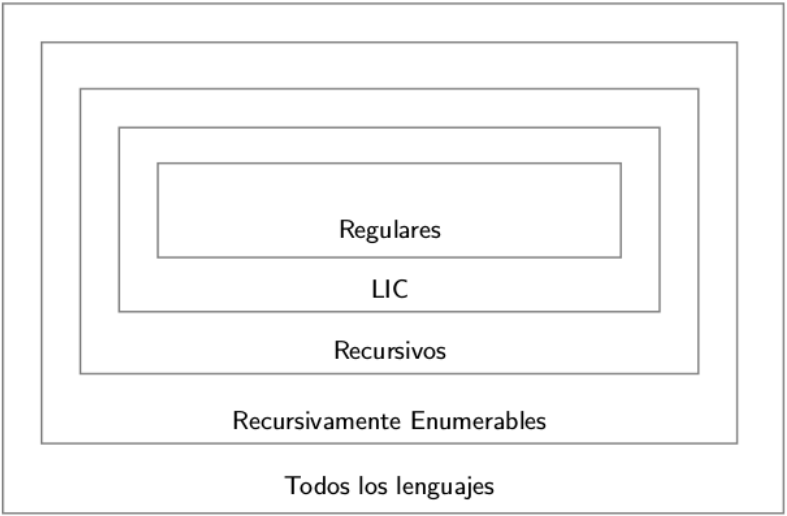
Lenguajes recursivos $\not\subseteq$ Lenguajes RE

- existen lenguajes que pueden ser aceptados por MT donde existe cómputos que no terminan
- *los cómputos interminables, o bucles infinitos, no se pueden eliminar de la teoría de la computación.*

$$L_u = \{M0w : M \text{ acepta a } w\}$$

Es RE pero no es recursivo

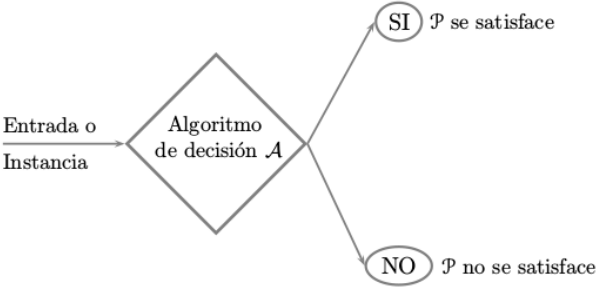
relaciones de contenencia entre las colecciones de lenguajes



Lenguajes y Máquinas que los aceptan

Lenguajes	Máquinas aceptadoras
Regulares	Autómatas finitos (AFD $\equiv$ AFN $\equiv$ AFN-λ)
LIC	Autómatas con pila no-deterministas (AFPN)
RE	Máquinas de Turing (MT)
Recursivos	Máquinas de Turing que se detienen con toda entrada

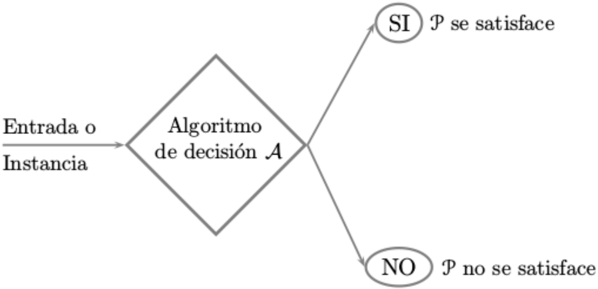
Problemas indecidibles o irresolubles



Dada una propiedad *p* referente a una MT *M*.

- Un problema consiste en buscar un algoritmo *A* aplicable a toda MT (codificación binaria) y responde a la pregunta: Satisface M la propiedad *p*?

Problemas indecidibles o irresolubles



Si existe un algoritmo de decisión se dice que el problema es **decidible** o **resoluble**

Busca soluciones para $x^n + y^n = z^n$

```
int exp(int i, n)
/* calcula i a la potencia n */
{
    int ans, j;
    ans = 1;
    for (j=1; j<=n; j++) ans *= i;
    return(ans);
}

main ()
{
    int n, total, x, y, z;
    scanf("%d", &n);
    total = 3;
    while (1) {
        for (x=1; x<=total-2; x++)
            for (y=1; y<=total-x-1; y++) {
                z = total - x - y;
                if (exp(x,n) + exp(y,n) == exp(z,n))
                    printf("hola, mundo\n");
            }
        total++;
    }
}
```

Indecibilidad

Codificación del último teorema de Fermat. solución solo para $n = 2$. Total $= 12$

Agenda

Problemas intratables

Discusión sobre los problemas que según su eficiencia se pueden o no tratar computacionalmente.

Problemas decidibles: calculados con MT y tienen una complejidad polinómica

Problemas indecidibles: Complejidad mayor o igual a la exponencial. Resoluble en casos sencillos.

Entendiendo el problema exponencial

- Ejemplo del Amir: granos de arroz en un ajedrez
  - 2^{64}: Se necesitarian con el grado de producción actual de arroz se necesitarian 400 años
  - El problema del sudoku

Problemas intratables

Discusión sobre los problemas que según su eficiencia se pueden o no tratar computacionalmente.

Problemas decidibles: calculados con MT y tienen una complejidad polinómica

Problemas indecidibles: Complejidad mayor o igual a la exponencial. Resoluble en casos sencillos.

Suponga que el tiempo para resolver una tarea que tiene como entrada N es 2^N

- Doblar la velocidad de la Maquina no hace diferencia.
- Adicionar mil maquinas solo incrementa la capacidad computacional para entradas $N + 10$
- Adicionar un millon de maquinas solo incrementa la capacidad computacional para entradas $N + 20$

Limite de tiempo de MT: Complejidad temporal

Complejidad computacional de MT

Una MT que dada una entreda w de longitud n, siempre se realiza con un máximo de $T(n)$ movimientos. Independiente si acepta o no la palabra w.

- $T(n)$ es la complejidad temporal de la MT
- Algunas veces puede ser no determinista.

Clase P

- Si una MT determinista tiene una complejidad temporal/ comptuacional $T(n)$ para algún polinomio $T(n)$, entonces la *MT* tiene un orden polinómico: $50n^2$
- Entonces el lenguaje de la (MT) pertenece a la clase **P**
- El conjunto de lenguajes polinomicos forman la clase **P**
- La complejidad de la clase **P** aplica para computadores y MT

Problemas que se resuelven con un problema polinomial.

Equivalencia polinómica de los computadores y las MT

- Una Maquina de Turing multi-cinta puede simular un computador que corre en un tiempo $O(T(n))$ en por lo menos $O(T^2(n))$
- Si $T(n)$ es polinómico, entonces $T^2(n)$ también será polinómico

- Si la complejidad computacional es de orden cubico en un PC en una MT es de orden 6

Ejemplos del problema **P**

Buscar el camino desde un nodo x hasta un nodo y en un grafo G

- Entrada: x. y, G
- La complejidad en la busqueda es $O(n)$ en un grafo de n nodos.
- En el caso particular de *Dijkstra* La complejidad es: $O(|E| + |V| \log |V|)$

Ejemplos del problema **P**

- conectividad (o la accesibilidad) en grafos no dirigidos.
- Multiplicación de matrices $O(n^3)$
- Ciclo Euleriano
- Camino Mínimo

Problemas considerados dentro de los polinómicos

- Problemas de complejidad $O(n \log n)$ en apariencia no son polinómicos
- Sin embargo para ser un problema de la clase **P** debe correr **menor o igual que** un factor polinómico
- $O(n \log n)$ es menor que el orden polinomial $O(n^2)$

Probelmas **NP**

Problemas NP

No se conoce un algoritmo con solución polinomica. Problemas que se comprueba su solución en tiempo polinomial.

- Resolver un Sudoku es complicado pero comprobar su solución es sencillo.
- Todo problema que esta en **P** esta en **NP**
- **Podemos decir lo contrario??: Todo problema NP es P**

La clase **NP**

- Esta definido en terminos de una MT no deterministica.
- El tiempo computacional esta definido como en la MT no determinista como el número máximo de pasos que toma a lo largo de alguna rama.
- Si el tiempo es polinomico, entonces la NTM es de tiempo polinómico.

Ejemplos

- El ejemplo de la mochila donde los numeros son normalmente representado en binario.
- Predecir la partición de un conjunto (coloreado de grafos)
- Sumar dos conjuntos y compararlos
- Isomorfismo de Grafos

Problemas NP completos

- Cualquier problema NP se puede reducir a un problema NP completo.
- Si encontramos un solo problema NP completo que sea P. Todos los problemas se resulven en tiempo P
- Toda la seguridad en internet se basa en restriciones de tipo NP

Problema del Millon de Dolares

P vs NP

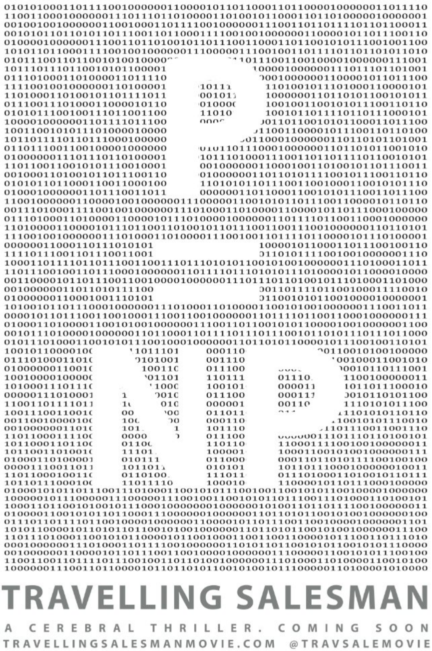
Demostrar que un problema NP no es P

- **Problema del milenio.** $P = NP$?
- P y NP representan los mismo lenguajes.
- Un problema que resuelve una NTM en tiempo polinomico lo puede resolver una MT en tiempo polinómico también?

- Existen muchos problemas NP que han sido resueltos pero no parecen tener una solución P
- Es una de las preguntas matemáticas mas importantes hoy en dia
- Intuición es que no son **P**

Problemas Completos NP

- Son problemas que son **NP** y si se puede demostrar que son **P** entonces se podría resolver cualquier problem **NP**
  - Por ejemplo el coloreado de grafos
- El Isomorfismo de grafos es el único problema que no es **NP** y se considera que problema completo. NO hay un algoritmo conocido de tiempo polinómico para resolver este problema


TRAVELLING SALESMAN
A CEREBRAL THRILLER. COMING SOON
TRAVELLINGSALESMANMOVIE.COM @TRAVSALEMOVIE

Muchas gracias por su atención

Universidad Industrial de Santander