Pontificia Universidad Católica de Chile

Departamento de Ciencia de la Computación

Matemáticas Discretas

Resumen

Autor
Cristóbal Rojas

Basado en apuntes de
Fernando Suárez
Marco Bucchi
Gabriel Diéguez

24 de mayo de 2022

1. Inducción

Principio del Buen Orden (PBO)

$$A \neq \varnothing, A \subseteq \mathbb{N} \Rightarrow \exists x \in A \text{ tal que } \forall y \in A, x \leq y$$

Principio de Inducción Simple (PIS)

- Primera formulación: Si

  1. $0 \in A$ (base inductiva).
  2. Si $n \in A$ (hipótesis inductiva), entonces $n+1 \in A$ (tesis inductiva).

  entonces $A = \mathbb{N}$.

- Segunda formulación: Si

  1. $P(0)$ es verdadero
  2. Si $P(n)$, entonces $P(n+1)$ (cada vez que n cumple P, $n+1$ también la cumple)

  entonces todos los elementos de $\mathbb{N}$ cumplen la propiedad P.

- Tercera formulación: Si

  1. $P(\mathbf{n_0})$ es verdadero
  2. Si $P(n)$, entonces $P(n+1)$ (cada vez que n cumple P, $n+1$ también la cumple)

  entonces todos los elementos de $\mathbb{N}$ **a partir de $\mathbf{n_0}$** cumplen la propiedad P.

Principio de Inducción por Curso de Valores (PICV)

- Primera formulación: Si

$$\{0, 1, \ldots, n-1\} \subseteq A \Rightarrow n \in A$$

  entonces $A = \mathbb{N}$.

- Segunda formulación: Si

$$\forall k \in \mathbb{N}, k < n, P(k) \text{ es verdadero} \Rightarrow P(n) \text{ es verdadero}$$

  entonces P es verdadero para todos los elementos de $\mathbb{N}$.

Teorema 1

Los 3 principios de inducción (PBO, PIS y PICV) son equivalentes.

Conjunto definido inductivamente

1. El conjunto es el menor que cumple las reglas.
2. Definir conjunto de elementos base.
3. Definir conjunto finito de reglas de construcción de nuevos elementos en base a los elementos iniciales.

Inducción estructural

Si

1. Todos los elementos base de A (conjunto definido inductivamente) cumplen la propiedad P
2. Para cada regla de construcción, si la regla se aplica sobre elementos en A que cumplen la propiedad P, entonces los elementos producidos por la regla también cumplen la propiedad P

entonces todos los elementos en A cumplen la propiedad P.

2. Lógica proposicional

Sintaxis

Sea P un conjunto de variables proposicionales. El conjunto de todas las **fórmulas** de lógica proposicional sobre P, denotado por $L(P)$, es el menor conjunto que cumple las siguientes reglas:

1. Si $p \in P$, entonces $p \in L(P)$.
2. Si $\alpha \in L(P)$, entonces $(\neg\alpha) \in L(P)$.
3. Si $\alpha, \beta \in L(P)$, entonces $(\alpha \wedge \beta) \in L(P)$, $(\alpha \vee \beta) \in L(P)$, $(\alpha \rightarrow \beta) \in L(P)$ y $(\alpha \leftrightarrow \beta) \in L(P)$.

Semántica

Una **valuación** o **asignación de verdad** para las variables proposicionales en un conjunto P es una función $\sigma : P \rightarrow \{0, 1\}$, donde 0 = falso y 1 = verdadero.

Tablas de verdad

p	$\neg p$
0	1
1	0

p	q	$p \to q$
0	0	1
0	1	1
1	0	0
1	1	1

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

p	q	$p \leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

p	q	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Equivalencia lógica $\equiv$

Dos fórmulas $\alpha, \beta \in L(P)$ son **lógicamente equivalentes** (denotado como $\alpha \equiv \beta$) si para toda valuación σ se tiene que $\sigma(\alpha) = \sigma(\beta)$.

Leyes de equivalencia

1. Doble negación
   $\neg(\neg\alpha) \equiv \alpha$
2. De Morgan
   $\neg(\alpha \wedge \beta) \equiv (\neg\alpha) \vee (\neg\beta)$
   $\neg(\alpha \vee \beta) \equiv (\neg\alpha) \wedge (\neg\beta)$
3. Conmutatividad
   $\alpha \wedge \beta \equiv \beta \wedge \alpha$
   $\alpha \vee \beta \equiv \beta \vee \alpha$
4. Asociatividad
   $\alpha \wedge (\beta \wedge \gamma) \equiv (\alpha \wedge \beta) \wedge \gamma$
   $\alpha \vee (\beta \vee \gamma) \equiv (\alpha \vee \beta) \vee \gamma$
5. Distributividad
   $\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
   $\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
6. Idempotencia
   $\alpha \wedge \alpha \equiv \alpha$
   $\alpha \vee \alpha \equiv \alpha$
7. Absorción
   $\alpha \wedge (\alpha \vee \beta) \equiv \alpha$
   $\alpha \vee (\alpha \wedge \beta) \equiv \alpha$
8. Implicancia
   $\alpha \to \beta \equiv (\neg\alpha) \vee \beta$
9. Doble implicancia
   $\alpha \leftrightarrow \beta \equiv (\alpha \to \beta) \wedge (\beta \to \alpha)$

Operadores generalizados

$$\bigwedge_{i=1}^{n} \alpha_i = \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$$

$$\bigvee_{i=1}^{n} \alpha_i = \alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n$$

Teorema 2

Podemos representar cualquier tabla de verdad con una fórmula. Ademas, podemos representar cualquier tabla de verdad con una fórmula que sólo usa $\neg$, $\wedge$ y $\vee$.

Conectivos funcionalmente completos

Un conjunto de conectivos se dice así si toda fórmula en $L(P)$ es lógicamente equivalente a una fórmula que sólo usa esos conectivos. Ejemplos:

- $\{\neg, \wedge, \vee\}$
- $\{\neg, \vee\}$
- $\{\neg, \wedge\}$
- $\{\neg, \to\}$

Satisfacibilidad

Una fórmula α es **satisfacible** si existe una valuación σ tal que $\sigma(\alpha) = 1$.

Contradicción

Una fórmula α es una **contradicción** si no es satisfacible; es decir, para toda valuación σ se tiene que $\sigma(\alpha) = 0$.

Tautología

Una fórmula α es una **tautología** si para toda valuación σ se tiene que $\sigma(\alpha) = 1$.

Teorema 3

Dos fórmulas $\alpha, \beta \in L(P)$ son **lógicamente equivalentes** si $\alpha \leftrightarrow \beta$ es una **tautología**.

Forma Normal Disyuntiva (DNF)

Una fórmula α está en DNF si es una disyunción de conjunciones de literales (variable proposicional o su negación); osea, si es de la forma

$$B_1 \vee B_2 \vee \ldots \vee B_k$$

donde cada B_i es una conjunción de literales, $B_i = (l_{i1} \wedge \ldots \wedge l_{ik_i})$.

Forma Normal Conjuntiva (CNF)

Una fórmula α está en CNF si es una conjunción de disyunciones de literales; osea, si es de la forma

$$C_1 \wedge C_2 \wedge \ldots \wedge C_k$$

donde cada C_i es una disyunción de literales, $C_i = (l_{i1} \vee \ldots \vee l_{ik_i})$.

- Una disyunción de literales se llama **cláusula**.

Teorema 4

Toda fórmula es equivalente a una fórmula en DNF.

Teorema 5

Toda fórmula es equivalente a una fórmula en CNF.

Propiedad formas normales

Toda fórmula $\alpha \in L(P)$ en DNF con a lo más n disyunciones es lógicamente equivalente a una fórmula β en CNF ($\alpha \equiv \beta$).

Consecuencia lógica $\models$

Una fórmula α es consecuencia lógica de Σ si para cada valuación σ tal que $\sigma(\Sigma) = 1$, se tiene que $\sigma(\alpha) = 1$. Se denota como $\Sigma \models \alpha$.

Consecuencias lógicas clásicas

- Modus ponens

  $\{p, p \to q\} \models q$

- Modus tollens

  $\{\neg q, p \to q\} \models \neg p$

- Demostración por partes

  $\{p \vee q \vee r, p \to s, q \to s, r \to s\} \models s$

- Resolución

  $\{p \vee q, \neg q \vee r\} \models p \vee r$

Conjunto satisfacible

Un conjunto de fórmulas Σ es **satisfacible** si existe una valuación σ tal que $\sigma(\Sigma) = 1$. En caso contrario, Σ es **inconsistente**.

Teorema 6

$\Sigma \models \alpha$ si y sólo si $\Sigma \cup \{\neg\alpha\}$ es inconsistente.

Teorema 7

Un conjunto de fórmulas Σ es inconsistente si y sólo si $\Sigma \models \square$, con $\square$ una cláusula vacía.

Equivalencia lógica para conjuntos

Dos conjuntos de fórmulas Σ_1 y Σ_2 son **lógicamente equivalentes** ($\Sigma_1 \equiv \Sigma_2$) si para toda valuación σ se tiene que $\sigma(\Sigma_1) = \sigma(\Sigma_2)$. También diremos que Σ es lógicamente equivalente a una fórmula α si $\Sigma \equiv \alpha$.

Conjuntos y conjunción

Todo conjunto de fórmulas Σ es equivalente a la conjunción de sus fórmulas.

$$\Sigma \equiv \bigwedge_{\alpha \in \Sigma} \alpha$$

Teorema 8

Todo conjunto de fórmulas Σ es equivalente a un conjunto de cláusulas.

Resolución para consecuencia lógica

Para resolver el problema de consecuencia lógica (es decir, determinar si $\Sigma \models \alpha$) tenemos que determinar si un conjunto de cláusulas Σ' construido desde $\Sigma \cup \{\neg\alpha\}$ es tal que $\Sigma' \models \square$.

Resolución proposicional

Regla de resolución

$$\frac{C_1 \vee l \vee C_2 \qquad C_3 \vee \bar{l} \vee C_2}{C_1 \vee C_2 \vee C_3 \vee C_4}$$

Regla de factorización

$$\frac{C_1 \vee l \vee C_2 \vee l \vee C_3}{C_1 \vee l \vee C_2 \vee C_3}$$

Dado un conjunto Σ de cláusulas, una **demostración por resolución** de que Σ es inconsistente es una secuencia de cláusulas $C_1, \ldots, C_n$ tal que $C_n = \square$ y para cada $i = 1 \ldots n$ se tiene que

- $C_i \in \Sigma$ o
- C_i se obtiene de dos cláusulas anteriores en la secuencia usando la regla de resolución, o

- C_i se obtiene de una cláusula anterior en la secuencia usando la regla de factorización.

Si existe tal demostración, escribimos $\Sigma \vdash \square$.

Teorema 8

Dado un conjunto de cláusulas Σ, se tiene que:
Correctitud: Si $\Sigma \vdash \square$, entonces $\Sigma \models \square$ (Σ es inconsistente).
Completitud: Si $\Sigma \models \square$, entonces $\Sigma \vdash \square$.

- **Corolario 1:** Si Σ es un conjunto de cláusulas, entonces $\Sigma \models \square$ si y sólo si $\Sigma \vdash \square$. Dicho de otra manera, un conjunto de cláusulas Σ es inconsistente si y sólo si existe una demostración por resolución de que es inconsistente.
- **Corolario 2:** Dados un conjunto de fórmulas Σ y una fórmula α cualesquiera,

  $$\Sigma \models \alpha \text{ si y sólo si } \Sigma' \vdash \square$$

  donde Σ' es un conjunto de cláusulas tal que $\Sigma \cup \{\neg\alpha\} \equiv \Sigma'$.

3. Lógica de predicados

Predicado

Un predicado n-ario $P(x_1, \ldots, x_n)$ es una afirmación con n variables, cuyo valor de verdad depende de los objetos en el cual es evaluado.

- P es el **símbolo** del predicado.
- Todos los predicados están restringidos a un **dominio** de evaluación.
- Para un predicado $P(x_1, \ldots, x_n)$ diremos que $x_1, \ldots, x_n$ son **variables libres** de P.
- Un predicado **0-ario** es un predicado sin variables y tiene valor de verdadero o falso sin importar la valuación.

Cuantificador universal $\forall$

Sea $P(x, y_1, \ldots, y_n)$ un predicado compuesto con dominio D. Definimos el cuantificador universal:

$$P'(y_1, \ldots, y_n) = \forall x(P(x, y_1, \ldots, y_n))$$

donde x es la variable cuantificada y $y_1, \ldots, y_n$ son las variables libres. Para $b_1, \ldots, b_n$ en D, definimos la valuación:

$$P'(b_1, \ldots, b_n) = 1$$

si **para todo** a en D se tiene que $P(a, b_1, \ldots, b_n) = 1$, y 0 en otro caso.

Cuantificador existencial $\exists$

Sea $P(x, y_1, \ldots, y_n)$ un predicado compuesto con dominio D. Definimos el cuantificador existencial:

$$P'(y_1, \ldots, y_n) = \exists x(P(x, y_1, \ldots, y_n))$$

donde x es la variable cuantificada y $y_1, \ldots, y_n$ son las variables libres. Para $b_1, \ldots, b_n$ en D, definimos la valuación:

$$P'(b_1, \ldots, b_n) = 1$$

si **existe** a en D tal que $P(a, b_1, \ldots, b_n) = 1$, y 0 en otro caso.

Interpretaciones $\mathcal{I}$

Una interpretación $\mathcal{I}$ para $P_1, \ldots, P_n$ está compuesta de:

- un dominio D que denotaremos $\mathcal{I}(dom)$ y
- un predicado P_i^D que denotaremos por $\mathcal{I}(P_i)$ para cada símbolo P_i.

Satisfacibilidad para interpretaciones

$$\mathcal{I} \models \alpha(a_1, \ldots, a_n)$$

si $\alpha(a_1, \ldots, a_n)$ es verdadero al interpretar cada símbolo en α según $\mathcal{I}$. Si $\mathcal{I}$ **no satisface** α sobre $a_1, \ldots, a_n$ en $\mathcal{I}(dom)$ lo denotamos como $\mathcal{I} \not\models \alpha(a_1, \ldots, a_n)$.

Equivalencia lógica para predicados

Sean $\alpha(x_1, \ldots, x_n)$ y $\beta(x_1, \ldots, x_n)$ dos fórmulas en lógica de predicados. Decimos que α y β son **lógicamente equivalentes**:

$$\alpha \equiv \beta$$

si para toda interpretación $\mathcal{I}$ y para todo $a_1, \ldots, a_n$ en $\mathcal{I}(dom)$ se cumple:

$$\mathcal{I} \models \alpha(a_1, \ldots, a_n) \text{ si y sólo si } \mathcal{I} \models \beta(a_1, \ldots, a_n)$$

- **Caso especial:** Si α y β son oraciones (no tienen variables libres), entonces:

  $$\mathcal{I} \models \alpha \text{ si y sólo si } \mathcal{I} \models \beta$$

Teorema 9

Sea $\alpha(x)$, $\beta(x)$ fórmulas con x su variable libre, entonces:

$$\neg\forall x(\alpha(x)) \equiv \exists x(\neg\alpha(x))$$
$$\neg\exists x(\alpha(x)) \equiv \forall x(\neg\alpha(x))$$
$$\forall x(\alpha(x) \wedge \beta(x)) \equiv \forall x(\alpha(x)) \wedge \forall x(\beta(x))$$
$$\exists x(\alpha(x) \vee \beta(x)) \equiv \exists x(\alpha(x)) \vee \exists x(\beta(x))$$

Consecuencia lógica para predicados

Una fórmula α es consecuencia lógica de un conjunto de fórmulas Σ:

$$\Sigma \models \alpha$$

si para toda interpretación $\mathcal{I}$ y $a_1, \ldots, a_n$ en $\mathcal{I}(dom)$ se cumple que:

$$\text{si } \mathcal{I} \models \Sigma(a_1, \ldots, a_n) \text{ entonces } \mathcal{I} \models \alpha(a_1, \ldots, a_n)$$

Reglas de inferencia para predicados

- Especificación universal

$$\frac{\forall x(\alpha(x))}{\alpha(a) \text{ para cualquier } a}$$

- Generalización universal

$$\frac{\alpha(a) \text{ para un } a \text{ arbitrario}}{\forall x(\alpha(x))}$$

- Especificación existencial

$$\frac{\exists x(\alpha(x))}{\alpha(a) \text{ para algún } a \text{ (nuevo)}}$$

- Generalización existencial

$$\frac{\alpha(a) \text{ para algún } a}{\exists x(\alpha(x))}$$

4. Demostraciones

Método directo

Por demostrar: $\forall x(P(x) \to Q(x))$.
Suponemos que $P(n)$ es verdadero para un n arbitrario y demostramos que $Q(n)$ también es verdadero.

Contrapositivo

Por demostrar: $\forall x(P(x) \to Q(x))$.
Suponemos que $Q(n)$ es falso para un n arbitrario y demostramos que $P(n)$ también es falso.

Contradicción

Por demostrar: $\forall x(P(x) \to Q(x))$.
Suponemos que existe un n tal que $P(n)$ es verdadero y $Q(n)$ es falso e inferimos una contradicción.

Por casos

Para cada subdominio $C_1, \ldots, C_n$ demostramos que:

$$\forall x(P(x) \to Q(x))$$

- Aquí, dividimos el dominio de la interpretación $\mathcal{I}$ con que trabajamos en una cantidad finita de casos $C_1, \ldots, C_n$, tal que:

$$\mathcal{I}(dom) = \bigcup_{i=1}^{n} C_i$$

Doble implicación

Por demostrar: $\forall x(P(x) \leftrightarrow Q(x))$.
Aquí, se deben demostrar ambas direcciones por separado. En términos formales:

$$\forall x(P(x) \to Q(x)) \land \forall x(Q(x) \to P(x))$$

Contra-ejemplo

Por demostrar: $\neg(\forall x(P(x)))$.
Aquí, hay que encontrar un elemento n (cualquiera) tal que $P(n)$ es falso.

Existencial

Por demostrar: $\exists x(P(x))$.
Aquí, debemos demostrar que existe un elemento n tal que $P(n)$ es verdadero (nótese que NO es estrictamente necesario mostrar n explícitamente).

5. Teoría de conjuntos

Conjunto

Es una colección *bien definida* de objetos. Estos objetos se llaman **elementos** del conjunto, y diremos que **pertenecen** a él.

Subconjunto

Sean A y B conjuntos. Diremos que A es **subconjunto** de B, denotado por $A \subseteq B$, si

$$\forall x(x \in A \to x \in B)$$

En otras palabras, A es subconjunto de B si cada elemento de A está tambien en B.

Igualdad de conjuntos

Dos conjuntos A y B son iguales si y sólo si $A \subseteq B$ y $B \subseteq A$. Otra definición equivalente es

$$\forall A \forall B \quad A = B \leftrightarrow \forall x(x \in A \leftrightarrow x \in B)$$

- Este es el **axioma de extensión**.
- **Observación:** Los conjuntos no pueden tener elementos repetidos. $\{x, x\} = \{x\}$.

Subconjunto propio

Diremos que A es un subconjunto propio de B, detonado por $A \subsetneq B$, si

$A \subseteq B \wedge A \neq B$, o alternativamente, $A \subseteq B \wedge B \nsubseteq A$

- **Corolario:** $B \nsubseteq A$ si y sólo si $\exists x \in B$ tal que $x \notin A$.

Axioma del conjunto vacío

$\exists X$ tal que $\forall x, x \notin X$. Lo denotaremos por $\varnothing$ o $\{\}$.

Teorema 10

Para todo conjunto A se tiene que $\varnothing \subseteq A$.

Teorema 11

Existe un único conjunto vacío.

Formas de definir un conjunto

- Por extensión (listando sus elementos):

$$\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$$

- Por comprensión:

$$\mathbb{Z}_5 = \{x \mid x \in \mathbb{N} \wedge x < 5\}$$

Axioma de abstracción

Si α es una propiedad sobre objetos, entonces $A = \{x \mid \alpha(x)\}$ es un conjunto.

Axioma de separación

Si α es una propiedad y C es un conjunto "sano", entonces $A = \{x \mid x \in C \wedge \alpha(x)\}$ es un conjunto.

- Un **conjunto sano** es uno el cual no fue creado usando el axioma de abstracción.

Operaciones

- Unión: $A \cup B = \{x \mid x \in A \vee x \in B\}$
- Intersección: $A \cap B = \{x \mid x \in A \wedge x \in B\}$
- Diferencia: $A \backslash B = \{x \mid x \in A \wedge x \notin B\}$
- Conjunto potencia: $\mathcal{P}(A) = \{X \mid X \subseteq A\}$

  Algunas observaciones:

  - $\varnothing \in \mathcal{P}(A)$
  - $A \in \mathcal{P}(\mathcal{A})$
  - $A \subseteq A \cup B$
  - $A \cap B \subseteq A$

Complemento

Sea $A \subseteq \mathcal{U}$ (siendo $\mathcal{U}$ un conjunto universal fijo) un conjunto cualquiera. El complemento de A (relativo a $\mathcal{U}$) es el conjunto

$$A^c = \mathcal{U} \backslash A = \{x \mid x \in \mathcal{U} \wedge x \notin A\}$$

Teorema 12

Si A, B y C son conjuntos cualquiera (subconjuntos de $\mathcal{U}$), entonces se cumplen las siguientes leyes:

- Asociatividad
  $A \cup (B \cup C) = (A \cup B) \cup C$
  $A \cap (B \cap C) = (A \cap B) \cap C$
- Conmutatividad
  $A \cup B = B \cup A$
  $A \cap B = B \cap A$
- Idempotencia
  $A \cup A = A$
  $A \cap A = A$
- Absorción
  $A \cup (A \cap B) = A$
  $A \cap (A \cup B) = A$
- Elemento neutro
  $A \cup \varnothing = A$
  $A \cap \mathcal{U} = A$
- Distributividad
  $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
  $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- Leyes de De Morgan
  $(A \cup B)^c = A^c \cap B^c$
  $(A \cap B)^c = A^c \cup B^c$

- Elemento inverso
  $A \cup A^c = \mathcal{U}$
  $A \cap A^c = \varnothing$
- Dominación
  $A \cup \mathcal{U} = \mathcal{U}$
  $A \cap \varnothing = \varnothing$

Unión generalizada

Siendo $\mathcal{S}$ un conjunto de conjuntos, la unión generalizada representa a la unión de todos los conjuntos componentes de $\mathcal{S}$; es decir, contiene a todos los elementos que pertenecen a algún conjunto de $\mathcal{S}$.

$$\bigcup \mathcal{S} = \{x \mid \exists A \in \mathcal{S} \text{ tal que } x \in A\} = \bigcup_{A \in \mathcal{S}} A$$

Intersección generalizada

Siendo $\mathcal{S}$ un conjunto de conjuntos, la intersección generalizada representa a la intersección de todos los conjuntos componentes de $\mathcal{S}$; es decir, contiene a todos los elementos que pertenecen a todos los conjuntos de $\mathcal{S}$.

$$\bigcap \mathcal{S} = \{x \mid \forall A \in \mathcal{S} \text{ se cumple que } x \in A\} = \bigcap_{A \in \mathcal{S}} A$$

6. Relaciones

Par ordenado

Sean $a, b \in \mathcal{U}$ (donde $\mathcal{U}$ es un conjunto universal). Definimos el par ordenado (a, b) como

$$(a, b) = \{\{a\}, \{a, b\}\}$$

- Propiedad: $(a, b) = (c, d)$ si y sólo si $a = c \wedge b = d$

n-tupla

Sean $a_1, \ldots, a_n \in \mathcal{U}$. Definimos una n-tupla como:

$$(a_1, \ldots, a_n) = ((a_1, \ldots, a_{n-1}), a_n)$$

Producto cartesiano

Dados dos conjuntos A y B, definimos el producto cartesiano entre A y B como

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

Se puede extender esta noción a más conjuntos. Dados conjuntos $A_1, \ldots, A_n$, definimos el producto cartesiano entre los A_i como

$$A_1 \times \ldots \times A_n = \{(a_1, \ldots, a_n) \mid a_1 \in A_1 \wedge \ldots \wedge a_n \in A_n\}$$

Relación

Dados conjuntos $A_1, \ldots, A_n$, diremos que R es una relación sobre tales conjuntos si $R \subseteq A_1 \times \ldots \times A_n$.

- La **aridad** de una relación R es el tamaño de las tuplas que la componen. Así, diremos que R es una relación n-aria.

Relación binaria

Dados conjuntos A y B, diremos que R es una relación binaria de A en B si $R \subseteq A \times B$.

- Podemos tener una relación sobre un solo conjunto, es decir, dado un conjunto A, diremos que R es una relación binaria sobre A si $R \subseteq A \times A = A^2$.
- Cuando tengamos productos cartesianos entre un mismo conjunto, usaremos una notación de *potencia*:

$$A \times \overset{(n-2veces)}{\cdots} \times = A^n$$

Relación binaria: Divide a $\mid$

Esta relación sobre los naturales sin el 0, es una tal que a está relacionado con b si y sólo si b es múltiplo de a:

$$a \mid b \text{ si y sólo si } \exists k \in \mathbb{N} \text{ tal que } b = ka$$

Relación binaria: Equivalencia módulo n $\equiv_n$

Esta relación sobre los naturales, es una tal que a está relacionado con b si y sólo si $|a - b|$ es múltiplo de n:

$$a \equiv_n b \text{ si y sólo si } \exists k \in \mathbb{N} \text{ tal que } |a - b| = kn$$

Propiedades de las relaciones

Una relación es

- **Refleja** si para cada $a \in A$ se tiene que $R(a, a)$.
- **Irrefleja** si para cada $a \in A$ *no* se tiene que $R(a, a)$.
- **Simétrica** si para cada $a, b \in A$, si $R(a, b)$ entonces $R(b, a)$.
- **Asimétrica** si para cada $a, b \in A$, si $R(a, b)$ entonces *no es cierto* que $R(b, a)$.
- **Antisimétrica** si para cada $a, b \in A$, si $R(a, b)$ y $R(b, a)$, entonces $a = b$.
- **Transitiva** si para cada $a, b, c \in A$, si $R(a, b)$ y $R(b, c)$, entonces $R(a, c)$.
- **Conexa** si para cada $a, b \in A$, se tiene que $R(a, b)$ o $R(b, a)$.

  En lógica de predicados (para demostrar):

- Refleja: $\forall x(R(x,x))$
- Irrefleja: $\forall x(\neg R(x,x))$
- Simétrica: $\forall x \forall y(R(x,y) \rightarrow R(y,x))$
- Asimétrica: $\forall x \forall y(R(x,y) \rightarrow \neg R(y,x))$
- Antisimétrica: $\forall x \forall y((R(x,y) \wedge R(y,x)) \rightarrow x = y)$
- Transitiva: $\forall x \forall y \forall z((R(x,y) \wedge R(y,z)) \rightarrow R(x,z))$
- Conexa: $\forall x \forall y(R(a,b) \vee R(b,a))$

Relaciones de equivalencia

Una relación R sobre A es una relación de equivalencia si es refleja, simétrica y transitiva.

Clase de equivalencia

Sea $\sim$ una relación de equivalencia sobre un conjunto A y un elemento $x \in A$. La clase de equivalencia de x bajo $\sim$ es el conjunto

$$[x]_{\sim} = \{y \in A \,|\, x \sim y\}$$

Teorema 13

Sea $\sim$ una relación de equivalencia sobre un conjunto A. Entonces:

1. $\forall x \in A, x \in [x]$
2. $x \sim y$ si y sólo si $[x] = [y]$
3. Si $[x] \neq [y]$ entonces $[x] \cap [y] = \varnothing$

Conjunto cuociente

Sea $\sim$ una relación de equivalencia sobre un conjunto A. El conjunto cuociente de A con respecto a $\sim$ es el conjunto de todas las clases de equivalencia de $\sim$:

$$A/\sim = \{[x] \,|\, x \in A\}$$

Índice

Es la cantidad de clases de equivalencia que induce. Es decir, la cantidad de elementos de su conjunto cuociente.

Partición

Sea A un conjunto cualquiera, y S una colección de subconjuntos de A ($S \subseteq \mathcal{P}(A)$). Diremos que S es una partición de A si cumple que:

1. $\forall X \in S, X \neq \varnothing$
2. $\bigcup S = A$
3. $\forall X, Y \in S$ si $X \neq Y$ entonces $X \cap Y = \varnothing$

Teorema 14

Si $\sim$ es una relación de equivalencia sobre un conjunto A, entonces $A/\sim$ es una partición de A.

Relación $\downarrow$

Esta relación sobre $\mathbb{N} \times \mathbb{N}$ se define como:

$$(m,n) \downarrow (r,s) \quad \leftrightarrow \quad m + s = n + r$$

Números enteros

El conjunto de los números $\mathbb{Z}$ se define como el conjunto cuociente de $\mathbb{N}^2$ respecto a $\downarrow$:

$$\mathbb{Z} = \mathbb{N}/\downarrow = \{[(0,0)], [(0,1)], [(1,0)], [(0,2)], [(2,0)], \ldots\}$$

- $[(0,0)]$ será el entero 0.
- $[(0,i)]$ sera el entero i.
- $[(i,0)]$ será el entero $-i$.

Nota: "-1" es sólo un **nombre** para la clase de equivalencia $[(1,0)]$. El símbolo "$-$" no significa nada por sí solo.

Suma sobre $\mathbb{Z}$

$$[(m,n)] +_{\downarrow} [(r,s)] = [(m + r, n + s)]$$

Producto sobre $\mathbb{Z}$

$$[(m,n)] \cdot_{\downarrow} [(r,s)] = [(m \cdot s + n \cdot r, m \cdot r + n \cdot s)]$$

7. Funciones

Función

Sea f una relación binaria de A en B; es decir, $f \subseteq A \times B$. Diremos que f es una función de A en B si dado cualquier elemento $a \in A$, si existe un elemento $b \in B$ tal que afb, este es único:

$$afb \wedge afc \rightarrow b = c$$

Si afb, escribimos $b = f(a)$.

- b es la **imagen** de a.
- a es la **preimagen** de b.
- **Notación:** $f : A \rightarrow B$

Función total

Una función $f : A \to B$ se dice total si todo elemento en A tiene imagen, es decir

- Para todo $a \in A$ existe $b \in B$ tal que $b = f(a)$.
- Una función que no sea total se dice **parcial**.
- Toda función será total a menos que se diga lo contrario.

Inyectividad

Una función $f : A \to B$ es inyectiva (o $1-1$) si para cada par de elementos $x, y \in A$ se tiene que $f(x) = f(y) \to x = y$. Es decir, no existen dos elementos distintos en A con la misma imagen.

Sobreyectividad

Una función $f : A \to B$ es sobreyectiva (o sobre) si cada elemento $b \in B$ tiene preimagen. Es decir, para todo $b \in B$ existe un $a \in A$ tal que $b = f(a)$.

Biyectividad

Una función $f : A \to B$ es biyectiva si es inyectiva y sobreyectiva **a la vez**.

Relación inversa

Dada una relación R de A en B, una relación inversa de R es una relación de B en A definida como

$$R^{-1} = \{(b, a) \in B \times A \mid aRb\}$$

Función invertible

Dada una función f de A en B, diremos que f es invertible si su relación inversa f^{-1} es una relación de B en A.

Composición de relaciones

Dadas relaciones R de A en B y S de B en C, la composición de R y S es una relación de A en C definida como

$$S \circ R = \{(a, c) \in A \times C \mid \exists b \in B \text{ tal que } aRb \wedge bSc\}$$

Composición de funciones

Dadas funciones f de A en B y g de B en C, la composición $g \circ f$ es una función de A en C.

Teorema 14

Si $f : A \to B$ es biyectiva, entonces la relación inversa f^{-1} es una función biyectiva de B en A.

- Si f es biyectiva, entonces es invertible.

Teorema 15

Dados dos funciones $f : A \to B$ y $g : B \to C$:

- Si f y g son inyectivas, entonces $g \circ f$ también lo es.
- Si f y g son sobreyectivas, entonces $g \circ f$ también lo es.

Principio del palomar

Se tienen m palomas y n palomares, con $m > n$. Entonces, si se reparten las m palomas en los n palomares, necesariamente existirá un palomar con más de una paloma.

Principio del palomar (matemático)

Si se tiene una función $f : \mathbb{N}_m \to \mathbb{N}_n$, con $m > n$, la función f no puede ser inyectiva. Es decir, necesariamente existirán $x, y \in \mathbb{N}_m$ tales que $x \neq y$, pero $f(x) = f(y)$.

Principio del palomar (sobreyectividad)

Si se tiene una función $f : \mathbb{N}_m \to \mathbb{N}_n$, con $m < n$, la función f no puede ser sobreyectiva.

Conjunto equinumeroso

Sean A y B dos conjuntos cualesquiera. Diremos que A es equinumeroso con B (o que A tiene el mismo tamaño que B) si existe una función biyectiva $f : A \to B$. Lo denotamos como

$$A \approx B$$

Esta es una **relación de equivalencia**.

Cardinalidad

La cardinalidad de un conjunto A es su clase de equivalencia bajo $\approx$:

$$|A| = [A]_{\approx}$$

Conjunto finito

Diremos que A es un conjunto finito si $A \approx n$, para algún $n \in \mathbb{N}$. Es decir, si existe una función biyectiva $f : A \to n = \{0, \ldots, n-1\}$.

- En tal caso, se tiene que $|A| = [n]_{\approx}$.
- Por simplicidad, diremos que $|A| = n$.
- También podemos decir que A tiene n elementos.

Lema 1

Sean A y B dos conjuntos finitos tales que $A \cap B = \varnothing$. Entonces, $|A \cup B| = |A| + |B|$.

Teorema 16

Sea A un conjunto finito. Entonces, se cumple que $|P(A)| = 2^{|A|}$.

- Esto implica que si A es un conjunto finito, entonces su cardinalidad es **estrictamente menor** que la de su conjunto potencia.

Conjunto enumerable

Un conjunto A se dice enumerable si $|A| = |\mathbb{N}|$.

Teorema 17 (Schröder-Bernstein)

$A \approx B$ si y sólo si existen funciones inyectivas $f : A \to B$ y $g : B \to A$.

Conjunto enumerable (alternativa)

Un conjunto A es enumerable si y sólo si todos sus elementos se pueden poner en una lista infinita; es decir, si existe una sucesión infinita

$$(a_0, a_1, a_2, \ldots, a_n, a_{n+1}, \ldots)$$

tal que *todos* los elementos de A aparecen en la sucesión *una única vez* cada uno.

Teorema 18 (Cantor)

El intervalo real $(0, 1) \subseteq \mathbb{R}$ es infinito pero no enumerable.

Teorema 19

$$(0, 1) \approx \mathbb{R} \approx \mathcal{P}(\mathbb{N})$$

Relación $\preceq$ entre conjuntos

Dados conjuntos A y B, diremos que $A \preceq B$ (A no es más grande que B), si existe una función inyectiva $f : A \to B$.

- $\preceq$ **no** es una relación de orden.
- Si $A \preceq B$, diremos que $|A| \leq |B|$.

Conjunto menos numeroso

Dados conjuntos A y B, diremos que $A \prec B$ (A es menos numeroso que B) si $A \preceq B$ pero $A \not\approx$ B.

- Esta noción con funciones se define de tal forma que existe una función inyectiva $f : A \to B$, pero no existe una función biyectiva $g : A \to B$.
- Si $A \prec B$, diremos que $|A| < |B|$.
- Corolario: $|\mathbb{N}| < |\mathcal{P}(\mathbb{N})|$.

Teorema 20 (Cantor, menos numeroso)

A es menos numeroso que su conjunto potencia, es decir

$$|A| < |\mathcal{P}(A)|$$

8. Análisis de algoritmos

Algoritmo

Es un método para convertir un **input** en un **output**. A estos métodos les exigiremos ciertas propiedades:

- **Precisión:** cada instrucción debe ser planteada de forma precisa y no ambigua.
- **Determinismo:** cada instrucción tiene un único comportamiento que depende sólo del input.
- **Finitud:** el algoritmo está compuesto por un conjunto finito de instrucciones.

Objetivos

El análisis de algoritmos tiene dos objetivos:

- Estudiar cuándo y por qué los algoritmos son **correctos** (es decir, hacen lo que dicen que hacen).
- Estimar la cantidad de **recursos** computacionales que un algoritmo necesita para su ejecución.

Pseudo-código

Se usa pseudo-código para escribir algoritmos:

- Instrucciones usuales como **if**, **while**, **return**, ...
- Notaciones cómodas para arreglos, conjuntos, etc.

Consideraremos que los algoritmos tienen:

- **Precondiciones:** representan el input del programa.
- **Postcondiciones:** representan el output del programa, lo que hace el algoritmo con el input.

Corrección de algoritmos

Un algoritmo es

- **correcto**, si para todo input válido, el algoritmo se detiene y produce un output correcto.
- **incorrecto**, si existe un input válido para el cual el algoritmo no se detiene o produce un output incorrecto.

Algoritmos iterativos

Debemos demostrar dos cosas:

- **Corrección parcial:** si el algoritmo se detiene, se cumplen las postcondiciones.
- **Terminación:** el algoritmo se detiene.

Demostración

Para demostrar la corrección parcial, buscamos un **invariante** $I(k)$ para los loops:

- Una propiedad I que sea verdadera en cada paso k de la iteración.
- Debe relacionar a las variables presentes en el algoritmo.
- Al finalizar la ejecución, debe asegurar que las postcondiciones se cumplan.

Una vez encontramos una invariante, demostramos la corrección del loop inductivamente:

- **Base:** las precondiciones deben implicar que $I(0)$ es verdadero.
- **Inducción:** para todo natural $k > 0$, si G e $I(k)$ son verdaderos antes de la iteración, entonces $I(k+1)$ es verdadero después de la iteración.
- **Corrección:** inmediatamente después de terminado el loop (i.e. cuando G es falso), si $k = N$ e $I(N)$ es verdadero, entonces las postcondiciones se cumplen.

Y para **demostrar terminación**, debemos mostrar que existe un k para el cual G es falso.

Algoritmos recursivos

En el caso de los algoritmos recursivos, no necesitamos dividir la demostración en corrección parcial y terminación.

- Basta con demostrar por inducción la propiedad (corrección) deseada.
- En general, la inducción se realiza sobre el tamaño del input.

Notación asintótica

Lo que nos interesa luego de determinar cuando un algoritmo es correcto, es su comportamiento a medida que crece el **input**, ya que el hecho de que un algoritmo sea correcto no implica que sea útil en la práctica: hay que determinar su tiempo de ejecución.
Vamos a ocupar funciones de dominio natural ($\mathbb{N}$) y recorrido real positivo ($\mathbb{R}^+$)

- El dominio será el tamaño del input de un algoritmo.
- El recorrido será el tiempo necesario para ejecutar el algoritmo.

Sea $f : \mathbb{N} \to \mathbb{R}^+$, entonces

$$O(f) = \{g : \mathbb{N} \to \mathbb{R}^+ \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0) : g(n) \leq c \cdot f(n)\}$$

Diremos que $g \in O(f)$ es a lo más de orden f o que es $O(f)$.

$$\Omega(f) = \{g : \mathbb{N} \to \mathbb{R}^+ \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0) : g(n) \geq c \cdot f(n)\}$$

Diremos que $g \in \Omega(f)$ es al menos de orden f o que es $\Omega(f)$.

$$\Theta(f) = O(f) \cap \Omega(f)$$

Diremos que $g \in \Theta(f)$ es exactamente de orden f o que es $\Theta(f)$.

Teorema 21

Si $f(n) = a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + \ldots a_2 \cdot n^2 + a_1 \cdot n + a_0$ con $a_i \in \mathbb{R}$ y $a_k > 0$, entonces f es $\Theta(n^k)$.

Teorema 22

Si $f(n) = \log_a(n)$ con $a > 1$, entonces para todo $b > 1$ se cumple que f es $\Theta(\log_b(n))$.

Funciones más usadas

Las funciones más usadas para los órdenes de notación asintótica tienen nombres típicos.

Notación	Nombre
$\Theta(1)$	Constante
$\Theta(\log(n))$	Logarítmico
$\Theta(n)$	Lineal
$\Theta(n \log(n))$	$n \log(n)$
$\Theta(n^2)$	Cuadrático
$\Theta(n^3)$	Cúbico
$\Theta(n^k)$	Polinomial
$\Theta(m^k)$	Exponencial
$\Theta(n!)$	Factorial

con $k \geq 0$ y $m \geq 2$.

Complejidad algoritmos iterativos

Queremos encontrar una función $T(n)$ que modele el tiempo de ejecución de un algoritmo.

- Donde n es el tamaño del input.
- No queremos valores exactos de T para cada n, sino que una notación asintótica para ella.
- Para encontrar T, contamos las instrucciones ejecutadas por el algoritmo.
- A veces contaremos cierto tipo de instrucciones que son relevantes para un algoritmo particular.

Consideremos el siguiente algoritmo de búsqueda de arreglos:

$$BÚSQUEDA(A, n, k)$$

Donde:

- **Input:** un arreglo de $A = [a_0, \ldots, a_{n-1}]$, un natural $n > 0$ correspondiente al largo del arreglo A y un entero k.
- **Output:** el índice de k en A, -1 si no está.

```
1       for i=0 to n-1 do
2           if a_i = k then
3               return i
4       return -1
```

¿Qué instrucción(es) contamos?

- Deben ser representativas de lo que hace el problema.
- En este caso, por ejemplo 3 y 4 no lo son.
- La instrucción 2 si lo sería, y más específicamente la comparación.
  - Las comparaciones estan entre las instrucciones que se cuentan típicamente, sobre todo en búsqueda y ordenación.

¿Respecto a qué parámetro buscamos la notación asintótica?

- En el ejemplo, es natural pensar en el tamaño de arreglo n.

En conclusión: queremos encontrar una notación asintótica (ojalá Θ) para la cantidad de veces que se ejecuta la comparación de la línea 2 en función de n. Llamaremos a esta cantidad $T(n)$.
Ahora, ¿$T(n)$ depende solo de n?

- El contenido del arreglo influye en la ejecución del algoritmo.
- Estimaremos entonces el tiempo para el **peor caso** (cuando el input hace que el algoritmo se demore la mayor cantidad de tiempo posible) y el **mejor caso** (lo contrario) para un tamaño de input n.

En nuestro ejemplo:

- **Mejor caso:** $a_0 = k$. Aquí la línea 2 se ejecuta una vez, y luego $T(n)$ es $\Theta(1)$.
- **Peor caso:** k no esta en A. La línea 2 se ejecutará tantas veces como elementos en A, y entonces $T(n)$ es $\Theta(n)$.
- Diremos entonces que el algoritmo $BÚSQUEDA(A, n, k)$ es de **complejidad** $\Theta(n)$ o lineal en el peor caso y $\Theta(1)$ o constante en el mejor caso.

En general, nos conformaremos con encontrar la complejidad del peor caso.

- Es la que más interesa, al decirnos qué tan mal se puede comportar un algoritmo en la práctica.

Además, a veces puede ser difícil encontrar una notación Θ.

- Nos basta y es suficiente con una buena estimación O, tanto para el mejor y el peor caso.
- Nos da una cota superior para el tiempo de ejecución del algoritmo.

Complejidad algoritmos recursivos

En este caso, el principio es el mismo: contar instrucciones.

- Buscamos alguna(s) instrucción(es) representativa(s).
- Contamos cuántas veces se ejecuta en cada ejecución del algoritmo.

Tenemos que considerar las llamadas recursivas del algoritmo.

- Esto hara que aparezcan fórmulas recursivas que deberemos resolver.

Por ejemplo, podemos encontrar una función $T(n)$ para la cantidad de comparaciones que realiza un algoritmo en el peor caso, en función del tamaño del arreglo. Este tipo de funciones se denomina **ecuación de recurrencia**.

Notación asintótica condicional

Sea $P \subseteq \mathbb{N}$. Luego,

$$O(f \mid P) = \{g : \mathbb{N} \to \mathbb{R}^+ \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N}) \\ (\forall n \geq n_0)(n \in P \to g(n) \leq c \cdot f(n))\}$$

Las notaciones $\Omega(f \mid P)$ y $\Theta(f \mid P)$ se definen de manera análoga.

Función asintóticamente no decreciente

Una función $f : \mathbb{N} \to \mathbb{R}^+$ se denomina así si cumple que

$$(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(f(n) \leq f(n+1))$$

Por ejemplo, las funciones $\log_2(n)$, n, n^k y 2^n son asíntoticamente no decrecientes.

Función b-armónica

Dado un natural $b > 0$, una función $f : \mathbb{N} \to \mathbb{R}^+$ se denomina b-armónica si $f(b \cdot n) \in O(f)$.

Teorema 23

Sean $f, g : \mathbb{N} \to \mathbb{R}^+$, un natural $b > 1$ y

$$POTENCIA_b = \{b^i \mid i \in \mathbb{N}\}$$

Si f, g son asintóticamente no decrecientes, g es b-armónica y $f \in O(g \mid POTENCIA_b)$, entonces $f \in O(g)$.

Teorema 24 (Maestro)

Si $a_1, a_2, b, c, c_0, d \in \mathbb{R}^+$ y $b > 1$, entonces para una recurrencia de la forma

$$T(n) = \begin{cases} c_0 & 0 \leq n < n_0 \\ a_1 \cdot T(\lceil \frac{n}{b} \rceil) + a_2 \cdot T(\lfloor \frac{n}{b} \rfloor) + c \cdot n^d & n \geq n_0 \end{cases}$$

se cumple que

$$T(n) \in \begin{cases} \Theta(n^d) & a_1 + a_2 < b^d \\ \Theta(n^d \cdot \log(n)) & a_1 + a_2 = b^d \\ \Theta(n^{\log_b(a_1+a_2)}) & a_1 + a_2 > b^d \end{cases}$$

9. Teoría de grafos

Grafo

Un grafo $G = (V, E)$ es un par donde V es un conjunto, cuyos elementos llamaremos *vértices o nodos*, y E es una relación binaria sobre V (es decir, $E \subseteq V \times V$), cuyos elementos llamaremos *aristas*.
Para representar un grafo, usamos puntos o círculos para dibujar vértices, y flechas para dibujar aristas. Cada arista será una flecha entre los nodos que relaciona. Estos son los llamados **grafos dirigidos**.

Rulo (loop)

Es una arista $(x, y) \in E$ tal que $x = y$. Es decir, es una arista que conecta un vértice con sí mismo.

Aristas paralelas

Dos aristas $(x, y) \in E$ y $(z, w) \in E$ son pararelas si $x = w$ e $y = z$. Es decir, si conectan a los mismos vértices.

Grafo no dirigido

Un grafo $G = (V, E)$ es no dirigido si toda arista tiene una arista paralela. Alternativamente, G es no dirigido si E es simétrica. En estos grafos se dibuja con trazos en lugar de flechas.

Grafo simple

Un grafo **no dirigido** $G = (V, E)$ es simple si no tiene rulos. Alternativamente, G es simple si E es irrefleja.

Convención sobre grafos

De ahora en adelante (a menos que se explicite otra cosa), cuando hablemos de grafos estaremos refiriéndonos a grafos **simples**, **no dirigidos**, **no vacíos** y **finitos**.

- $V \neq \varnothing$ y $|V| = n$, con $n \in \mathbb{N}$.
- E es simétrica e irrefleja.

Vértices adyacentes o vecinos

Dado un grafo $G = (V, E)$, dós vértices $x, y \in V$ son adyacengtes o vecinos si $(x, y) \in E$.

Isomorfismo

Dos grafos $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ son **isomorfos** si existe una función biyectiva $f : V_1 \to V_2$ tal que $(x, y) \in E_1$ si y sólo si $(f(x), f(y)) \in E_2$. En tal caso:

- Diremos que f es un **isomorfismo** entre G_1 y G_2.
- Escribiremos $G_1 \cong G_2$.

Teorema 25

La relación de isomorfismo $\cong$ es una relación de equivalencia (es decir, es refleja, simétrica y transitiva).

Camino (informal)

Un camino es un grafo cuyos vértices pueden dibujarse en una línea tal que dos vértices son adyacentes si y sólo si aparecen consecutivos en la línea.

Camino (formal)

Considere un grafo $G_n^P = (V_n^P, E_n^P)$, donde

$$V_n^P = \{v_1, \ldots, v_n\}$$
$$E_n^P = \{(v_i, v_j) \mid i \in \{1, \ldots, n-1\} \wedge j = i+1\}$$

Un **camino** (de n vértices) es un grafo isomorfo a G_n^P.
A la clase de equivalencia $[G_n^P]_{\cong}$ la llamaremos P_n: los caminos con n vértices.

Ciclo (informal)

Un ciclo es un grafo cuyos vértices pueden dibujarse en un círculo tal que dor vértices son adyancentes si y sólo si aparecen consecutivos en él.

Ciclo (formal)

Considere un grafo $G_n^C = (V_n^C, E_n^C)$, donde

$$V_n^C = \{v_1, \ldots, v_n\}$$
$$E_n^C = \{(v_i, v_j) \mid i \in \{1, \ldots, n-1\} \wedge j = i+1\} \cup \{(v_n, v_1)\}$$

Un **ciclo** (de n vértices) es un grafo isomorfo a G_n^P.
A la clase de equivalencia $[G_n^C]_{\cong}$ la llamaremos C_n: los ciclos con n vértices.

Grafo completo

Un grafo completo es un grafo en el que todos los pares de vértices son adyacentes.
A la clase de equivalencia de los grafos completos de n vértices la llamaremos K_n.

Grafo bipartito

Un grafo $G = (V, E)$ se deice bipartito si V se puede particionar en dos conjuntos no vacíos V_1 y V_2 tales que para toda arista $(x, y) \in E$, $x \in V_1$ e $y \in V_2$, o $x \in V_2$ e $y \in V_1$.
Es decir,

- $V = V_1 \cup V_2$
- $V_1 \cap V_2 = \varnothing$
- Cada arista une a dos vértices en conjuntos distintos de la partición.

Grafo bipartito completo

Un grafo bipartito completo es un grafo bipartito en que cada vértice es adyacente a todos los de la otra partición.
A la clase de equivalencia de los grafos bipartitos completos la llamaremos $K_{n,m}$, donde n y m son los tamaños de las particiones.

Subgrafo

Dado un grafo $G = (V_G, E_G)$, un grafo $H = (V_H, E_H)$ es un subgrafo de G (denotado como $H \subseteq G$) si $V_H \subseteq V_G$, $E_H \subseteq E_G$ y E_H sólo contiene aristas entre vértices de V_H.

Clique

Dado un grafo $G = (V_G, E_G)$, un clique en G es un conjunto de vértices $K \subseteq V_G$ tal que

$$\forall\, v_1, v_2 \in K.\, (v_1, v_2) \in E_G$$

Conjunto independiente

Dado un grafo $G = (V_G, E_G)$, un conjunto independiente en G es un conjunto de vértices $K \subseteq V_G$ tal que

$$\forall\, u, v \in K.\, (u, v) \notin E_G$$

Complemento

Dado un grafo $G = (V_G, E_G)$, el complemento de G es el grafo $\overline{G} = (V_G, \overline{E_G})$, donde $(u, v) \in E_G \leftrightarrow (u, v) \notin \overline{E_G}$.

Grafo autocomplementario

Un grafo G se dice autocomplementario si $G \cong \overline{G}$.

Teorema 26

Dado un grafo $G = (V, E)$, un conjunto $V' \subseteq V$ es un clique en G si y sólo si es un conjunto independiente en $\overline{G}$.

Matriz de adyacencia

Dado un grafo $G = (V, E)$, como E es una relación binaria podemos representarla en una matriz, llamada matriz de adyacencia de G.
Por ejemplo, si $V = \{1, 2, 3, 4\}$ y $E = \{(1, 2), (1, 4), (2, 1), (2, 3), (2, 4), (3, 2), (3, 4), (4, 1), (4, 2), (4, 3)\}$, entonces

$$M_G = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

- Si el grafo es simple, la diagional sólo contiene ceros.
- Si el grafo es no dirigido, entonces $M_G = M_G^T$.

Matriz de incidencia

Además de la matriz de adyacencia, podemos usar una matriz de incidencia A_G:

- Etiquetamos las aristas de G.
- Cada fila de la matriz representará un vértice, y cada columna a una arista.
- Cada posición de la matriz tendra un 1 si la arista de la columna *incide* en el vértice de al fila.

Por ejemplo, si $V = \{1,2,3,4\}$ y $E = \{(1,2), (1,4), (2,1), (2,3), (2,4), (3,2), (3,4), (4,1), (4,2), (4,3)\}$, entonces

$$A_G = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Grado de un grafo

Dado un grafo G y un vértice v de él, el grado de v (denotado como $\delta_G(v)$) es la cantidad de aristas que inciden en v.

Vecindad de un grafo

Dado un grafo G y un vértice v de él, la vecindad de v es el conjunto de vecinos de v:

$$N_G(v) = \{u \,|\, (v,u) \in E\}$$

En un grafo simple, $\delta_G(v) = |N_G(v)|$

Teorema 27 (Handshaking lemma)

Si $G = (V, E)$ es un grafo sin rulos, entonces

$$\sum_{v \in V} \delta_G(v) = 2|E|$$

Es decir, la suma de los grafos de los vértices es dos veces la cantidad de aristas.

- **Corolario:** En un grafo sin rulos siempre hay una cantidad par de vértices de grado impar.

Caminatas

Una caminata en un grafo $G = (V, E)$ es una secuencia de vértices $(v_0, v_1, v_2, \ldots, v_k)$, con $v_0, \ldots, v_k \in V$, tal que $(v_{i-1}, v_i) \in E$, con i entre 1 y k.
Una caminata **cerrada** en un grafo es una caminata que empieza y termina en el mismo vértice: $v_0 = v_k$.

Caminos y ciclos

Un **camino** es un grafo es una caminata en la que no se repiten aristas.
Un **ciclo** en un grafo es una caminata cerrada en la que no se repiten aristas.

Largo de una caminata, camino o ciclo

Corresponde a la cantidad de aristas que lo componen. Si está compuesto por un único vértice (sin aristas), diremos que tiene largo 0.

Vértices conectados

Dos vértices x e y en un grafo G estan conectados si existe un camino en G que empieza en x y termina en y.
Nota: Que dos vértices "estén conectados" implica una relación de equivalencia.

Componente conexa

Dado un vértice v de un grafo G, su clase de equivalencia bajo la relación "estar conectados" es una componente conexa de G.
En general, diremos que la componente conexa tambien contiene a las aristas entre los vértices de ella.

Grafo conexo

Un grafo G se dice conexo si todo par de vértices $x, y \in V$ está conectado. En otro caso, G es disconexo.
En otras palabras, esta definición es equivalente a decir que G tiene sólo una componente conexa.

Teorema 28

Un grafo G con n vértices y k aristas tiene al menos $n - k$ componentes conexas.

- **Corolario:** Si un grafo G con n vértices es conexo, tiene al menos $n - 1$ aristas.

Arista de corte

Una arista de corte en un grafo G es una arista tal que al eliminarla aumenta la cantidad de componentes conexas de G.

Vértice de corte

Un vértice de corte en un grafo G es un vértice tal que al eliminarlo (junto con todas sus aristas incidentes) aumenta la cantidad de componentes conexas de G.

Teorema 29

Una arista en un grafo G es de corte si y sólo si no pertenece a ningún ciclo en G.

Lema 2

En un grafo simple G, toda caminata cerrada de largo impar contiene un ciclo de largo impar.

Teorema 30

Un grafo simple conexo G es bipartito si y sólo si no contiene ningún ciclo de largo impar.

Multigrafo

Sea V un conjunto de vértices, E un conjunto de aristas y $S \subseteq \mathcal{P}(V)$ tal que

$$S = \{\{u, v\} \mid u \in V \wedge v \in V\}$$

Un multigrafo $G = (V, E, f)$ es un trío ordenado donde $f : E \to S$ es una función que asigna un par de vértices a cada arista en E.

Ciclo Euleriano

Un ciclo euleriano en un (multi)grafo G es un ciclo que contiene a todas las aristas y vértices de G.

- Es un ciclo, por lo tanto no puede repetir aristas.
- Pueden repetirse vértices.
- Iremos que G es un **grafo Euleriano** si contiene un ciclo Euleriano.

Teorema 31

Un (multi)grafo sin rulos es Euleriano si y sólo si es conexo y todos sus vértices tienen grado par.

Camino Euleriano

Un camino Euleriano en un (multi)grafo G es un camino no cerrado que contiene a todas las aristas y vértices de G.

Teorema 32

Un (multi)grafo tiene un camino Euleriano si y sólo si es conexo y contiene exactamente dos vértices de grado impar.

Ciclo Hamiltoniano

Un ciclo Hamiltoniano en un grafo G es un ciclo en G que contiene a todos sus vértices una única vez cada uno (excepto por el inicial y el final).

- Diremos que G es un **grafo Hamiltoniano** si contiene un ciclo Hamiltoniano.
- No hay ninguna relación entre grafos Eulerianos y Hamiltonianos.

Árbol

Un grafo $T = (V, E)$ es un **árbol** si para cada par de vértices $x, y \in V$ existe un único camino entre ellos. Por lo tanto, siempre es conexo.

Bosque

Un grafo $T = (V, E)$ es un **bosque** si para cada par de vértices $x, y \in V$, si existe un camino entre ellos, este es único. Un bosque es un conjunto de árboles.

Árboles con raíz

Distinguimos uno de los vértices $r \in V$, al que llamaremos la **raíz** del árbol. Los vértices de grado menor o igual a 1 se llaman **hojas**. Los dibujamos con la raíz arriba y los demás vértices hacia abajo.

Definiciones alternativas de árbol

Hay muchas definiciones equivalentes para los árboles:

1. Un grafo $T = (V, E)$ es un **árbol** si y sólo si es conexo y acíclico.
2. Un grafo $T = (V, e)$ es un **árbol** si y sólo si es conexo y todas sus aristas son de corte.

Teorema 33

Todo árbol es un grafo bipartito.

Teorema 34

Si T es un árbol y v es una hoja de él, entonces el grafo $T - v$ (el grafo que resulta de quitar el vértice y sus aristas incidentes) es un árbol.

Otra definición útil de árbol

Un grafo $T = (V, E)$ con n vértices es un **árbol** si y sólo si es conexo y tiene exactamente $n - 1$ aristas.

Profundidad, altura, ancestros, padre, hijos y hermanos

Sea $T = (V, E)$ un árbol con raíz r y x un vértice cualquiera.

- La **profundidad** de x es el largo del camino que lo une con r (r tiene profundidad 0).
- La **altura** o **profundidad** del árbol es el máximo de las profundidades de sus vértices.
- Los **ancestros** de x son los vértices que aparecen en el camino entre él y r. Note que x es ancestro de sí mismo.
- El **padre** de x es su ancestro (propio) de mayor profundidad. Diremos que x es **hijo** de su padre.
- Dos vértices x e y con el mismo padre son **hermanos**.

Árboles binarios

Un árbol con raíz se dice **binario** si todo vértice tiene grafo a lo más 3; o equivalentemente, si todo vértice tiene a lo más dos hijos. Podemos distinguir entre hijos izquierdos y derechos.

Teorema 35

La cantidad de vértices sin hijos de un árbol binario es la cantidad de vértices con exactamente dos hijos más 1.

Árbol binario completo

Un **árbol binario completo** es un árbol binario tal que:

1. Todas las hojas están a la misma profundidad.
2. Todos los vértices que no son hojas tienen exactamente dos hijos.

Teorema 36

1. Un árbol binario completo de altura H tiene exactamente 2^H hojas.
2. Un árbol binario completo de altura H tiene exactamente $2^{H+1} - 1$ vértices.
3. Si H es la altura de un árbol binario completo con n vértices, entonces $H \leq \log_2(n)$.

10. Teoría de números

Recordando la relación |

La relación *divide a*, denotada por |, sobre los enteros sin el 0, es una relación tal que a está relacionado con b si y sólo b es múltiplo de a:

$$a|b \text{ si y sólo si } \exists k \in \mathbb{Z} \text{ tal que } b = ka$$

Recordando la relación $\equiv_n$

La relación *equivalencia módulo n*, denotada por $\equiv_n$, sobre los enteros, es una relación tal que a está relacionado con b si y sólo si $n|(b-a)$:

$$a \equiv_n b \text{ si y sólo si } n|(b-a)$$
$$a \equiv_n b \text{ si y sólo si } \exists k \in \mathbb{Z} \text{ tal que } (b-a) = kn$$

- La relación $\equiv_n$ es una relación de equivalencia.
- Podemos tomar el conjunto cuociente generado por ella sobre $\mathbb{Z}$.
- Usando las clases de equivalencia, definimos la suma y la multiplicación.

Suma y multiplicación

Dado $n \in \mathbb{N}, n > 0$, definimos

$$\mathbb{Z}_n = \mathbb{Z}/\equiv_n$$

y sus operaciones

$$[i] + [j] = [i + j]$$
$$[i] \cdot [j] = [i \cdot j]$$

Operación módulo n

La operación **módulo** n entrega el resto de la división por n. Se escribe $a \bmod n$ o $a\%n$.
Con esta operación podemos redefinir la suma y multiplicación en $\mathbb{Z}_n$:

$$[i] + [j] = (i + j) \bmod n$$
$$[i] \cdot [j] = (i \cdot j) \bmod n$$

- Una observación importante es que siempre se cumple que
$$0 \leq a \bmod n < n$$

Teorema 37

$a \equiv_n b$ si y sólo si $a \bmod n = b \bmod n$.

- **Corolario:** $a \equiv_n a \bmod n$.

Teorema 38

Si $a \equiv_n b$ y $c \equiv_n d$, entonces

$$(a + c) \equiv_n (b + d)$$
$$(a \cdot c) \equiv_n (b \cdot d)$$

- **Corolario:**

$$(a + b) \bmod n = ((a \bmod n) + b \bmod n) \bmod n$$
$$a \cdot b \bmod n = ((a \bmod n)(b \bmod n)) \bmod n$$

Teorema 39 (Fermat)

Si p es un numero primo, para cualquier entero a se cumple que $a^p \equiv_p a$.

- **Corolario:** Si p es un número primo y a es un entero que no es múltiplo de p, entonces $a^{p-1} \equiv_p 1$.

Máximo común divisor (informal)

Dados dos números a y b, su máximo común divisor, denotado como $MCD(a, b)$, es el máximo natural n tal que $n|a$ y $n|b$.

Teorema 40

Si $b > 0$, entonces $MCD(a, b) = MCD(b, a \bmod b)$.

$$MCD(a, b) = \begin{cases} a & b = 0 \\ MCD(b, a \bmod b) & b > 0 \end{cases}$$

Calculando enteros con MCD

Con el algoritmo (ecuación) anterior, se puede calcular $s, t \in \mathbb{Z}$ sabiendo que

$$MCD(a, b) = s \cdot a + t \cdot b$$

Algoritmo extendido del MCD

Sea $a \geq b$.

1. Definimos una sucesión $\{r_i\}$ como:
$$r_0 = a,\ r_1 = b,\ r_{i+1} = r_{i-1} \bmod r_i$$
2. Definimos sucesiones $\{s_i\}, \{t_i\}$ tales que:
$$s_0 = 1,\ t_0 = 0$$
$$s_1 = 0,\ t_1 = 1$$
$$r_i = s_i \cdot a + t_i \cdot b$$
3. Calculamos estas sucesiones hasta un k tal que $r_k = 0$.
4. Entonces, $MCD(a, b) = r_{k-1} = s_{k-1} \cdot a + t_{k-1} \cdot b$.

Algoritmo extendido del MCD (alternativa)

Sea $a \geq b$.

1. Definimos una sucesión $\{r_i\}$ como:
$$r_0 = a,\ r_1 = b,\ r_{i+1} = r_{i-1} \bmod r_i$$
2. Definimos sucesiones $\{s_i\}, \{t_i\}$ tales que:
$$s_0 = 1,\ t_0 = 0$$
$$s_1 = 0,\ t_1 = 1$$
$$s_{i+1} = s_{i-1} - \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor \cdot s_i,\ t_{i+1} = t_{i-1} - \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor \cdot t_i$$
3. Calculamos estas sucesiones hasta un k tal que $r_k = 0$.
4. Entonces, $MCD(a, b) = r_{k-1} = s_{k-1} \cdot a + t_{k-1} \cdot b$.

Inverso

b es inverso de a en módulo n si $a \cdot b \equiv_n 1$. Se puede denotar como a^{-1}. Ojo: no es lo mismo que $\frac{1}{a}$.

Teorema 41

a tiene inverso en módulo n si y sólo si $MCD(a, n) = 1$. Si se cumple esto, diremos que a y n son **primos relativos** o **coprimos**.

Método para calcular el inverso

La demostración del teorema 41 nos da un método para calcular el inverso:

- Usamos el algoritmo extendido del máximo común divisor para encontrar s y t tales que $1 = s \cdot a + t \cdot n$.
- s será el inverso de a en módulo n.

Dados $a, b, n \in \mathbb{Z}$, si $a \equiv_n b$ tambien podemos escribir:

$$a \equiv b \pmod{n}$$

Esta es la notación más usada en la literatura.

Congruencia lineal

Una congruencia lineal es una ecuación de la forma

$$ax \equiv b \pmod{n}$$

donde $n \in \mathbb{N} - \{0\}$, $a, b \in \mathbb{Z}$ y x es una variable.

Corolario teorema 41

Si a y n son primos relativos, entonces $ax \equiv b \pmod{n}$ tiene solución en $\mathbb{Z}_n$.

Teorema 42 (Chino del Resto)

Sean $m_1, m_2, \ldots, m_n$ con $m_i > 1$ tal que m_i, m_j son primos relativos con $i \neq j$. Para $a_1, a_2, \ldots, a_n \in \mathbb{Z}$, el sistema de ecuaciones:

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_n \pmod{m_n} \end{aligned}$$

tiene una única solución en $\mathbb{Z}_m$ con $m = \prod_{i=1}^{n} m_i$

Lema 3

Sean $m_1, m_2 > 1$ coprimos y $u, v \in \mathbb{Z}$. Si $u \equiv v \pmod{m_1}$ y $u \equiv v \pmod{m_2}$, entonces

$$u \equiv v \pmod{m_1 \cdot m_2}$$

Este lema se puede generalizar para n coprimos.

11. Complejidad computacional

Problemas de decisión (informal)

Son los problemas para los cuales sus respuestas posibles son SI o NO.

Problemas de decisión (formal)

Un problema de decisión π se compone de un conjunto de instancias I_π y un lenguaje $L_\pi \subseteq I_\pi$, y se define como

Dado un elemento $w \in I_\pi$, determinar si $w \in L_\pi$

Ejemplos:

- PRIMO:
  - $I_{\text{PRIMO}} = \mathbb{N}$.
  - $L_{\text{PRIMO}} = \{p \in \mathbb{N} \mid p \text{ es primo}\}$
- EULERIANO:
  - $I_{\text{EULERIANO}} = \{G \mid G \text{ es un grafo}\}$
  - $L_{\text{EULERIANO}} = \{G \mid G \text{ es un grafo con ciclo eulariano}\}$
- SAT:
  - $I_{\text{SAT}} = L(P)$
  - $L_{\text{SAT}} = \{\alpha \in L(P) \mid \alpha \text{ es satisfacible}\}$

Un algoritmo A resuelve un problema de decisión π si para cada input $w \in I_\pi$, el algoritmo A responde SI cuando $w \in L_\pi$, y responde NO cuando $w \in I_\pi - L_\pi$ (en este caso diremos que $w \notin L_\pi$).

Clase de complejidad DTIME

La clase de complejidad **DTIME**$(T(n))$ es el conjunto de problemas para los cuales existe un algoritmo que lo resuelve de complejidad $O(T(n))$, donde n es el tamaño del input:

$\text{DTIME}(T(n)) = \{\pi \mid \pi \text{ es un problema de decisión para el cual existe un algoritmo } A \text{ que lo resuelve y que para todo } w \in I_\pi \text{ corre en } O(T(|w|))\}$.

Los ejemplos anteriores pertenecen a las clases de complejidad:

- PRIMO $\in \text{DTIME}(10^n)$
- EULERIANO $\in \text{DTIME}(n^2)$
- SAT $\in \text{DTIME}(2^n)$

Clase de complejidad PTIME

La clase de complejidad **PTIME** o simplemente P se define como

$$\text{P} = \bigcup_{k=0}^{\infty} \text{DTIME}(n^k)$$

La clase PTIME contiene a todos los problemas que pueden resolverse en tiempo polinomial. En general diremos que estos problemas son *tratables*. Por ejemplo, EULERIANO $\in$ P.

Clase de complejidad NP

Como definición informal, la clase de complejidad NP contiene a todos los problemas de decisión para los cuales es posible verificar una solución al problema de forma eficiente. Sin embargo, no necesariamente encontrar la solución es fácil.

En una vía más formal, la clase de complejidad NP contiene a todos los problemas de decisión π para los cuales se cumple lo siguiente:

Si $w \in L_\pi$, entonces existe un *certificado* $c(w)$, de tamaño polinomial respecto a w, tal que existe un algoritmo que usando $c(w)$ puede determinar en tiempo polinomial si $w \in L_\pi$.

Teorema 43

P $\subseteq$ NP. Este teorema nos permite establecer una primera cota para las clases P y NP.

Reducciones polinomiales

Dados dos problemas de decisión $\pi = (I_\pi, L_\pi)$ y $\pi' = (I_{\pi'}, L_{\pi'})$, diremos que π' *se reduce desde* π si dada una instancia $w \in I_\pi$ existe una función $A : I_\pi \to I_{\pi'}$ polinomial en $|w|$ tal que

$$w \in L_\pi \leftrightarrow A(w) \in L_{\pi'}$$

En este caso diremos que π' es **por lo menos tan difícil** como π y lo denotaremos como $\pi \propto \pi'$.

Fórmula en 3-CNF

Decimos que una fórmula α está en 3-CNF si es una conjunción de cláusulas de exactamente 3 literales cada una.

- En general aplicamos la restricción de 3 literales al conjunto que representa a cada cláusula.
- Por lo tanto, no pueden repetirse literales en la misma cláusula.

SAT-3CNF

Definimos SAT-3CNF como

$$I_{\text{SAT-3CNF}} = \alpha \in L(P) \text{ en 3-CNF}$$
$$L_{\text{SAT-3CNF}} = \{\alpha \in L(P) \text{ en 3-CNF} \mid \alpha \text{ es satisfacible}\}$$

En este caso, diremos que SAT-3CNF es **por lo menos tan difícil** como SAT-3CNF.

Teorema 44

$\propto$ es un preorden (refleja y transitiva).

NP-hard

El hecho de que $\propto$ sea un preorden nos permite establecer una jerarquía en cuanto a la dificultad de los problemas.
Sea π un problema de decisión. Diremos que π es NP-hard si para todo $\pi' \in NP$ se tiene que $\pi' \propto \pi$.
En otras palabras, los problemas NP-hard son por lo menos tan difíciles como todos los problemas en NP.

NP-completo

El único problema con la definición de NP-hard es que no nos dice nada sobre las cotas superiores de los problemas. La siguiente definición logra capturar la verdadera complejidad de la clase NP:
Sea π un problema de decisión. Diremos que π es NP-completo si:

- $\pi \in$ NP.
- $\pi \in$ NP-hard.

Dado que cualquier problema en NP se puede reducir a uno NP-completo, podemos concluir que los problemas NP-completos son los más difíciles de la clase NP.

Teorema 45 (de Cook)

SAT-CNF es NP-completo.

Problema de decisión: Clique

$$I_{\text{CLIQUE}} = \text{Todos los grafos no dirigidos.}$$
$$L_{\text{CLIQUE}} = \{(G, k) \mid G \text{ tiene un clique de tamaño } k \wedge k > 2\}$$

CLIQUE es NP-completo.

Nota al lector

Este documento está libre tanto para su edición como para su uso o distribución. Si deseas clonar el repositorio de GitHub (`crow-rojas/Resumen-IIC1253-PUC`) y editar el documento a tu manera, eres totalmente bienvenido/a.

Son también bienvenidas recomendaciones y/o sugerencias para mejorar aún más el documento, así que si se te ocurre alguna no dudes en escribirme a mi correo `carojas37@uc.cl`.

¡Éxito con el estudio!