Escuela Técnica Superior de Ingeniería Informática

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

etsinf

Computabilidad y Complejidad

Tema 7: Resultados básicos de la Teoría de la Complejidad Computacional

Tema 7: Resultados básicos de la Teoría de la Complejidad Computacional

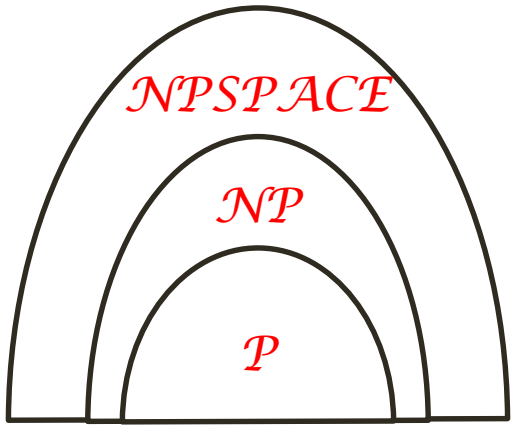
Índice

1. Relaciones entre clases de complejidad polinómicas.
2. El Teorema de Savitch.
3. Aceleración lineal de algoritmos.
4. La reducción de Karp. Problemas completos y duros. Clases complementarias.
5. Relaciones entre clases complementarias.
6. La clase NP.

Bibliografía básica recomendada

- Introduction to automata theory, languages and computation. J.E. Hopcroft, J.D. Ullman, R. Motwani. Ed. Addison-Wesley. 2001.
- Theory of Computational Complexity. S. Du, K. Ko. John Wiley & Sons. 2000
- Computers and intractability : A guide to the theory of NP-completeness. M. Garey, D. Johnson. Ed. W.H. Freeman. 1979.

Relaciones entre clases de complejidad polinómicas



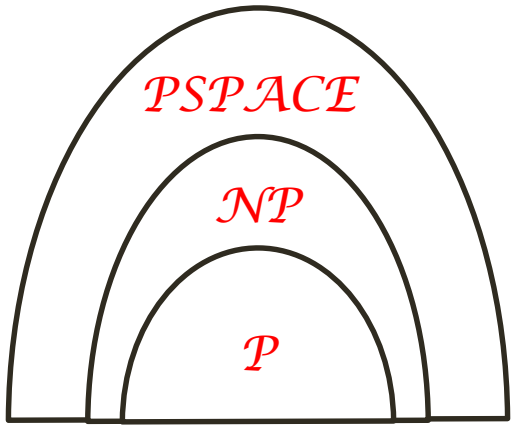
$\mathcal{P}$	**DTIME($n^{O(1)}$)**
$\mathcal{NP}$	**NTIME($n^{O(1)}$)**
$\mathcal{PSPACE}$	**DSPACE($n^{O(1)}$)**
$\mathcal{NPSPACE}$	**NSPACE($n^{O(1)}$)**

$\mathcal{P} \subseteq \mathcal{NP}$ se deduce de que las máquinas de Turing no deterministas incluyen a las deterministas en su definición.

$\mathcal{NP} \subseteq \mathcal{NPSPACE}$ se deduce de que cualquier máquina de Turing no determinista que emplee un tiempo no determinista **$n^{O(1)}$** en aceptar una cadena de entrada no puede explorar más de **$n^{O(1)}$** celdas nuevas y, por lo tanto su complejidad espacial es **NSPACE($n^{O(1)}$)**

¿ Qué relación existe entre $\mathcal{PSPACE}$ y $\mathcal{NPSPACE}$?

Relaciones entre clases de complejidad polinómicas



$\mathcal{P}$	DTIME($n^{O(1)}$)
$\mathcal{NP}$	NTIME($n^{O(1)}$)
$\mathcal{PSPACE}$	DSPACE($n^{O(1)}$)
$\mathcal{NPSPACE}$	NSPACE($n^{O(1)}$)

El Teorema de Savitch (W. Savitch, 1970): Para toda función $f(n) \geq \log_2 n$ y para todo lenguaje L se cumple que si $L \in$ NSPACE(f(n)) entonces $L \in$DSPACE($f(n)^2$)

Corolario $\mathcal{PSPACE}=\mathcal{NPSPACE}$

La demostración del Teorema de Savitch se basa en que se puede simular a una máquina de Turing no determinista mediante una máquina determinista reutilizando el espacio necesario para determinar si podemos alcanzar una descripción instantánea de aceptación (cuya espacio ocupado máximo es f(n)) desde la descripción instantánea inicial.

Walter J. Savitch "Relationships between nondeterministic and deterministic tape complexities", ***Journal of Computer and System Sciences*** **4 (2): 177–192, (1970)**

Relaciones entre clases de complejidad polinómicas

El Teorema de Savitch (W. Savitch, 1970): Para toda función $f(n) \geq \log_2 n$ y para todo lenguaje L se cumple que si $L \in NSPACE(f(n))$ entonces $L \in DSPACE(f(n)^2)$

La clave para demostrar el Teorema de Savitch es establecer un procedimiento que permita reutilizar el espacio. Un procedimiento de ALCANZABILIDAD permite establecer si desde la configuración instantánea S_1 se puede alcanzar la configuración instantánea S_2 en, como máximo, 2^i movimientos en una máquina de Turing no determinista.

```
alcanzable(S1,S2,i)
begin
    si i=0 entonces
            si S1=S2 o existe un movimiento para pasar de S1 a S2
            return(True)
            sino return(False)
     sino begin
            para cada configuración S3 hacer
               si alcanzable(S1,S3,i-1) ∧ alcanzable(S3,S2,i-1)
               entonces return(True)
            fPara
            return(False)
          end
end
```

Relaciones entre clases de complejidad polinómicas

El Teorema de Savitch (W. Savitch, 1970): Para toda función $f(n) \geq \log_2 n$ y para todo lenguaje L se cumple que si $L \in NSPACE(f(n))$ entonces $L \in DSPACE(f(n)^2)$

Podemos emplear el procedimiento de alcanzabilidad para establecer si desde la configuración inicial con una cadena de entrada $S_0=q_0w$ se puede alcanzar una configuración de aceptación con una longitud máxima f(n). El siguiente test así lo establece:

begin
Sea $|w|=n$ y $m=\lceil \log_2 c \rceil$
Sea S_0 la configuración inicial de la máquina de Turing no determinista con entrada w
Para cada configuración instantánea de aceptación S_f con longitud f(n) hacer
alcanzable(S_0,S_f,$m \cdot f(n)$)
end

En el anterior test, c es una constante que garantiza que el número máximo de configuraciones instantáneas de longitud f(n) es $c^{f(n)}$, para cualquier entrada de longitud n.

Es fácil comprobar que:

(1) El procedimiento de alcanzabilidad tiene una complejidad espacial f(n) (para almacenar las configuraciones instantáneas)
(2) El número de llamadas recursivas es como máximo de f(n)

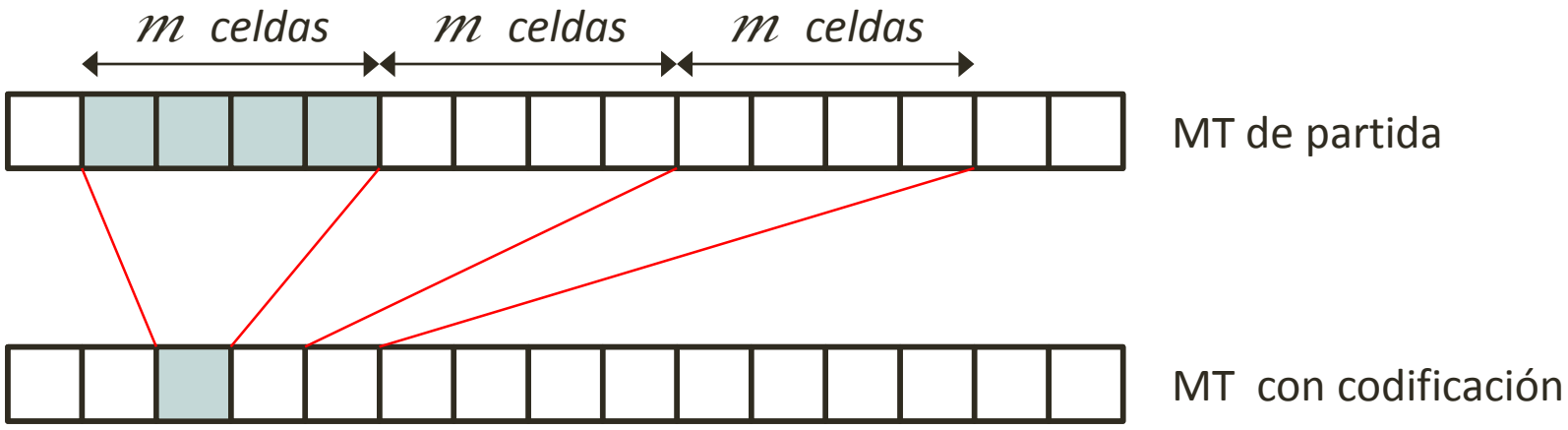
En conclusión, el anterior test se puede establecer con una complejidad espacial $f(n)^2$

Aceleración lineal de algoritmos

Teorema Si el lenguaje L es aceptado por una máquina de Turing multicinta con k cintas y complejidad temporal T(n), entonces para cualquier constante c > 0 L es aceptado por una máquina de Turing multicinta con k cintas y complejidad temporal c·T(n), siempre que k > 1 y

$$\inf_{n\to\infty} \frac{T(n)}{n} = \infty$$

La demostración del teorema se basa en construir una máquina de Turing equivalente a la máquina de Turing de partida pero que actúe con una codificación de su entrada de forma que se comprima el espacio a utilizar y se codifiquen los movimientos empaquetándolos en un ratio de 8/m (siendo m el número de símbolos que se comprime cada vez)



Reducciones entre problemas

La **reducción de Karp** permite transformar unos problemas en otros garantizando que: 1) el tiempo de transformación es polinómico con la talla de las instancias, y 2) las soluciones se conservan (la instancia transformada tiene solución afirmativa sii la instancia original la tiene)

Definición: Sean dos lenguajes L_1 y L_2 definidos sobre el alfabeto Σ y sea la función f computable en tiempo polinómico, total y no inyectiva[1] definida como $f:\Sigma^* \rightarrow \Sigma^*$. Diremos que L_1 es m-reducible polinómicamente a L_2 y lo denotaremos como $L_1 \leq_m^p L_2$ si se cumple el siguiente enunciado

$$(\forall\, x \in \Sigma^*) \quad x \in L_1 \text{ sii } f(x) \in L_2$$

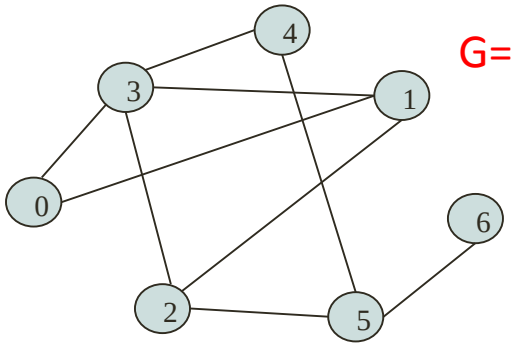
Definición: Sean dos problemas Π_1 y Π_2. Diremos que Π_1 es m-reducible polinómicamente a Π_2 y lo denotaremos como $\Pi_1 \leq_m^p \Pi_2$ si se cumple que bajo cualquier esquema de codificación e $L(\Pi_1,e) \leq_m^p L(\Pi_2,e)$.

[1] Una función inyectiva f cumple que $(\forall\, x_1 \neq x_2)\ f(x_1) \neq f(x_2)$. En el caso de que la función f de la definición fuera inyectiva se obtiene lo que se denomina una reducción "*uno-a-uno*". En nuestro caso la reducción es "*muchos-a-uno*"

Problemas y lenguajes

Ejemplo:

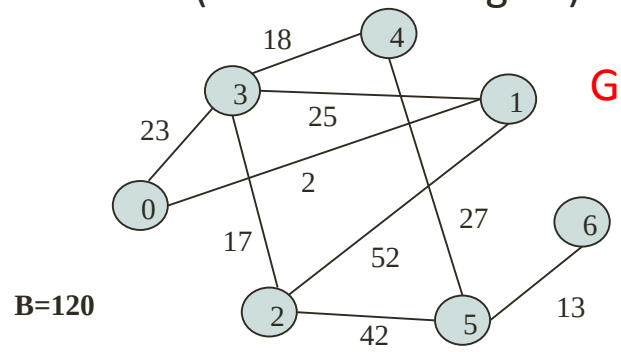
El Camino Hamiltoniano
CH (versión no dirigida)



G=(V,E)

¿ Existe un camino que recorra todos los vértices del grafo una sóla vez ?

El Viajante de Comercio
VC (versión no dirigida)

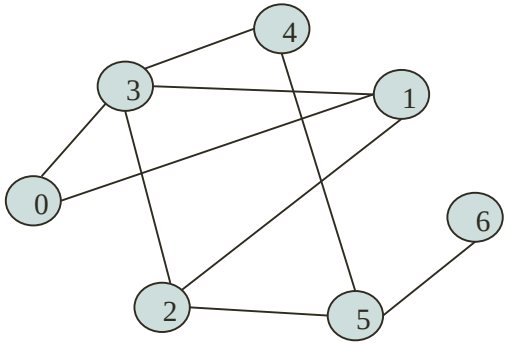


$G=(V,E_p)$

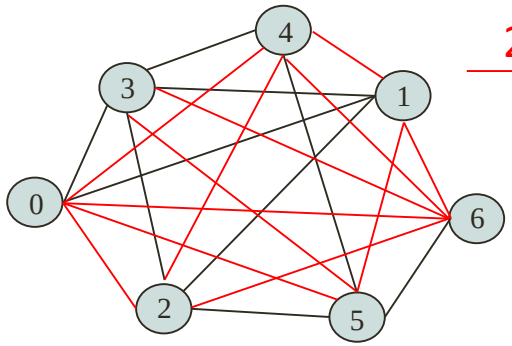
B=120

¿ Existe una ruta que recorra todas las ciudades una única vez y cuya distancia acumulada sea menor o igual a un valor predefinido B ?

$$CH \leq_{m}^{p} VC$$



$\frac{1}{2}$



B=6 (que es |V|-1)

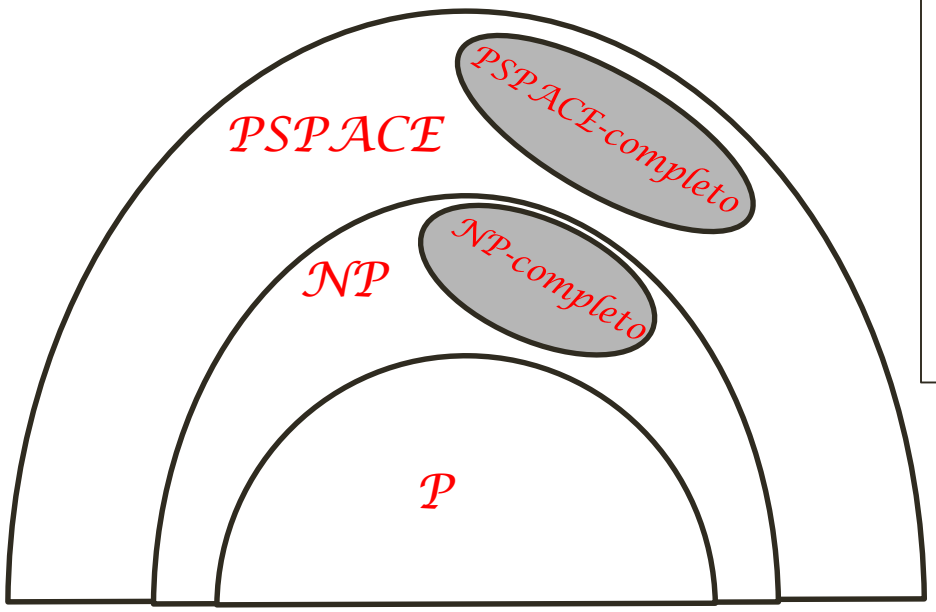
Clases de complejidad completas y duras

Sea C una clase de complejidad.

- Diremos que un lenguaje L es C-duro si se cumple el siguiente enunciado

$$(\forall L' \in C)\ L' \leq_m^p L$$

- Diremos que un lenguaje L es C-completo si L es C-duro y $L \in C$



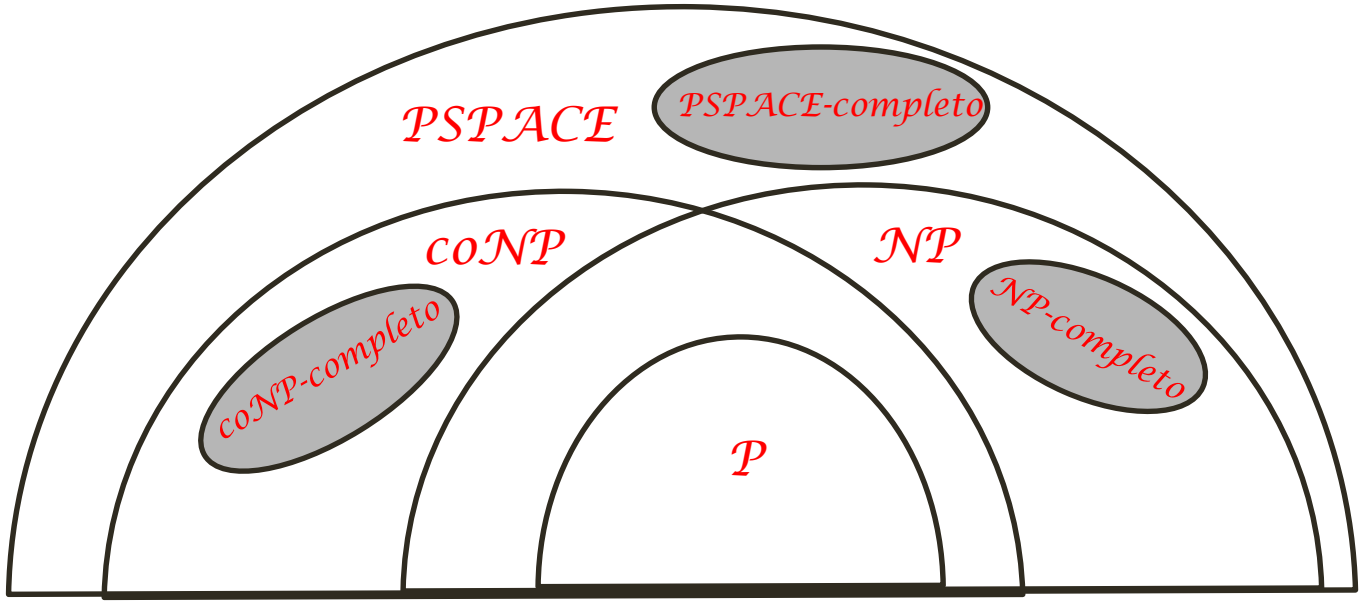
De forma inmediata se puede demostrar que todos los problemas de $\mathcal{P}$ son $\mathcal{P}$-*completos*. Tomemos un lenguaje arbitrario L que pertenezca a $\mathcal{P}$ y que sea distinto del lenguaje vacío y de Σ^* (que denominaremos lenguajes triviales). Tomemos ahora cualquier lenguaje L' no trivial que también pertenezca $\mathcal{P}$. Para reducir L' a L es suficiente con que la reducción a utilizar resuelva el problema L' (mediante un algoritmo determinista en tiempo polinómico) y transforme la cadena de entrada en una cadena de salida predefinida de L si la cadena de entrada pertenece a L' ó en una cadena predefinida que no pertenezca a L si la cadena de entrada no pertenece a L'.

Clases de complejidad complementarias

Sea C una clase de complejidad. La clase complementaria de C, que denotaremos por coC, se define como la clase que contiene aquellos lenguajes cuyos complementarios pertenecen a C.

Lema: Para cualquier función constructiva en tiempo f, se cumple que DTIME(f(n))=coDTIME(f(n)).

Por lo tanto $\mathcal{P}=co\mathcal{P}$



La clase NP

La clase de complejidad $\mathcal{NP}$ se define mediante máquinas de Turing no deterministas con una complejidad temporal polinómica.

Alternativamente, podemos caracterizarla mediante aquellos problemas que se pueden resolver mediante algoritmos no deterministas con una fase de verificación polinómica:

Algoritmo no determinista A_{Π}

Sea Π un problema de decisión sujeto a un conjunto de restricciones R.

Sea S una estructura de soluciones para el problema Π

Fase I: Conjetura

generar_estructura(S)

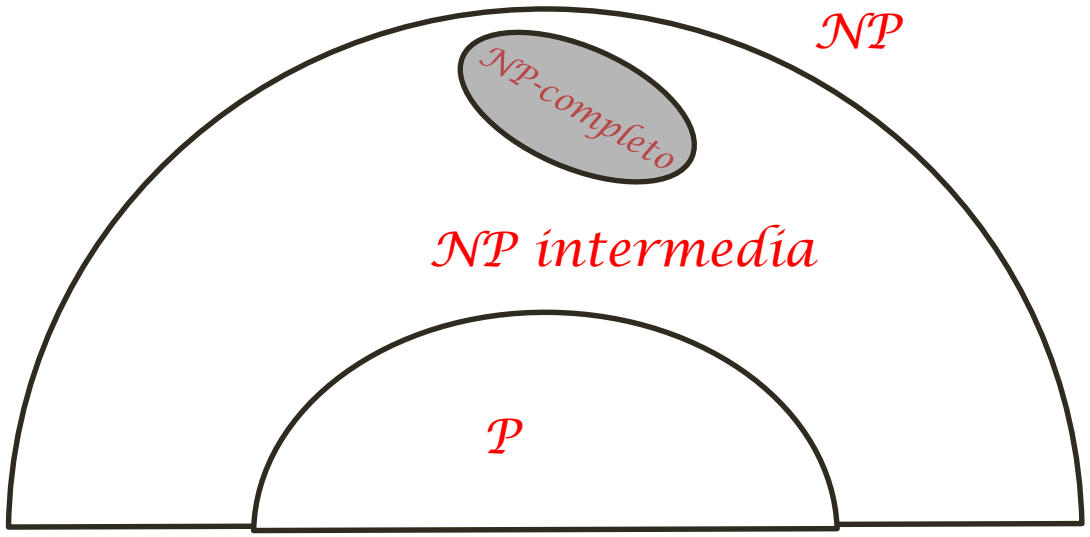
Complejidad lineal en S
Fase no determinista

Fase II: Verificación

verificar(S,R)

Complejidad polinómica en S y R
Fase determinista

La clase NP



La clase de complejidad *NP intermedia* se compone de aquellos problemas de *NP* que no pertenecen a *NP-completo*

Los problemas de la clase NP intermedia pueden clasificarse como problemas de P o de NP-completo. Un ejemplo histórico de un problema de NP intermedia que se demostró en 2004 por Agraval, Kayal, & Saxena que pertenecía a P es el problema de establecer la primalidad de un número natural n (problema PRIMOS).

La clase NP

Algunas caracterizaciones de la clase NP

Proposición $\text{Si } L \in \text{NP-completo} \text{ entonces } \overline{L} \in \text{coNP-completo}$

Proposición $\text{Si } \text{NP} \cap \text{coNP-completo} \neq \varnothing \text{ entonces } \text{NP} = \text{coNP}$

Lema Si existe un lenguaje L tal que $L \in \text{NP-completo}$ y $\overline{L} \in \text{NP}$ entonces $\text{NP} = \text{coNP}$

Una de las cuestiones abiertas más relevantes en la teoría de la complejidad es la de establecer si las clases P y NP son la misma ó no.

Proposición $\text{Si } \text{P} \cap \text{NP-completo} \neq \varnothing \text{ entonces } \text{P} = \text{NP}$

Se han establecido a lo largo del tiempo numerosas condiciones necesarias y suficientes tanto para establecer P=NP como para establecer P≠NP.

Las consecuencias prácticas en el caso P=NP serían muy relevantes (p.ej. habría que redefinir gran parte de la criptografía de clave pública)

P vs. NP

(Cómo ganar dinero mediante la teoría de la complejidad)

El *Clay Mathematics Institute* otorga los *Premios para los Problemas del Milenio*. El 24 de mayo de 2000 se propusieron un total de siete problemas dotados de un millón de dólares US para el que logre resolver alguno de ellos. Entre los problemas seleccionados se encuentra el de proporcionar una respuesta (afirmativa o negativa) al problema abierto de si P=NP.

- http://www.claymath.org/millennium-problems/p-vs-np-problem