# Computabilidad y Complejidad

Otoño 2018

Eduardo Hidalgo García 117036

# Índice general


I **Parte I: Computabilidad**

1 **Alfabetos, Lenguajes, Gráficas y Problemas** *(.5pc).7

**1.1 Introducción a la computabilidad** **7**

**1.2 Alfabetos** **8**

**1.3 Lenguajes** **9**

**1.4 Graficas** **9**

**1.5 Problemas** **10**

2 **Computabilidad** *(.5pc).11

**2.1 Automatas finitos** **11**

2.1.1 Introducción . . . . . 11

2.1.2 Maquina de Turing . . . . . 11

2.1.3 Maquinas de Turing como Algoritmos . . . . . 13

**2.2 Lenguajes Computables Enumerablemente y Computables** **13**

2.2.1 Introducción . . . . . 13

2.2.2 Problemas Representantes de las clases: Computable Enmerablemente y Computable . . . . . 14

2.2.3 Funciones Computables Enmerablemente y Computables . . . . . 14

**2.3 Reducciones computables de lenguajes** **14**

**2.4 Teorema de Rice** **14**

II **Parte II: Complejidad**


3 **Clases de Complejidad** *(.5pc).17

**3.1 Introducción 17**

**3.2 Tiempo, Espacio y fca 17**

**3.3 Teoria Asintotica 18**

3.3.1 Cotas Asintoticas . . . 18

**3.4 Problemas o Lenguajes Comunes en Complejidad 18**

3.4.1 Problema de Satisficibilidad Booleana . . . 18

3.4.2 Problemas de Grafos . . . 19

3.4.3 Problema Numerico . . . 19

3.4.4 Problema de la membresia con tiempo . . . 20

3.4.5 Clases de Complejidad Básicas . . . 20

3.4.6 Familias de Complejidad . . . 20

3.4.7 Conjuntos Complemento . . . 21

3.4.8 Resultados y Teoremas . . . 21

4 **Reducciones y Completes** *(.5pc).23

**4.1 Introducción 23**

**4.2 Reducciones Polinomiales 23**

5 **Problemas Completos y Duros** *(.5pc).25

**5.1 Introducción 25**

**5.2 Teorema de Cook 25**

**Bibliografía** *(.5pc).27

**Libros 27**

**Links 27**

**Clase 27**

**Índice Alfabético** *(.5pc).29

# Parte I: Computabilidad


**1 Alfabetos, Lenguajes, Gráficas y Problemas *(.5pc).7**

1.1 Introducción a la computabilidad
1.2 Alfabetos
1.3 Lenguajes
1.4 Graficas
1.5 Problemas

**2 Computabilidad *(.5pc).11**

2.1 Automatas finitos
2.2 Lenguajes Computables Enumerablemente y Computables
2.3 Reducciones computables de lenguajes
2.4 Teorema de Rice

# 1. Alfabetos, Lenguajes, Gráficas y Problemas

## 1.1 Introducción a la computabilidad

En este capítulo se enunciaran las nociones básica para la construcción de la teoría de la complejidad. Los *problemas* y *algoritmos* se pueden formalizar y analizar matematicamente, por ejemplo mediante su conversión a *lenguajes* y la *capacidad de decidir* sobre ellos de una *Máquina de Turing*.

La noción de *complejidad computacional* requiere de la existencia de algoritmos y con ellos la noción de problemas (más adelante veremos que tambien se les puede llamar *lenguajes*) posibles de *computar*. En la **parte II** veremos que la noción de *complejidad* es una propiedad **deseable** de los problemas posibles de computar, pues esta relacionada con la capacidad de en la práctica poder resolver un problema.

Con este fin se desarrolla la teroria de *lenguaje* perteneciente a algún alfabeto. Además con base en la noción de *problema de deicisón* se plantea la equivalencia, antes mencionada, entre *problemas de deicisión* y *lenguajes*

En resumen, al final de este capitulo se plantea la idea de que cualquier problema puede traducirse a una cadena de caracteres de algún *lenguaje* (entenderemos *lenguaje* de manera informal como el conjunto de todas las cadenas de caracteres de tamaño finito pertencientes a algún alfabeto) y generar una *Maquina de Turing* que, en el mejor de los casos, pueda decidir sí la cadena pertecenece o no al *lenguaje aceptado* por esta.

## 1.2 Alfabetos

**Definición 1.2.1** *Alfabeto:* **Conjunto finito de simbolos** $\Sigma : |\Sigma| < \infty$.

**Definición 1.2.2** *Simbolos o Letras:* $x \in \Sigma : |\Sigma| < \infty, |x| = 1$.

**Definición 1.2.3** *Cadena o Palabra:* **Secuencia de simbolos de tamaño n**

$$X = x_1, .., x_n : \forall i \in [1, n], X_i \in \Sigma, n < \infty$$

.

**Definición 1.2.4** *Longitud de una cadena:* $|X| \in \mathbb{N} : |X| = n$.

**Definición 1.2.5** *Cadena Vacia:* $\lambda \in \Sigma : |\lambda| = 0, !\exists \lambda \in \Sigma$.

**Definición 1.2.6** *Cerradura de un alfabeto:* **Conjunto de cadenas finitas**

$$\Sigma^* \in \Sigma : \Sigma^* = \{X : |X| < \infty, x \in X \Leftrightarrow x \in \Sigma\}$$

.

**Definición 1.2.7** *Concatenación de cadenas:* **Operaciones con símbolos**

$$XY \in \Sigma^* : |X| = n, |Y| = m, n, m < \infty \Rightarrow XY = x_1, ..., x_n, y_1, ..., y_m$$

.

**Propiedad 1.2.1** *Cerradura de la concatenación de cadenas:*

$$\forall X, Y \in \Sigma^* \Rightarrow XY \in \Sigma^*$$

.

**Propiedad 1.2.2** *Asociatividad de la concatenación de cadenas:*

$$\forall X, Y, Z \in \Sigma^* \Rightarrow (XY)Z = (x_1 ... x_m y_1 ... y_n) z_1 ... z_l = x_1 ... x_m (y_1 ... y_n z_1 ... z_l) = X(YZ), con: m, n, l < \infty$$

.

**Definición 1.2.8** *Monoide o Casi-grupo:* **Dupla** $(A, \circ)$ talque:

$$A \in \Sigma^*$$

,

$$\circ \; Operación : \Sigma^* \to \Sigma^*$$

*Dicha operación cumple: Cerradura, Asociatividad y Elemento neutro*

.

Dentro del contexto de *computabilidad* los candidatos ideales para dicha dupla son los elementos recien definidos:

- Alfabeto: $\Sigma$
- Concatenación de cadenas: $XY$

Por lo que se dice que la dupla de elementos $(\Sigma, XY)$ forman un *monoide o casi-grupo*.

## 1.3 Lenguajes

**Definición 1.3.1** *Lenguaje:* **Subconjunto de la cerradura de un alfabeto**

$$\{A : A \subseteq \Sigma^*\}$$

.

**Definición 1.3.2** *Concatenación de lenguajes:* **Operaciones con lenguajes**

$$AB = \{XY : X \in A, Y \in B, A, B \subseteq \Sigma^*\}$$

.

**Definición 1.3.3** *Cerradura o estrella de Kleene:* **Cerradura de un lenguaje**

$$A^* = \bigcup_{i=0}^{\infty} A^i : A^0 = \{\lambda\}, A^{n+1} = A^n A$$

.

**Propiedad 1.3.1** *Idempotencia de la cerradura de Kleene:*

$$A^* A^* = A^*, \forall A \subseteq \Sigma^*$$

.

**Propiedad 1.3.2** *Lenguaje neutro:* $A^0 = \{\lambda\} : A^* = A^0 \cup A^*$.

**Definición 1.3.4** *Codificación válida de B a partir de A:* **Función de alfabetos**

$$v : \Gamma \to \Sigma_2^*, \Gamma \subseteq \Sigma_1^*, \text{con: } v \textit{ sobreyectiva}$$

.

En resumen, en esta seección vimos como los lenguajes son conjuntos de elementos -conjuntos de cadenas de simbolos de distinta longitud- de la cerradura del alfabeto, que respetan las propiedades asociadas a este. Ademas vimos que es posible definir funciones $v(a) = b$ que mapeen todos (o algunos) elementos de la cerradura de un alfabeto $\Gamma \subseteq \Sigma_1^*$ a la cerradura de otro $\Sigma_1^*, \Sigma_2^*$.

## 1.4 Graficas

**Definición 1.4.1** *Gráfica:* **Grafo no dirigido** Pareja ordenada $G = (V, E)$ con:

$$\textit{(Vertices) } V = \{v_1, ..., v_n\}$$

.

$$\textit{(Aristas) } E = \left\{\overline{v_i v_j}, \forall i, j \in [1, n], i \neq j\right\}$$

.

**Definición 1.4.2** *Gráfica Dirigida:* **Grafo dirigido** Pareja ordenada $G = (V, E)$ con:

$$\textit{(Vertices) } V = \{v_1, ..., v_n\}$$

.

$$\textit{(Arcos) } E = \left\{\overrightarrow{v_i v_j}, \forall i, j \in [1, n], i \neq j\right\}$$

.

**Definición 1.4.3** *Matriz de Adyacencia:* **Matriz simétrica** $A = (a_{i,j})$ con: $i, j = 1, 2, ..., n$ donde

$$a_{i,j} = \begin{cases} 1 & \textit{sii es adyacente a } j \\ 0 & \textit{en otro caso} \end{cases}$$

.

**Definición 1.4.4** *Conjunto independiente de vertices de un grafo:* $\left\{I = \{v_1, ..., v_l\} : \forall v_i, v_j \in I \Rightarrow v_i, v_j \notin E\right\}$.

## 1.5 Problemas

**Definición 1.5.1** *Problema de decisión:* **Funcion con output de un bit:** 'sí' o 'no'

$$f : A \to \{1,0\} \text{ con: 'sí-instancias' } Y \subseteq A$$

.

En esta sección se explora la idea de convertir una instancia de un problema, por ejemplo una gráfica dada, a una cadena de símbolos en el alfabeto $\{1,0\}$. Con el objetivo de **computar** esta cadena para saber si se 'acepta' o se 'rechaza'. Existe una correspondencia donde la 'aceptación' corresponde a una 'sí-instacia' y el 'rechazo' a una 'no-instancia'.

**Instancia** $\to$ **Traducción a Lenguaje** $\to$ **Lectura de Máquina** $\to$ **Acepta/Rechaza**

A continuación se definen problemas interesantes desde la perspectiva de la computabilidad.

- Alcanzabilidad de vertices en una gŕafica *REACHABILITY*
- Conjunto de vertices independientes de tamaño máximo $\langle G,k\rangle$
- Problema (Optimización) del flujo máximo *MAXIMUM FLOW*
- Problema (Optimización) del agente viajero *TSP*

# 2. Computabilidad

## 2.1 Automatas finitos

### 2.1.1 Introducción

El concepto de **Automata finito** es el modelo bajo el cual se busca capturar a los sistemas del tipo *finite-state transistion system*, sistemas con número de estados finitos.

**Definición 2.1.1** *Automata Finito Deterministico* **(5-tupla):** $M = (Q, \Sigma, \delta, q_0, q_y)$ donde:

- Conjunto Finito de Estados: $Q$
- Alfabeto: $\Sigma$
- Función de transición: $\delta(q_i, x) = q_{i+1} : Q \times \Sigma \to Q$
- Estado Inicial: $q_0 \in Q$
- Estado de Aceptación: $q_y \in Q$

### 2.1.2 Maquina de Turing

En esta sección se introduce el modelo del automata mas utilizado en la ciencia de la computación. Conceptualizado en 1935 por Alan Turing, la máquina de turing (MT) es una abstracción de una máquina con una estructura única de datos (de hecho una muy primitiva, pues son cadenas de caracteres). Las operaciones que puede realizar tambien son muy sencillas, pues puede: mover el cursor a la derecha o a la izquierda para escribir en la posición actual y moverse deacuerdo al simbolo actual.

No obstante, parecer un modelo de computo muy sencillo, a lo largo del capitulo se argumenta como éste es capas de expresar cualquier algorimo y simular cualquier lenguaje de programación

**Definición 2.1.2** *Maquina de Turing* **MT (9-tupla) :** $M = (Q, \Sigma, \delta, q_0, q_y)$ donde:

- Conjunto Finito de Estados: $Q$
- Alfabeto del input: $\Sigma$
- Alfabeto de cinta: $\Sigma \subseteq \Gamma$
- Programa o Función de terminación: $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L\}$
- Símbolo de inicio de la cinta: $\vdash \in \Gamma$
- Símbolo de espacio en blanco: $\sqcup \in \Gamma$
- Estado Inicial: $q_0 \in Q$
- Estado de Aceptación: $q_y \in Q$
- Estado de Rechazo: $q_n \in Q$

**Definición 2.1.3** *Configuración de una MT:* **(Descripción instantanea)** $(q, X, n) \in Q \times \Gamma^* \times \mathbb{N}$.
Se dice: '$M$ está en el estado $q$ viendo el *n-ésimo* elemento de $X$'

**Definición 2.1.4** *Siguiente configuración de una MT:*
**(Descripción instantanea despues de una acción de la MT)** $(q, X, n) \underset{1}{\to} (q, X, n+1) \in Q \times \Gamma^* \times \mathbb{N}$.

**Definición 2.1.5** *Maquina de Turing No Deterministica* **MTND:**
$N = (Q, \Sigma, \Gamma, R, \vdash, \sqcup, q_0, q_y, q_n$ *con* $R \in Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$ *relación* .

**Definición 2.1.6** *Relación entre estados* **Correspondencia (R):** $R \subseteq Q_1 \times Q_2 \times \cdots \times Q_n$

**Definición 2.1.7** *Maquina de Turing que calcula funciones de cadenas* **MT (6-tupla):**
$M = (Q, \Gamma, \delta, q_0, q_h, q_e)$ con $Q, q_0, \Gamma, \delta$ igual a la definición 2.1.2, además:

- Estado de detención: $q_h$
- Estado de error: $q_e$
- Función de cadenas: $f : \Sigma^* \to \Sigma^*$

**Definición 2.1.8** *Maquina de Turing Universal* **(U):** $M \in MT : \forall M' \in MT, MsimuleaM'$ con:

- Alfabeto: $\Sigma = \{0, 1, \#\}$
- Instancias (inputs): $\{M\#X : M$ *es la descripción de la MT* $a\}$
- Función de cadenas: $f : \Sigma^* \to \Sigma^*$

**Definición 2.1.9** *Máquina de Turing Total* **'Siempre para' (MTT) :**
$\left\{ M \in MT : \forall X \in \Sigma^*, M(X) \underset{n}{\overset{M}{\longrightarrow}} q_h, q_h \in \{q_h, q_n\} \right\}$

**Definición 2.1.10** *Máquina de Turing con 2-cintas* **Input y Output :**

$\{M \in MT : M$ *recibe la cadena* $X$ *en la cinta READ ONLY, calcula el output en la cinta WRITE ONLY*$\}$

**Definición 2.1.11** *Máquina de Turing con k-cintas:*

$$\{M \in MT : M \text{ tiene } k \text{ cintas para realizar sus operaciones}\}$$

### 2.1.3 Maquinas de Turing como Algoritmos

Las máquinas de Turing parecen las candidatas ideales para resolver cierto tipo de problemas sobre cadenas de caracteres. En particular, para *computar* **funciones de cadenas** y *aceptar* o *decidir* sobre lenguajes. En este apartado se define los conceptos para definir matematicamente, la tarea anterior.

**Definición 2.1.12** *Lenguaje aceptado por una máquina de Turing:* $L(M) = \left\{A \in \Sigma^* : M(A) \xrightarrow[n]{M} q_y\right\}$

**Definición 2.1.13** *Lenguaje aceptado por una máquina de Turing Universal (U):* $L(U) = \{M\#X : M \text{ acepta a } X\}$

**Tesis 2.1.1** *Tesis de Church-Turing:* **Todo problema que puede ser resuelto (i.e llegar a su** $q_A$ **) en un número polinomial de pasos, puede ser resuelto por una MTD**

## 2.2 Lenguajes Computables Enumerablemente y Computables

### 2.2.1 Introducción

Al finalizar la sección anterior estudiamos la **Tesis de Church-Turing** que permite caracterízar la complejidad en función del modelo de computación generado por la **MT**.
En la historia se han planteado vaios modelos de computo, por ejemplo:

- *Post systems*
- $\mu - recursive\ functions$
- $\lambda - calculus$
- *Combinatory logic*

Aunque todos son distintos es posible *emular* a travéz del modelo de la **MT** lo que hace cualquiera de los otros.
La noción de lenguaje computable no se define en relación a un cierto modelo, pues todos son equivalentes. Sin embargo, en este texto se eligió al modelo de la máquina de Tuing **MT** pues es el que mejor captura la esencia de ser computable.

**Definición 2.2.1** *Lenguaje Computable Enumerablemente* **'Siempre para en caso de aceptar' :**

$$\mathscr{CE} = \left\{A \subseteq \Sigma^* : \forall X \in L(M), \exists M \in MT, M(X) \xrightarrow[n]{M} q_y\right\}$$

**Definición 2.2.2** *Lenguaje Co-computable Enumerablemente* **'Su alfabeto compelemento siempre para en caso de aceptar' :**

$$\{L \in \Sigma^* : \Sigma^* - L \in \mathscr{CE}\}$$

**Definición 2.2.3** *Lenguaje Computable* **'Siempre para' :**

$$\mathscr{C} = \left\{A \subseteq \Sigma^* : \forall X \in \Sigma^*, \exists M \in MT, M(X) \xrightarrow[n]{M} q_h, q_h \in \{q_h, q_n\}\right\}$$

**Resultado 2.2.1** $\mathscr{CE} \subseteq 2^{\Sigma^*}$

**Resultado 2.2.2** $\mathscr{CE} \neq \emptyset$

**Resultado 2.2.3** $\mathscr{C} \subseteq \mathscr{CE}$

### 2.2.2 Problemas Representantes de las clases: Computable Enmerablemente y Computable

**Definición 2.2.4** *Problema de detención* **(HP) :**

$$HP = \{M\#X : M \text{ se detiene en } X\}$$

**Definición 2.2.5** *Problema de la membresia* **(MP) :**

$$MP = \{M\#X : X \in L(M)\}$$

**Resultado 2.2.4** $HP \notin \mathscr{C}$

**Resultado 2.2.5** $HP \in \mathscr{CE}$

**Resultado 2.2.6** $MP \notin \mathscr{C}$ se muestra vía:$(HP \leq_m MP)$

### 2.2.3 Funciones Computables Enmerablemente y Computables

**Definición 2.2.6** *Función efectivamente computable :*

$$\left\{\sigma : \{0,1\}^* \to \{0,1\}^* : \forall X \in \{0,1\}^* \exists M \in MT, M \xrightarrow[n]{M} q_h \Leftrightarrow \sigma(X) \in (q,X,n)\right\}$$

**Definición 2.2.7** *Función computable :*

$$\left\{\sigma : \{0,1\}^* \to \{0,1\}^* : \forall X \in \{0,1\}^* \exists M \in MT, M \xrightarrow[n]{M} q_h \Leftrightarrow \sigma(X) \in (q,X,n), \sigma \text{ sea computable por una MTT}\right\}$$

## 2.3 Reducciones computables de lenguajes

En esta sección se estudia una relación entre distintos lenguajes llamada **reducción muchos a uno**, esta permite 'mapear' de un lenguaje a otro. Además permite establecer nociones de comparabilidad entre conjuntos en términos de su *complejidad*.
En esta sección dicha relación se estudia exclusivamente desde el punto de vista de las nociones aprendidas de *computabilidad* (i.e. $\mathscr{C}, \mathscr{CE}$. Ademas, con base en ella se enuncian resultados sobre conjuntos comunes en términos de *computabilidad*.

**Definición 2.3.1** *Reducción (función) muchos a uno de A en B :* $\sigma$ dado $A \subseteq \Sigma_1^*, B \subseteq \Sigma_2^*$

$$\{\sigma : \Sigma_1^* \to \Sigma_2^* : \forall X \in \Sigma_1^*, X \in A \Leftrightarrow \sigma(X) \in B\}$$

**Definición 2.3.2** *Reducción de lenguajes :* $A \leq_m B \Leftrightarrow \exists \sigma \in REDUCCION$

**Resultado 2.3.1** $\leq_m$ es *transitiva*

**Resultado 2.3.2** $\leq_m$ es *reflexiva*

**Resultado 2.3.3** $FIN = \{N : |L(N)| \leq \infty\} \notin \mathscr{CE}$ se muestra vía:$(HP^c \leq_m FIN)$

**Resultado 2.3.4** $FIN^c = \{N : |L(N)| = \infty\} \notin \mathscr{CE}$ se muestra vía:$(HP^c \leq_m FIN^c)$

## 2.4 Teorema de Rice

**Teorema 2.4.1** *Teorema de Rice:* Toda propiedad $\mathbb{P}$ NO *trivial* de los conjuntos $\mathscr{CE}$ es NO computable (i.e $\mathbb{P} \notin \mathscr{C}$)

# Parte II: Complejidad


**3 Clases de Complejidad** *(.5pc).17
3.1 Introducción
3.2 Tiempo, Espacio y fca
3.3 Teoria Asintotica
3.4 Problemas o Lenguajes Comunes en Complejidad

**4 Reducciones y Completes** *(.5pc).23
4.1 Introducción
4.2 Reducciones Polinomiales

**5 Problemas Completos y Duros** *(.5pc).25
5.1 Introducción
5.2 Teorema de Cook

**Bibliografía** *(.5pc).27
Libros
Links
Clase

**Índice Alfabético** *(.5pc).29

# 3. Clases de Complejidad

## 3.1 Introducción

En esta este capítulo nos concentramos en el estudio del *tiempo y espacio* usado por un algoritmo en el modelo de la maquina de Turing. Para ello se definen conceptos como *función de complejidad apropiada* y nuevos conjuntos de lenguajes.
Para el estudio de la complejidad se parte del supuesto que los lenguajes en cuestión forman parte de los lenguajes computables. Es decir en este capítulo sólo se estudiaran leguajes $A : A\mathscr{C}$. Y, el modelo dado que el modelo de computo que emplearemos es la máquina de Turing, dentro del capitulo es válido asumir que: *dado* $A \in \Sigma^*$ *se cumple que:* $\forall X \in \Sigma^*, \exists M \in MT, M(X) \xrightarrow[n]{M} q_h, q_h \in \{q_h, q_n\}$
En este capítulo el modelo de computo de la máquina de Turing surge una modificación. El modelo que utilizaremos es el de la definición 2.1.11 Máquina de Turing con 2-cintas Input y Output.
Al finalizar el capitulo se definiran nuevas clases de complejidad (todas dentro del ámbito de los lenguajes o problemas computables), con base en las definiciones de tiempo y espacio que se definen en la siguiente sección.

## 3.2 Tiempo, Espacio y fca

**Definición 3.2.1** *Tiempo de decisión:* $|\delta| =$ *número de transiciones, tales que:*

$$M(X) = \begin{cases} q_y & , si\ X \in L(M) \\ q_n & , si\ X \notin L(M) \end{cases}$$

**Definición 3.2.2** *Espacio de decisión:* $\#_{max}\sqcup$ usadas por $M$ para decidir si $X \in L(M)$ o $X \notin L(M)$ en la cinta **output**.

**Definición 3.2.3** *Funcion de complejidad apropiada* **(fca):** $f \in \mathbb{F} : \forall n \in \mathbb{N}$ *se cumple:*

- *La función es positiva:* $f(n) > 0$
- *La función es creciente* $f(n) < f(n+1)$
- $\exists M \in MTD : M(f(n) \xrightarrow[kf(n)]{M} f(n), k > 0)$

## 3.3 Teoria Asintotica

De manera similar que en la estadistica el uso de la teoría asintotica sirve para analizar las propiedades del objeto de interés (en este caso un algorítmo) en su forma *estable*.
La definición 2.1.11 función de complejidad apropiada será la base para el estudio del comportamiento del algoritmo en *el largo plazo* y las definiciones 3.2.2 espacio y 3.2.1 tiempo seran el marco para definir la eficiencia del mismo.

### 3.3.1 Cotas Asintoticas

**Definición 3.3.1** *Cota Superior o Inferior al tiempo de decisión:*

$|\delta^{max/min}| = n^{max/min} : \forall f : \mathbb{N} \to \mathbb{N}, f$ *es el pero/mejor caso de ejecución en el que M decide si* $X \in L(M)$ *o* $X \notin L(M)$

.

**Definición 3.3.2** *Cota Asintótica* $g(n)$ *de* $f(n)$:

$$\Theta(g(n)) \equiv g(n) = \left\{ f : \mathbb{N} \to \mathbb{N}, \exists n_0 \in \mathbb{N}, \exists (c_1, c_2) \in R^+, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0 \right\}.$$

.

**Definición 3.3.3** *Cota Superior Asintótica* $g(n)$ *de* $f(n)$:

$$\mathcal{O}(g(n)) \equiv g(n) = \left\{ f : \mathbb{N} \to \mathbb{N}, \exists n_0 \in \mathbb{N}, \exists c \in R^+, 0 \leq f(n) \leq c g(n), \forall n \geq n_0 \right\}.$$

.

**Definición 3.3.4** *Cota Inferior Asintótica* $g(n)$ *de* $f(n)$:

$$\mathcal{O}(g(n)) \equiv g(n) = \left\{ f : \mathbb{N} \to \mathbb{N}, \exists n_0 \in \mathbb{N}, \exists c \in R^+, 0 \leq c g(n) \leq f(n), \forall n \geq n_0 \right\}.$$

.

## 3.4 Problemas o Lenguajes Comunes en Complejidad

### 3.4.1 Problema de Satisficibilidad Booleana

**Definición 3.4.1** *Literales o afirmaciones sobre* $\Sigma^* \{A \in \Sigma^*\}$.

**Definición 3.4.2** *Literales negativas o negaciones sobre* $\Sigma^* \left\{ \vec{A} \in \Sigma^* \right\}$.

**Definición 3.4.3** *Asignación de verdad:* $t : \Sigma^* \to \{V, F\}$, por ejemplo:

$$t(A) \begin{cases} VERDADERO & , si\ t(A) = V \\ FALSO & , si\ t(A) = F \end{cases}$$

.

**Definición 3.4.4** *Clausula sobre* $\Sigma^*$ **conjunto de afirmaciones:** $C = (A_1 \vee A_2 \vee \cdots \vee A_n)$

**Definición 3.4.5** *Clausula Satisficible sobre* $\Sigma^*$ **(Clausula con al menos una literal verdadera):** $C = (A_1 \vee A_2 \vee \cdots \vee A_n), \exists A_i tal\ que:\ t(A_i) = V$

**Definición 3.4.6** *Formulas o clausula Booleana:*

$$\{\Phi \in \{V,F\} : \Phi = t(C)\ con\ C\ clausula\}$$

**Definición 3.4.7** *Formulas Normal Conjuntiva* **FNC:**

$$\left\{\Phi \in \{V,F\} : \Phi = t(C) = \bigvee_{j=1}^{m} C_j = C_1 \vee C_2 \vee \cdots \vee C_m\right\}$$

**Definición 3.4.8** *Formulas Normal Conjuntiva Satisficible* **FNCS:**

$$\{\Phi \in \mathbf{FNC} : \exists t(A_i) = F\ o\ t(A_i) = V, \forall A_i \in C \Leftrightarrow \Phi(C) = V\}$$

**Definición 3.4.9** *SAT:* $SAT = \{A_1.A_2,...,A_n : n \leq 1\ y\ \phi\ es\ satisficible\}$

**Definición 3.4.10** *kSAT:* $kSAT = \{\phi \in FP : \phi\ tenga\ k\ literales\ por\ clausula\ y\ sea\ satisficible\}$

### 3.4.2 Problemas de Grafos

Dado un grafo $G = (V,E) \in \mathbb{G}$, se define:

**Definición 3.4.11** *Coloración de los Vérices de un grafo no dirigido:*

$$\{c : V \to \{1,2,3,...,k\} : \forall y,z \in E, y \neq z \to c(y) \neq c(z)\}$$

.

**Definición 3.4.12** *Grafo o grafica colorable :*

$$\{G \in \mathbb{G}, E \in \mathbb{E} : \exists c : V \to \{1,2,3,...,k\}, \forall y,z \in E, y \neq z \to c(y) \neq c(z)\}$$

.

**Definición 3.4.13** *Alcanzabilidad de dos vertices* $(REACHABILITY)$ :

$$\{Dado\ G = (V,E)\ v_i, v_j \in REACHABILITY \Leftrightarrow \exists v_i v_j \in E\}$$

.

**Definición 3.4.14** *Menor grafo completo dentro de un grafo* $(CLIQUE)$ :

$$G_{min} = \{(U,D) \subseteq (V,E) : \forall u_i, u_j \in U, \exists u_i u_j \in D\}$$

.

**Definición 3.4.15** *Menor numero de nodos que contiene almenos un lado por arista* $(CUBIERTA)$ :

$$N_{min} = \{N \subseteq V(G) : \forall u_i, u_j \in G, \exists u_i \in N \cup u_j \in N\}$$

.

### 3.4.3 Problema Numerico

**Definición 3.4.16** *Problema Numérico* **(SNAPSACK) :**

$$SNAPSACK = \{(N,w,k) : \exists S \subseteq N, \Sigma_{i \in S} v_i \leq k\ s.a.\ \Sigma_{i \in S} w_i \leq w\}$$

.

### 3.4.4 Problema de la membresia con tiempo

**Definición 3.4.17** *Problema de la membresia con tiempo* $(MP_f)$ :

$$MP_f = \left\{ M\#X : q_h = q_y \; si \; X \in L(M) \bigcup q_h = q_n \; si \; n > t, n = \#\delta \right\}$$

.

### 3.4.5 Clases de Complejidad Básicas

**Definición 3.4.18** *Conjunto de lenguajes* **(TIME** $f(n)$ **):**

$$TIME = \left\{ X \in \Sigma : \exists M \in MTD_k, M(X) \xrightarrow[f(n)]{M} q_h, q_h \in \{q_y, q_n\} \right\}$$

.

**Definición 3.4.19** *Conjunto de lenguajes* **(NTIME** $f(n)$ **):**

$$NTIME = \left\{ X \in \Sigma : \exists M \in MTND_k, M(X) \xrightarrow[f(n)]{M} q_h, q_h \in \{q_y, q_n\} \right\}$$

.

**Definición 3.4.20** *Conjunto de lenguajes* **(SPACE** $f(n)$ **):**

$$SPACE = \left\{ X \in \Sigma : \exists M \in MTD_k, \#_{L(M)} \sqcup = f(n) \right\}$$

.

**Definición 3.4.21** *Conjunto de lenguajes* **(NSPACE** $f(n)$ **):**

$$SPACE = \left\{ X \in \Sigma : \exists M \in MTND_k, \#_{L(M)} \sqcup = f(n) \right\}$$

.

### 3.4.6 Familias de Complejidad

**Definición 3.4.22** *Conjunto de Conjuntos de lenguajes* $(P = \bigcup_{k>0} TIME(n^k)$

$$P = \bigcup_{k>0} \left\{ X \in \Sigma^* : \exists M \in MTD_k, M(X) \xrightarrow[n^k]{M} q_h, q_h \in \{q_y, q_n\} \right\}$$

.

**Definición 3.4.23** *Conjunto de Conjuntos de lenguajes* $(NP = \bigcup_{k>0} NTIME(n^k)$

$$NP = \bigcup_{k>0} \left\{ X \in \Sigma^* : \exists M \in MTND_k, M(X) \xrightarrow[n^k]{M} q_h, q_h \in \{q_y, q_n\} \right\}$$

.

**Definición 3.4.24** *Conjunto de Conjuntos de lenguajes* $(L = \bigcup_{k>0} TIME(log(n))$

$$L = \bigcup_{k>0} \left\{ X \in \Sigma^* : \exists M \in MTD_k, M(X) \xrightarrow[log(n)]{M} q_h, q_h \in \{q_y, q_n\} \right\}$$

.

**Definición 3.4.25** *Conjunto de Conjuntos de lenguajes* $(NL = \bigcup_{k>0} TIME(log(n))$

$$NL = \bigcup_{k>0} \left\{ X \in \Sigma^* : \exists M \in MTND_k, M(X) \xrightarrow[log(n)]{M} q_h, q_h \in \{q_y, q_n\} \right\}$$

.

**Definición 3.4.26** *Conjunto de Conjuntos de lenguajes* $(PSPACE = \bigcup_{k>0} SPACE(n^k)$

$$PSPACE = \bigcup_{k>0} \left\{ X \in \Sigma^* : \exists M \in MTD_k, \#_{L(M)} \sqcup = n^k \right\}$$

.

**Definición 3.4.27** *Conjunto de Conjuntos de lenguajes* $(NPSPACE = \bigcup_{k>0} NSPACE(n^k)$

$$NPSPACE = \bigcup_{k>0} \left\{ X \in \Sigma^* : \exists M \in MTND_k, \#_{L(M)} \sqcup = n^k \right\}$$

.

**Definición 3.4.28** *Conjunto de Conjuntos de lenguajes* $(EXP = \bigcup_{k>0} TIME(2^{nk})$

$$EXP = \bigcup_{k>0} \left\{ X \in \Sigma^* : \exists M \in MTD_k, M(X) \xrightarrow[2^{nk}]{M} q_h, q_h \in \{q_y, q_n\} \right\}$$

.

### 3.4.7 Conjuntos Complemento

**Definición 3.4.29** *Complemento del lenguaje aceptado por una MT* $(L(M) - co)$

$$L(M) - co = \left\{ A \in \Sigma^* : \exists M \in MTND_k, M(A) \xrightarrow[log(n)]{M} q_N \right\}$$

.

**Definición 3.4.30** *Complemento de una clase de complejidad* $(co - \mathscr{C})$

$$(co - \mathscr{C}) = \{ A \subseteq \Sigma^* : A - co \in \mathscr{C} \}$$

.

### 3.4.8 Resultados y Teoremas

**Resultado 3.4.1** *Si* $f : \mathbb{N} \to \mathbb{N}$ *y* **fca** $\Rightarrow TIME(f(n)) = co - TIME(f(n))$

**Resultado 3.4.2** *Si* $f : \mathbb{N} \to \mathbb{N}$ *y* **fca** $\Rightarrow SPACE(f(n)) = co - SPACE(f(n))$

**Resultado 3.4.3** $co - P = P; co - L = L; co - PSPACE = PSACE; co - EXP = EXP$

**Teorema 3.4.1** *Teorema del aceleramiento lineal:* Sea $A \in TIME(f(n)) \Rightarrow \forall \varepsilon > 0, A \in TIME(f_\varepsilon(n))$ donde $f_\varepsilon(n) = \varepsilon f(n) + n + 2$

**Resultado 3.4.4** $MP_f \in TIME(f(n^3)$

**Resultado 3.4.5** $MP_f \notin TIME(f(\lfloor \frac{n}{2} \rfloor)$

**Teorema 3.4.2** *Teorema de la jerarquia del tiempo:* Sea $f(n)$ **fca** , $f(n) > n \Rightarrow TIME(f(n)) \subsetneq TIME(f(2n+1))$

**Teorema 3.4.3** *Teorema de la jerarquia del espacio:* Sea $f(n)$ **fca** , $f(n) > n \Rightarrow SPACE(s(n)) \subsetneq SPACE(s(n)log(s(n)))$

**Resultado 3.4.6** $P \subseteq TIME(2^n) \subsetneq TIME(2^{(n+1)} \subseteq EXP$

**Resultado 3.4.7** *Jerarquia de clases:* Dado $f : \mathbb{N} \to \mathbb{N}, f ca$ *se cumple:*

- $SPACE(f(n)) \subseteq NSPACE(f(n))$

- $TIME(f(n)) \subseteq NTIME(f(n))$
- $NTIME(f(n)) \subseteq SPACE(f(n))$
- $NSPACE(f(n)) \subseteq TIME(k^{(log(n)+f(n))})$

**Resultado 3.4.8** $TIME(f(n)) \subseteq NTIME(f(n)) \subseteq SPACE(f(n)) \subseteq NSPACE(f(n)) \subseteq EXP(k^{(log(n)+f(n))})$

**Resultado 3.4.9** $REACHABILITY \subseteq TIME(n^2)$ *con* $n = |V|$

**Resultado 3.4.10** $REACHABILITY \subseteq SPACE(log^2(n))$ *con* $n = |V|$

**Resultado 3.4.11** $NSPACE(s(n)) = SPACE(s^2(n))$ *con* $s > log(n), s : \mathbb{N} \to \mathbb{N}$ **fca**

**Resultado 3.4.12** *NPes cerrado bajo estrella de Kleene*

**Resultado 3.4.13** *Pes cerrado bajo estrella de Kleene*

# 4. Reducciones y Completes

## 4.1 Introducción

En este capitulo se busca establecer una *jerarquia* entre distintos problemas o lenguajes. La idea clave del capítulo es la nocion de poder *transformar* un lenguaje $L_1$ a otro lenguaje $L_2$ de tal suerte que si lograres resolver a $L_2$ esto resolverá a $L_1$ tambien. Y, por lo tanto, en este caso, $L_1$ **es a lo más tan difícil como** $L_2$

Con este objeto en el capitulo se postula la noción de este tipo de función que permite llevar a cabo la tarea antes mencionada. Dicha función es conocida como: **reducción muchos a uno**, sin embargo a diferencia del capitulo 2 de la parte anterior, en esta ocasión obligaremos que dichas funciones cumplan con la noción de eficiencia basada en $TIME(n^k)$.

## 4.2 Reducciones Polinomiales

**Definición 4.2.1** *Función Computable en tiempo polinomial* **(FP)**:

$$FP = \left\{ f : \Sigma^* \to \Sigma^* : \forall X \in \Sigma^*, \exists M \in MTD_k, M(X) \xrightarrow[n^k]{M} q_h, q_h \in \{q_y, q_n\} \right\}$$

.

**Definición 4.2.2** *Reducción en tiempo polinomial* $A \leq_P B$ **A se reduce a B**:

$$\leq_P = \{ f : \Sigma^* \to \Sigma^* : dados\ A, B \subseteq \Sigma^*, f \in \mathbf{FP}, \forall X \in A, X \in A \Rightarrow f(X) \in B \}$$

.

**Resultado 4.2.1** $\leq_P$ *es transitiva*

**Resultado 4.2.2** $SAT \leq_P 3SAT$

**Teorema 4.2.1** $\phi \in 3SAT \Leftrightarrow f(\phi) \in INDEPENDENT\ SET$

**Resultado 4.2.3** $INDEPENDENT\ SET \leq_P CLIQUE$

**Resultado 4.2.4** $INDEPENDENT\ SET \leq_P CUBIERTA$

# 5. Problemas Completos y Duros

## 5.1 Introducción

**Definición 5.1.1** *Problemas NP-completos y duros* $A \subseteq \Sigma^*$ Sea $A \subseteq \Sigma^*, D \in \mathscr{C}$:

- $A \in D - HARD$, $si \forall B \in D, B \leq_P A$
- $A \in D - COMPLETO$, $si A \in D - HARD, A \in D$

## 5.2 Teorema de Cook

**Teorema 5.2.1** *Teorema de Cook:* $SAT \in NP - completo$

# Bibliografía


## Books

- PAPADIMITRIOU, Christos H. *Computational complexity*. Reading, Mass.: Addison Wesley Longman, 1994. xv, 523 s. ISBN 0-201-53082-1.

## Links

- $https://github.com/alfcar9/Complejidad/blob/master/Complejidad_yComputabilidad.pdf$

## Otors

- Curso: Çomplejidad y Computabilidad", Otoño 2018, Instituto Tecnológico Autónomo de México, prof. Rodolfo Conde

# Índice alfabético


## A

Alfabetos . . . . . 8
Automatas finitos . . . . . 11

## C

Clases de Complejidad Básicas . . . . . 20
Conjuntos Complemento . . . . . 21
Cotas Asintoticas . . . . . 18

## F

Familias de Complejidad . . . . . 20
Funciones Computables Enmerablemente y Computables . . . . . 14

## G

Graficas . . . . . 9

## I

Introducción . . . . . 7, 11, 13, 17, 23, 25

## L

Lenguajes . . . . . 9
Lenguajes Computables Enumerablemente y Computables . . . . . 13

## M

Maquina de Turing . . . . . 11
Maquinas de Turing . . . . . 13

## P

Problema de la membresia con tiempo . . . . 20
Problema de Satisficibilidad Booleana . . . . . 18
Problemas . . . . . 10
Problemas de Grafos . . . . . 19
Problemas o Lenguajes Comunes en Complejidad . . . . . 18
Problemas Representantes . . . . . 14
Propiedad de los lenguajes CE . . . . . 14

## R

Reducciones computables de lenguajes . . . . 14

Reducciones Polinomiales . . . . . 23
Resultados y Teoremas . . . . . 21

## S

SNAPSACK . . . . . 19

## T

Teorema de Cook . . . . . 25
Teoria Asintotica . . . . . 18
Tiempo, Espacio y fca . . . . . 17