

INF155: Informática Teórica

Clase 25: P vs NP

Aldo Berrios Valenzuela

Martes 14 de Junio de 2016

1. P vs NP

Las funciones polinomiales son cerradas respecto composición.

Si $p(n)$ y $q(n)$ son polinomios, $p \circ q(n) = p(q(n))$ también es un polinomio.

Si el tiempo de ejecución de un algoritmo no está acotado por un polinomio en el tamaño de los datos, no hay esperanza de resolver instancias más “grandes”.

1.1. Máquinas de Turing Deterministas vs otros modelos

Para cada modelo de computación podemos definir una medida “razonable” de “tiempos de ejecución”.

- Pasos de una TM determinista.
- Número de instrucciones ejecutadas por RAM.
- Número de llamadas de función en cálculo λ .

⋮

Con un poco de cuidado, si el modelo A toma tiempo T , podemos simularlo en tiempo $p(T)$ en el modelo B , donde $p()$ es un polinomio ([por ejemplo, simular el funcionamiento de una RAM en una TM](#)). Misterio: En todos los casos, el grado de p es más bien chico...

Importante Todo algoritmo que tenga solución en un tiempo polinomial p , podrá ser simulado por cualquier otro modelo de computación que tenga solución en tiempo polinomial q .

Como podemos “traducir” entre modelos de computación pagando un costo en “tiempo” acotado por un polinomio \rightsquigarrow podemos obviar el modelo exacto (no hablamos de tiempo cronológico, sino polinomial).

Problemas que no admiten solución en “tiempo” acotado por un polinomio en el tamaño de la entrada claramente no son solubles en la práctica.

Estudio de esta problemática es la teoría de complejidad. Se habla de la complejidad de un algoritmo como la función (que acota) el uso de recursos en función del tamaño del problema. Recursos = tiempo, para nosotros (en general, es el recurso más importante). Decimos que un **problema tiene solución eficiente** si el tiempo está acotado por un polinomio.

[Algunas Observaciones hasta el momento:](#)

- La complejidad del algoritmo es la cota (en tiempo polinomial) en que el algoritmo logra resolver el problema.
- Complejidad del algoritmo es cuanto se demora en función de la cantidad de datos de entrada.

Para una TM no determinista, consideramos el tiempo de ejecución como el número de pasos de la computación más corta (el camino más corto, y siempre elige el camino más corto) que lleva a aceptar.

Definición 1.1 (P y NP). Un problema está en P (determinista polinomial) si hay una TM determinista que determina si σ pertenece al lenguaje en un número de pasos acotado por un polinomio en $|\sigma|$.

Un problema está en NP (no determinista polinomial) si hay una TM no determinista que acepta σ en un número de pasos acotado por un polinomio en $|\sigma|$.

Notar que esto puede depender de la “codificación del problema.”

Es claro que $P \subseteq NP$ (todo problema en P está en NP) ¿Es $P = NP$?

Ejemplo de problema NP : Resolver un sudoku, dado que simplemente podemos adivinar dónde colocar todos los números.

Reducción $P \leq_p Q$ (reducción de P a Q en tiempo polinomial p):

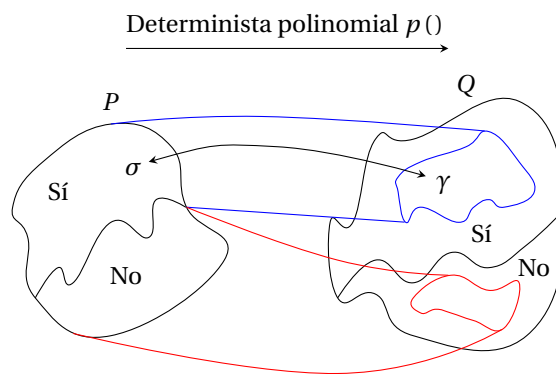


Figura 1: En el futuro, nos interesarán aplicar las reducciones que vimos anteriormente. Note que $|\gamma| \leq p(|\sigma|)$

```
/* Certificado es la solución del problema */
/* Aun falta incorporar los comentarios del profe */
```