Apuntes de clase — Matemática Discreta

Luis Dissett

Primer Semestre, 2006

Índice general

Prólogo (provisorio)

Estos apuntes (o mejor dicho, este borrador de apuntes) resumen el contenido de los cursos de Matemática Discreta que he dictado en las Facultades de Matemática e Ingeniería entre los años 2000 y 2004.

Espero que le sean útiles, en primer lugar, a mis alumnos en futuras versiones de estos cursos, y también a otros profesores que deseen usarlos como referencia para sus cursos.

Capítulo 1

Lógica Proposicional

1.1. Proposiciones, conectivos, fórmulas proposicionales

Definición 1. Una *proposición* es una afirmación que puede ser verdadera o falsa.
Una proposición es *atómica* si es imposible descomponerla en proposiciones más simples.

Para combinar proposiciones y formar nuevas proposiciones más complejas usamos los llamados *conectivos lógicos*.

1.1.1. Algunos conectivos

Negación La negación de una proposición es la que afirma que la proposición original no es verdadera. Generalmente se obtiene agregando "no" (o "no es verdad que") antes de la proposición original.

Conjunción La conjunción de dos proposiciones es la que afirma que ambas proposiciones son verdaderas. Se obtiene intercalando "y" entre las dos proposiciones originales.

Disyunción La disyunción de dos proposiciones es la que afirma que al menos una de las dos proposiciones es verdadera. Se obtiene intercalando "o" entre las dos proposiciones originales.

Condicional La proposición condicional entre dos proposiciones (el *antecedente* y el *consecuente*) es la que afirma que, cada vez que el antecedente es verdadero, el consecuente también lo es. Puede ser obtenido precediendo el antencedente por "si" e intercalando "entonces" entre el antecedente y el consecuente.

Bicondicional La proposición bicondicional entre dos proposiciones es la que afirma que, o ambas son verdaderas, o ambas son falsas. Puede ser obtenida intercalando la frase "si y sólo si", o bien "siempre y cuando" entre las dos proposiciones originales.

Ejercicio. ¿Cuántos conectivos binarios (esencialmente diferentes) es posible definir?

Respuesta. *Hay un total de* $2^4 = 16$ *conectivos binarios distintos.*

Postergaremos la justificación de esto hasta que veamos *tablas de verdad*.

1.2. Fórmulas proposicionales

Para trabajar con proposiciones, las representamos por *fórmulas*, llamadas —apropiadamente— *fórmulas proposicionales*. En estricto rigor, una fórmula proposicional es simplemente una secuencia de símbolos, a la cual se asocia una proposición.

Las proposiciones atómicas son representadas por *variables proposicionales*, que generalmente son letras mayúsculas: P, Q, R, S, etc. Si debemos utilizar demasiadas variables proposicionales, recurrimos a sub-índices; así, podemos tener variables proposicionales P_1, P_2, P_3,..., Q_1, Q_2, Q_3, etc.

Las proposiciones compuestas son representadas como sigue: si dos proposiciones φ y ψ son representadas, respectivamente, por las fórmulas proposicionales p y q, entonces representamos (y leemos) las siguientes proposiciones compuestas como sigue:

Proposición	Representación	Lectura
Negación de φ	$(\neg p)$	*no p.*
Conjunción de φ y ψ	$(p \wedge q)$	*p y q.*
Disyunción de φ y ψ	$(p \vee q)$	*p o q.*
Condicional entre φ y ψ	$(p \to q)$	*si p entonces q.*
Bicondicional entre φ y ψ	$(p \leftrightarrow q)$	*p si y sólo si q.*

1.3. Algunos comentarios

Note que tenderemos a identificar las fórmulas proposicionales con las proposiciones que representan; o sea, a veces diremos "proposición" cuando lo correcto sería decir "fórmula proposicional".

En particular, identificaremos las proposiciones atómicas con las variables proposicionales que las representan.

Además, en la medida de lo posible, cuando no se preste a confusiones, eliminaremos los paréntesis más exteriores de $(p \wedge q)$, $(p \vee q)$, etc. En general, intentaremos eliminar la mayor cantidad posible de paréntesis, en la medida en que esto no deje ambigua a la fórmula.

1.4. Valor de verdad de proposiciones compuestas

Una proposición compuesta, formada por subproposiciones, será verdadera o falsa dependiendo de los valores de verdad de las subproposiciones que la forman.

Ejemplo. Considérese la proposición $((\neg P \wedge Q) \vee (R \vee P))$, y supóngase que P y Q son verdaderas y R falsa. Entonces:

- $\neg P$ será falsa (negación de una proposición verdadera).
- $(\neg P \wedge Q)$ será falsa (conjunción de una proposición falsa y una verdadera).
- $(R \vee P)$ será verdadera (disyunción de una proposición falsa y una verdadera).
- Finalmente, $((\neg P \wedge Q) \vee (R \vee P))$ será verdadera (disyunción de una proposición falsa y una verdadera).

1.5. Asignaciones de verdad

Definición 2. Una *asignación de verdad* es una lista de "valores de verdad" (VERDADERO o FALSO) asociadas a las proposiciones atómicas que forman las proposiciones con las que estamos trabajando[1]

Así, una asignación de verdad determina el valor de verdad de cada una de las proposiciones compuestas con las que estamos trabajando.

En el ejemplo anterior, nuestra suposición de que P y Q eran verdaderas y R falsa constituye una asignación de verdad.

[1] Estrictamente hablando, una asignación de verdad asigna VERDADERO o FALSO a las *variables proposicionales* que estamos considerando.

1.6. Tablas de Verdad

Resumiremos los valores de verdad de que toma una proposición compuesta, para todas las posibles asignaciones de verdad, en una *tabla de verdad*. En dichas tablas, usaremos los símbolos 1 para indicar VERDADERO y 0 para indicar FALSO.

Ejemplo. La tabla de verdad para $((\neg P \wedge Q) \vee (R \vee P))$ es como sigue:

P	Q	R	$((\neg P$	$\wedge$	$Q)$	$\vee$	$(R$	$\vee$	$P)$
0	0	0	1	0		0		0	
0	0	1	1	0		1		1	
0	1	0	1	1		1		0	
0	1	1	1	1		1		1	
1	0	0	0	0		1		1	
1	0	1	0	0		1		1	
1	1	0	0	0		1		1	
1	1	1	0	0		1		1	

1.7. Tautologías y contradicciones

Definición 3. Una *tautología* es una proposición que es verdadera en toda asignación de verdad.

Ejemplo.

$$((P \to Q) \leftrightarrow (\neg P \vee Q)).$$

Una *contradicción* es una proposición que es falsa en toda asignación de verdad.

Ejemplo.

$$(P \wedge \neg P).$$

Notación. Denotaremos las tautologías por T_o y las contradicciones por F_o.

Ejemplo. Al tratar de demostrar que una cierta conclusión c se desprende de una serie de premisas $p_1, p_2, \ldots, p_n$, en el fondo estamos tratando de probar que

$$(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \to c$$

es una tautología.

1.8. Consecuencia lógica

Consideremos dos situaciones:

1. Las proposiciones

$$\begin{aligned} p &: (3 < 2 \vee 1 + 3 = 4), \quad \text{y} \\ q &: (2 < 5 \to 1 + 3 = 4). \end{aligned}$$

son verdaderas, mientras que la proposición $r : 1 + 3 = 4$ también lo es.

2. Las proposiciones

$$\begin{aligned} s &: (2 < 3 \to 2 < 4), \quad \text{y} \\ t &: (2 < 4 \to 2 + 2 = 4). \end{aligned}$$

son verdaderas, mientras que la proposición $u : (2 < 3 \to 2 + 2 = 4)$ también lo es.

Consideremos sólo las *estructuras* de las proposiciones involucradas. Podemos escribir:

$$\begin{aligned} p &: X \vee Y & s &: F \to G \\ q &: Z \to Y & t &: G \to H \\ r &: Y & u &: F \to H, \end{aligned}$$

donde X, Y, Z, F, G y H son proposiciones atómicas.

Vistas de esta forma, en que p, q, r, s, t y u son sólo consideradas en función de su estructura,

¿Será verdad que toda asignación de verdad que hace verdaderas a p y a q debe también hacer verdadera a r?

¿Será verdad que toda asignación de verdad que hace verdaderas a s y a t debe también hacer verdadera a u?

Observamos que, para p, q y r, es posible hallar una asignación de verdad (para las proposiciones atómicas X, Y y Z) que hace verdaderas a p y q, pero hace falsa a r. Simplemente, debemos asignar VERDADERO a X, FALSO a Y y FALSO a Z.

En el segundo caso, sin embargo, es *imposible* hallar una asignación de verdad que haga verdaderas a s y t y haga falsa a u (¿cómo es posible convencerse de esto, aparte de probando todas las asignaciones de verdad posibles?).

1.9. Definición de consecuencia lógica

Definición 4. Dados un conjunto $\Sigma = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ de proposiciones, y una proposición ψ, diremos que ψ *es consecuencia lógica de* Σ si toda asignación de verdad que hace verdaderas a $\varphi_1, \varphi_2, \ldots, \varphi_n$ hace verdadera también a ψ.

Notación. Si $\Sigma = \{\varphi\}$, en lugar de decir que ψ es consecuencia lógica de $\{\varphi\}$ diremos simplemente que ψ es consecuencia lógica de φ.

Notación. Denotaremos el hecho de que ψ es consecuencia lógica de Σ por $\Sigma \models \psi$

Observación. Note que ψ es tautología si y sólo si $\emptyset \models \psi$. Denotamos este hecho por $\models \psi$.

¿Cómo *probar* consecuencia lógica?

- Tablas de verdad (fuerza bruta).
- Tratando de construir una asignación de verdad que haga verdaderas todas las proposiciones de Σ y falsa a ψ, y mostrando que esto es imposible.

¿Cómo *refutar* consecuencia lógica?

Respuesta: Encontrando una asignación de verdad que haga verdaderas todas las proposiciones de Σ y falsa a ψ.

Esto puede ser hecho usando tablas de verdad (de nuevo, fuerza bruta) o bien tratando de construir dicha asignación de verdad (y teniendo éxito en el intento).

Una estrategia que puede ser útil en este sentido es la de *resolución*, que será explicada en la primera ayudantía.

1.10. Equivalencia lógica

Definición 5. Sean P y Q dos proposiciones cualesquiera (atómicas o compuestas).

Diremos que P y Q son *lógicamente equivalentes* si toda asignación de verdad que hace verdadera a P hace verdadera a Q y viceversa.

Notación. Denotaremos la equivalencia lógica entre P y Q por $P \Leftrightarrow Q$.

Una formulación equivalente

Teorema. Dadas dos proposiciones P y Q, se tiene que $P \Leftrightarrow Q$ si y sólo si $P \leftrightarrow Q$ es una tautología.

Demostración. Ejercicio. □

1.11. Las leyes de la lógica

Las siguientes equivalencias lógicas son conocidas como las *leyes de la lógica*:

Ley de la *doble negación*

$$\neg\neg p \Leftrightarrow p.$$

Leyes de *de Morgan*

$$\begin{aligned} \neg(p \vee q) &\Leftrightarrow \neg p \wedge \neg q. \\ \neg(p \wedge q) &\Leftrightarrow \neg p \vee \neg q. \end{aligned}$$

Leyes *conmutativas*

$$\begin{aligned} p \vee q &\Leftrightarrow q \vee p. \\ p \wedge q &\Leftrightarrow q \wedge p. \end{aligned}$$

Leyes *asociativas*

$$\begin{aligned} p \vee (q \vee r) &\Leftrightarrow (p \vee q) \vee r. \\ p \wedge (q \wedge r) &\Leftrightarrow (p \wedge q) \wedge r. \end{aligned}$$

Leyes *distributivas*

$$\begin{aligned} p \vee (q \wedge r) &\Leftrightarrow (p \vee q) \wedge (p \vee r). \\ p \wedge (q \vee r) &\Leftrightarrow (p \wedge q) \vee (p \wedge r). \end{aligned}$$

Leyes *de idempotencia*

$$\begin{aligned} p \vee p &\Leftrightarrow p. \\ p \wedge p &\Leftrightarrow p. \end{aligned}$$

Leyes *de elemento neutro*

$$\begin{aligned} p \vee F_o &\Leftrightarrow p. \\ p \wedge T_o &\Leftrightarrow p. \end{aligned}$$

Leyes *de elemento inverso*

$$\begin{aligned} p \vee \neg p &\Leftrightarrow T_o. \\ p \wedge \neg p &\Leftrightarrow F_o. \end{aligned}$$

Leyes *de dominación*

$$\begin{aligned} p \vee T_o &\Leftrightarrow T_o. \\ p \wedge F_o &\Leftrightarrow F_o. \end{aligned}$$

Leyes *de absorción*

$$\begin{aligned} p \vee (p \wedge q) &\Leftrightarrow p. \\ p \wedge (p \vee q) &\Leftrightarrow p. \end{aligned}$$

Ley *de la implicación*

$$p \to q \Leftrightarrow \neg p \vee q.$$

1.12. Reglas de sustitución

- En las leyes anteriores, es posible reemplazar *todas las ocurrencias* de una proposición atómica (p, q, r, etc.) por *cualquier proposición*, y la ley seguirá siendo válida.
- Sea P una proposición cualquiera. Si en P se reemplazan una o más ocurrencias de una proposición Q por una proposición Q' *lógicamente equivalente a* Q, la proposición P' resultante será lógicamente equivalente a P.

1.13. El principio de dualidad

Las leyes anteriores (excepto en dos casos) están agrupadas en pares. En cada caso, una de las leyes es lo que llamamos el *dual* de la otra.

Definición 6. Sea F una proposición que contiene sólo los conectivos $\neg$, $\wedge$ y $\vee$. Entonces la *proposición dual* de F (que denotamos por F^d) es la proposición que resulta de reemplazar cada aparición de $\wedge$ por $\vee$ (y viceversa), y cada aparición de T_o por F_o (y viceversa).

Teorema (Principio de dualidad). *Si $F \Leftrightarrow G$, entonces $F^d \Leftrightarrow G^d$.*

No veremos la demostración de este teorema.

Así, basta probar una de las leyes de cada par de duales.

1.14. Formas Normales

Recordemos que estamos estudiando *fórmulas proposicionales*, o sea, representaciones de proposiciones como *strings* de sí mbolos. Estos símbolos, o bien son conectivos, o bien representan proposiciones atómicas.

Llamamos *literales* a los símbolos que representan proposiciones atómicas, y al símbolo de negación ($\neg$) seguido de un símbolo que representa a una proposición atómica.

A continuación demostraremos que toda *fórmula proposicional* puede ser escrita como:

- conjunción de disyunciones de literales, o bien
- disyunción de conjunciones de literales.

A la primera forma la llamamos *Forma Normal Conjuntiva* (FNC), y a la segunda *Forma Normal Disyuntiva* (FND).

Toda proposición puede ser re-escrita en FNC o en FND.

Más precisamente, se tiene el siguiente

Teorema. Dada una fórmula proposicional φ, existen fórmulas proposicionales φ' y φ'' tales que φ' está en FNC, φ'' está en FND, $\varphi \Leftrightarrow \varphi'$ y $\varphi \Leftrightarrow \varphi''$.

Demostración. Sean $X_1, X_2, \ldots, X_n$ las variables proposicionales que aparecen en φ.

Considere la tabla de verdad para la fórmula proposicional φ. Esta tabla de verdad tiene 2^n líneas.

Cada línea de la tabla de verdad corresponde a una asignación de verdad a las variables proposicionales $X_1, X_2, \ldots, X_n$, y puede ser interpretada como una conjunción de n literales $L_1 \wedge L_2 \wedge \ldots \wedge L_n$ (donde $L_i = X_i$ si la asignación de verdad asigna VERDADERO a X_i, y $L_i = \neg X_i$ si la asignación de verdad asigna FALSO a X_i).

Así, una fórmula φ'' en FND que es lógicamente equivalente a φ está dada por la disyunción de las conjunciones de literales correspondientes a las líneas de la tabla de verdad donde φ se hace verdadera.

Para hallar una fórmula φ' en FNC que sea lógicamente equivalente a φ, podemos hallar una fórmula ψ en FND lógicamente equivalente a $\neg\varphi$, y después aplicar De Morgan dos veces para transformar $\neg\psi$ en una fórmula en FNC. Como $\psi \Leftrightarrow \neg\varphi$, tenemos que $\neg\psi \Leftrightarrow \varphi$. □

Ejercicio. Aplique el método indicado arriba para encontrar fórmulas en FNC y FND que sean lógicamente equivalentes a $(P \wedge Q) \rightarrow (R \wedge \neg Q)$.

1.15. Reglas de inferencia

Queremos enunciar reglas que nos permitan justificar nuestras deducciones de conclusiones a partir de premisas dadas.

Así, por ejemplo, al hacer una demostración del tipo

$$(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \to c,$$

nos gustaría poder asegurar que la implicación es *válida* (lógicamente verdadera), sin tener que probar todas las combinaciones de valores de verdad (que pueden ser demasiados).

Estudiaremos a continuación *reglas de inferencia* que nos permitirán ir obteniendo conclusiones a partir de un conjunto de premisas, de modo de terminar obteniendo la conclusión deseada.

En lo que sigue, P, Q, R, etc., representan proposiciones cualesquiera, no necesariamente atómicas.

Las primeras reglas de inferencia que consideraremos están dadas por las equivalencias que aparecen en las leyes de la lógica, a las que posiblemente habremos aplicado las *reglas de sustitución*.

Así, si tenemos como premisa P y una ley de la lógica nos dice que $P \Leftrightarrow Q$, entonces podemos deducir Q.

Otras reglas que estudiaremos son:

- La *ley del silogismo*.
- *Modus Ponens*, o "método de la afirmación".
- *Modus Tollens*, o "método de la negación".
- La *regla de resolución*.
- La *regla de conjunción*.
- La *ley del silogismo disyuntivo*.
- La *regla de contradicción*.
- La *regla de simplificación conjuntiva*.
- La *regla de amplificación disyuntiva*.
- La *regla de demostración condicional*.
- La *regla de demostración por casos*.
- La *regla del dilema constructivo*.
- La *regla del dilema destructivo*.

1.16. Las reglas

Ley del silogismo Cada vez que tengamos como premisas proposiciones de las formas $P \to Q$ y $Q \to R$, tenemos derecho a deducir $P \to R$.

En símbolos:

$$\frac{\begin{array}{l} P \to Q \\ Q \to R \end{array}}{P \to R}$$

Modus ponens Cada vez que tengamos como premisas proposiciones de las formas $P \to Q$ y P, tenemos derecho a deducir Q.

En símbolos:

$$\frac{\begin{array}{l} P \to Q \\ P \end{array}}{Q}$$

Ejemplo. Supongamos que tenemos por premisas $(p \wedge q)$ y $((p \wedge q) \to (\neg q \vee r))$. Aplicando *modus ponens*, vemos que

$$\frac{\begin{array}{l} (p \wedge q) \to (\neg q \vee r) \\ (p \wedge q) \end{array}}{(\neg q \vee r)}$$

Modus tollens Cada vez que tengamos como premisas proposiciones de las formas $P \to Q$ y $\neg Q$, tenemos derecho a deducir $\neg P$.

En símbolos:

$$\frac{\begin{array}{l} P \to Q \\ \neg Q \end{array}}{\neg P}$$

Ejemplo. Supongamos que tenemos por premisas $(p \vee q) \to r$ y $\neg r$. Aplicando *modus tollens*, vemos que

$$\frac{\begin{array}{l} (p \vee q) \to r \\ \neg r \end{array}}{\neg(p \vee q)}$$

Regla de conjunción Cada vez que tengamos como premisas proposiciones de las formas P y Q, tenemos derecho a deducir $P \wedge Q$.

En símbolos:

$$\frac{\begin{array}{l} P \\ Q \end{array}}{P \wedge Q}$$

Ley del silogismo disyuntivo Cada vez que tengamos como premisas proposiciones de las formas $P \vee Q$ y $\neg P$, tenemos derecho a deducir Q.

En símbolos:

$$\frac{\begin{array}{l} P \vee Q \\ \neg P \end{array}}{Q}$$

Regla de contradicción Cada vez que tengamos como premisa una proposición de la forma $P \to F_o$, tenemos derecho a deducir $\neg P$.

En símbolos:

$$\frac{P \to F_o}{\neg P}$$

Regla de simplificación conjuntiva Cada vez que tengamos como premisa una proposición de la forma $P \wedge Q$, tenemos derecho a deducir P.

En símbolos:

$$\frac{P \wedge Q}{P}$$

Regla de amplificación disyuntiva Cada vez que tengamos como premisa una proposición de la forma P, tenemos derecho a deducir $P \vee Q$.

En símbolos:

$$\frac{P}{P \vee Q}$$

Regla de demostración condicional Cada vez que tengamos como premisas proposiciones de las formas $P \wedge Q$ y $P \to (Q \to R)$, tenemos derecho a deducir R.

En símbolos:

$$\frac{\begin{array}{l} P \wedge Q \\ P \to (Q \to R) \end{array}}{R}$$

Regla de demostración por casos Cada vez que tengamos como premisas proposiciones de las formas $P \to R$ y $Q \to R$, tenemos derecho a deducir $(P \vee Q) \to R$.

En símbolos:

$$\frac{\begin{array}{l} P \to R \\ Q \to R \end{array}}{(P \vee Q) \to R}$$

Regla del dilema constructivo Cada vez que tengamos como premisas proposiciones de las formas $P \to Q$, $R \to S$ y $P \vee R$, tenemos derecho a deducir $Q \vee S$.

En símbolos:

$$\frac{\begin{array}{l} P \to Q \\ R \to S \\ P \vee R \end{array}}{Q \vee S}$$

Regla del dilema destructivo Cada vez que tengamos como premisas proposiciones de las formas $P \to Q$, $R \to S$ y $\neg Q \vee \neg S$, tenemos derecho a deducir $\neg P \vee \neg R$.

En símbolos:

$$\frac{\begin{array}{l} P \to Q \\ R \to S \\ \neg Q \vee \neg S \end{array}}{\neg P \vee \neg R}$$

1.17. Sistemas deductivos

Llamamos *sistema deductivo* a cualquier conjunto de reglas (de entre las mencionadas, u otras) que, agregadas a las leyes de la lógica, nos permitan deducir conclusiones a partir de premisas.

Entre las características que nos interesa que tenga un posible sistema deductivo se destacan dos:

(a) Que sea *correcto* (en inglés, *sound*), o sea, que cualquier conclusión que se obtenga a partir de las premisas deba ser, necesariamente, consecuencia lógica de éstas; en otras palabras, que no sea posible deducir nada que no sea consecuencia lógica de las premisas; y

(b) Que sea *completo*, o sea, que si φ es consecuencia lógica de las premisas, entonces φ puede ser deducido de éstas.

1.18. Ejemplo de uso de las reglas

Consideremos el sistema deductivo formado por todas las reglas enunciadas anteriormente.

Las usemos para demostrar que

$$p \to q$$

es consecuencia lógica de

$$\{p \wedge (r \vee q), \neg r \vee q, q \to r\}.$$

El argumento que damos es el siguiente:

$p \wedge (r \vee q)$	Premisa	(1.1)
$\neg r \vee q$	Premisa	(1.2)
$q \to r$	Premisa	(1.3)
$r \vee q$	Simplificación conjuntiva, 1.1	(1.4)
$(\neg r \vee q) \wedge (r \vee q)$	Regla de conjunción, 1.2 y 1.4	(1.5)
$(\neg r \wedge r) \vee q$	Ley distributiva, 1.5	(1.6)
q	Elemento neutro, 1.6	(1.7)
$\neg p \vee q$	Amplificación disyuntiva, 1.7	(1.8)
$p \to q$	Ley de la implicación, 1.5	(1.9)

1.19. Otro ejemplo

Usemos este mismo sistema deductivo para demostrar que

$$q \wedge r$$

es consecuencia lógica de

$$\{p, p \to q, s \vee r, \neg s \wedge \neg t\}\,.$$

Nuestro argumento es como sigue:

p	Premisa	(1.10)
$p \to q$	Premisa	(1.11)
q	Modus Ponens, 1.10 y 1.11	(1.12)
$s \vee r$	Premisa	(1.13)
$\neg\neg s \vee r$	Ley de la doble negación, 1.13	(1.14)
$\neg s \to r$	Ley de la implicación, 1.14	(1.15)
$\neg s \wedge \neg t$	Premisa	(1.16)
$\neg s$	Simplificación conjuntiva, 1.16	(1.17)
r	Modus ponens, 1.15 y 1.17	(1.18)
$q \wedge r$	Regla de conjunción, 1.12 y 1.18	(1.19)

1.20. Resolución

Un sistema deductivo muy importante en Inteligencia Artificial es el formado por la única regla de *resolución*:

Cada vez que tengamos como premisas proposiciones de las formas $P \vee Q$ y $\neg Q \vee R$, tenemos derecho a deducir $P \vee R$.

En símbolos:

$$\frac{\begin{array}{c} P \vee Q \\ \neg Q \vee R \end{array}}{P \vee R}$$

Este sistema deductivo es correcto (*sound*) y completo (no lo demostraremos aquí), y no sólo puede ser usado para *deducir* sino también para refutar.

Note que de P y $\neg P$ podemos deducir F_o, ya que $P \Leftrightarrow P \vee F_o$ y $\neg P \Leftrightarrow \neg P \vee F_o$.

Deducción usando resolución

Para demostrar (o refutar) el que $\Sigma \models Q$, hacemos lo siguiente:

- Transformamos todas las fórmulas de Σ a FNC.
- Negamos la conclusión deseada (Q) y la ponemos en FNC.
- Aplicamos la regla de resolución, hasta que: o bien derivamos una contradicción, o bien la regla no puede ser aplicada.
- Si se llegó a una contradicción, entonces Q es consecuencia lógica de Σ. En caso contrario, no lo es.

Ejemplo. Demostraremos que $\Sigma = \{P \vee Q, P \to R, Q \to R\} \models R$, usando resolución.
Solución:
T ras transformar las fórmulas de Σ a FNC, y agregar la negación de R en FNC, obtenemos:

$$\{P \vee Q, \neg P \vee R, \neg Q \vee R, \neg R\}\,.$$

Sucesivas aplicaciones de la regla de resolución dan:

- de $P \vee Q$ y de $\neg P \vee R$, obtenemos $Q \vee R$;
- de $\neg Q \vee R$ y $\neg R$ obtenemos $\neg Q$;
- de $Q \vee R$ y $\neg Q$ obtenemos R;
- finalmente, de R y $\neg R$ obtenemos nuestra contradicción.

O sea, $\Sigma \models R$.

Árboles de refutación

PENDIENTE

1.21. Ejercicios

1. Sean p y q dos proposiciones atómicas tales que $p \to q$ es falsa. Determine el valor de verdad de:

   a) $p \wedge q$,
   b) $\neg p \vee q$,
   c) $q \to q$,
   d) $\neg q \to \neg p$.

2. Sean p, q, r las siguientes proposiciones:

   p**:** hago la tarea;
   q**:** juego al tenis;
   r**:** el sol está brillando;
   s**:** la humedad es baja.

   Traduzca a símbolos:

   a) Si el sol está brillando, entonces juego tenis.
   b) Hacer la tarea es requisito para que jugar al tenis.
   c) Si el sol está brillando y la humedad es baja entonces juego tenis.

d) Ni el sol está brillando ni la humedad es baja.

e) La humedad no es baja, a menos que el sol esté brillando .

3. Demuestre que $p \to q$ es consecuencia lógica de $\{p \wedge (r \vee q), \neg r \vee q, q \to r\}$.

4. Demuestre que $p \to q$ no es consecuencia lógica de $\{p \vee r, \neg r \vee q, (p \wedge r) \to (q \vee r\}$

5. Demuestre, *sin usar tablas de verdad*, que si p, q y r son proposiciones atómicas, entonces

$$(p \to (q \vee r)) \Leftrightarrow ((p \wedge \neg q) \to r)).$$

6. Sean p y q dos proposiciones atómicas.

a) Verifique que $p \to (q \to (p \wedge q))$ es una tautología.

b) Demuestre, usando la parte (a), las reglas de sustitución y las leyes de la lógica, que $(p \vee q) \to (q \to q)$ es una tautología.

c) ¿Es $(p \vee q) \to (q \to (p \wedge q))$ una tautología?

7. Repita los ejercicios de los tipos "¿es la fórmula Q consecuencia lógica de ..." , esta vez usando árboles de refutación.

8. Recuerde que:

a) un *literal* es, o bien una proposición atómica, o la negación de una proposición atómica;

b) una disyunción de literales es llamada una *cláusula*;

c) una fórmula proposicional está en *Forma Normal Conjuntiva* (FNC) si es una conjunción de cláusulas:

$$\bigwedge_{i=0}^{n} (\bigvee_{j=0}^{m_i} l_{ij}).$$

Demuestre que, dada una proposición φ en forma normal conjuntiva (o sea, $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_n$ donde $C_1, C_2, \ldots C_n$ son cláusulas), es posible encontrar una fórmula φ', que también está en forma normal conjuntiva, **donde cada cláusula está formada por exactamente tres literales**, y tal que φ' es *satisfactible* (o sea, existe una asignación de verdad que la hace verdadera) si y sólo si φ lo es.

Capítulo 2

Lógica de predicados

2.1. Definiciones básicas

Ejemplo. La frase "x es par e y es impar" no es una proposición (no es ni verdadera ni falsa).

Si reemplazamos x e y por dos números enteros, la frase se transforma en una proposición, y su valor de verdad dependerá de los valores que tengan x e y.

Diremos que la frase anterior es una "proposición abierta" o "predicado". Como este predicado depende de x y de y, lo denotaremos por una letra con x e y entre paréntesis, por ejemplo:

$$P(x, y).$$

Si denotamos "x es par" por $Q(x)$ e "y es impar" por $R(y)$, podemos escribir

$$P(x, y) \Leftrightarrow Q(x) \wedge R(y).$$

2.1.1. Predicados atómicos

En el ejemplo anterior, podemos distinguir entre los predicados $P(x, y)$ por un lado, y los predicados $Q(x)$ y $R(y)$ por el otro: el primero está formado por otros predicados, mientras que los últimos no pueden ser descompuestos en predicados más pequeños.

A los predicados que no pueden ser descompuestos en predicados más pequeños los llamaremos *predicados atómicos*. Usamos estos predicados para representar relaciones.

A veces escribimos las relaciones como símbolos *entre* los elementos que relacionan (ejemplo: no escribimos $<(x, y)$ sino $x < y$).

2.1.2. Variables, constantes, funciones y operaciones

En un predicado, encontramos símbolos que representan *variables* (x, y, z, etc.), *constantes* (0, 1, 2, π, y otros), *funciones* y *operaciones*, y otros predicados.

Ejemplo. En el predicado

$$u + f(v, 0) = 2 \cdot w$$

encontramos las constantes 0 y 2, las variables u, v y w, el símbolo de función f, y los símbolos de operación $+$ y $\cdot$.

2.1.3. Interpretaciones y dominios

No podemos estudiar un predicado sin asignarle un significado a los distintos símbolos que en él aparecen.

Para esto, al analizar un predicado, consideraremos una *interpretación*, que consiste en un *dominio* o *universo* $\mathcal{D}$ y en asignaciones de significado a las constantes, símbolos de función y operación y a los símbolos que representan relaciones (predicados atómicos).

Ejemplo. Al considerar el predicado

$$S(x) : x \neq 0 \rightarrow x \cdot x \neq 0,$$

el dominio de interpretación puede ser $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ o incluso algún conjunto no numérico (por ejemplo, el conjunto de matrices de 2×2).

En este último caso, el símbolo 0 no representará a un número, sino a una matriz.

2.1.4. Cuantificadores

En matemáticas, muchas afirmaciones son de la forma "todos los elementos de $\mathcal{D}$ (un dominio dado) satisfacen el predicado $P(x)$" o bien "hay al menos un elemento de $\mathcal{D}$ que satisface $P(x)$".

En el primer caso, abreviaremos usando el símbolo $\forall$ y en el segundo usaremos el símbolo $\exists$. Así, si $P(x)$ es un predicado que depende sólo de x, podemos formar las proposiciones:

$\forall x(P(x))$	:	Si reemplazamos x por cualquier elemento de $\mathcal{D}$, entonces $P(x)$ se hace verdadera.
$\exists x(P(x))$	:	En $\mathcal{D}$ hay al menos un valor tal que, al reemplazar x por dicho valor, la proposición resultante es verdadera.

Los símbolos $\forall$ y $\exists$ son llamados *cuantificador universal* y *cuantificador existencial* respectivamente.

2.1.5. Variables *libres* y *ligadas*

En un predicado, las variables pueden aparecer relacionadas con un cuantificador (*ligadas*) o *libres*.

Un predicado que no tiene variables libres es una proposición.

Un predicado con variables libres es una *proposición abierta.*

2.2. Verdad lógica, consecuencia lógica y equivalencia lógica

2.2.1. Interpretaciones y valores de verdad

Las proposiciones en lógica de predicados son verdaderas o falsas dependiendo de la interpretación en que sean consideradas.

Ejemplo. Sea $P(x, y)$ un predicado binario (con dos argumentos). La proposición

$$\exists x \forall y(P(x, y))$$

es falsa en la interpretación en que $\mathcal{D} = \mathbb{N}$ y $P(x, y)$ representa la relación $x > y$.

La misma proposición es verdadera si $\mathcal{D} = \mathbb{N}$, y $P(x, y)$ representa la relación "*x divide a y*".

2.2.2. Proposiciones válidas (lógicamente verdaderas)

El concepto de tautología, o *proposición lógicamente verdadera* de la lógica proposicional tiene su contraparte en lógica de predicados.

Si P es una proposición en lógica de predicados, decimos que P es *lógicamente verdadera* (o *válida*) si se hace verdadera en toda interpretación.

Ejemplo. Sea $P(x)$ un predicado.

La proposición

$$\forall x(P(x) \vee \neg P(x))$$

es lógicamente verdadera.

Demostración. Sea $\mathfrak{I}$ una interpretación *arbitraria* con dominio $\mathcal{D}$, y para cada $c \in \mathcal{D}$ consideremos el valor de verdad que $\mathfrak{I}$ le asigna a $P(c)$. Si este valor de verdad es VERDADERO, entonces $P(c) \vee \neg P(c)$ es VERDADERO(disyunción de una proposición verdadera y una falsa); y en caso contrario $P(c) \vee \neg P(c)$ *también es* VERDADERO (disyunción de una proposición falsa y una verdadera).

O sea: la proposición $P(c) \vee \neg P(c)$ es VERDADERO, *sin importar qué elemento* $c \in \mathcal{D}$ *tomemos*.

Pero entonces la proposición $\forall x(P(x) \vee \neg P(x))$ es verdadera *en la interpretación* $\mathfrak{I}$.

Como $\mathfrak{I}$ es una interpretación arbitraria, hemos demostrado que $\forall x(P(x) \vee \neg P(x))$ se hace verdader a en toda interpretación, o sea, es lógicamente verdadera. □

2.2.3. Consecuencia lógica

Sea Σ un conjunto de proposiciones en lógica de predicados. Sea Q otra proposición en lógica de predicados.

Diremos que Q es *consecuencia lógica de* Σ (o que Σ *lógicamente implica* Q) si toda interpretación que hace verdaderas todas las proposiciones de Σ necesariamente hace verdadera a Q.

Si $\Sigma = \{P\}$ (o sea, si Σ consiste de una sola proposición) entonces diremos que Q es consecuencia lógica de P (en lugar de decir que lo es de $\{P\}$).

Al igual que en el caso proposicional, si Q es consecuencia lógica de Σ, anotaremos $\Sigma \models Q$.

Ejemplo. La proposición

$$\forall x \exists y(P(x, y))$$

es consecuencia lógica de

$$\exists y \forall x(P(x, y)).$$

Demostración. Sea $\mathfrak{I}$ una interpretación arbitraria que hace verdadera a $\exists y \forall x(P(x, y))$, y sea $\mathcal{D}$ su dominio.

Como $\exists y \forall x(P(x, y))$ es verdadera en $\mathfrak{I}$, existe algún elemento $d \in \mathcal{D}$ tal que

$$\forall x(P(x, d)) \tag{2.1}$$

es verdadera en $\mathfrak{I}$. Queremos demostrar que $\forall x \exists y(P(x, y))$ es verdadera bajo la interpretación $\mathfrak{I}$.

Para esto, debemos demostrar que dado cualquier elemento $c \in \mathcal{D}$, la proposición $\exists y(P(c, y))$ es verdadera en $\mathfrak{I}$. Pero esto es cierto ya que, dado $c \in \mathcal{D}$, debido a (2.1) se tiene $P(c, d)$, por lo que $\exists y(P(c, y))$ es verdadera en $\mathfrak{I}$.

Como $c \in \mathcal{D}$ era arbitrario, hemos demostrado que $\forall x \exists y(P(x, y))$ es verdadera bajo la interpretación $\mathfrak{I}$.

Finalmente, como $\mathfrak{I}$ es una interpretación arbitraria (de la que sólo supusimos que hacía verdadera a $\exists y \forall x(P(x, y))$), hemos demostrado que

$$\exists y \forall x(P(x, y)) \models \forall x \exists y(P(x, y)).$$

□

2.2.4. Equivalencia lógica

Si P y Q son dos proposiciones en lógica de predicados, diremos que ellas son *lógicamente equivalentes* si toda interpretación le asigna el mismo valor de verdad a ambas.

Notación. Al igual que en el caso proposicional, si P y Q son lógicamente equivalentes, anotaremos $P \Leftrightarrow Q$.

Ejemplo. La proposición

$$\forall x(Q(x) \wedge R(x))$$

es lógicamente equivalente a

$$\forall x(Q(x)) \wedge \forall x(R(x)).$$

Demostración. Demostrar que dos proposiciones son lógicamente equivalentes es lo mismo que demostrar que cada una de ellas es consecuencia lógica de la otra. Dejamos cada una de estas dos demostraciones como ejercicio al lector. □

2.2.5. Resumen de definiciones

Una proposición P es ...	En lógica proposicional	En lógica de predicados
lógicamente verdadera ...	si toda asignación de verdad la hace verdadera.	si toda interpretación la hace verdadera.
lógicamente equivalente a otra Q ...	si toda asignación de verdad las hace ambas verdaderas o ambas falsas.	si toda interpretación las hace verdaderas o ambas falsas.
consecuencia lógica de otra Q ...	si toda asignación de verdad que hace verdadera a Q hace verdadera a P.	si toda interpretación que hace verdadera a Q hace verdadera a P.

2.3. Negación de proposiciones con cuantificadores

Para negar proposiciones (o predicados) que contienen cuantificadores pueden usarse las siguientes equivalencias:

$$\begin{aligned} \neg\forall x(P(x)) &\Leftrightarrow \exists x(\neg P(x)), \\ \neg\exists x(P(x)) &\Leftrightarrow \forall x(\neg P(x)). \end{aligned}$$

Dejamos la demostración como ejercicio.

2.4. Reglas de inferencia usando predicados

Además de las reglas de inferencia dadas en el capítulo sobre lógica proposicional, en lógica de predicados pueden usarse las siguientes:

Especificación universal

Si se tiene la proposición $\forall x(P(x))$, y $a \in \mathcal{D}$ es arbitrario, podemos deducir $P(a)$.

Generalización existencial

Si se tiene la proposición $P(a)$ (donde $a \in \mathcal{D}$), podemos deducir $\exists x(P(x))$.

Generalización universal

Si, dado $a \in \mathcal{D}$ *arbitrario*, es posible demostrar $P(a)$, entonces es posible deducir $\forall x(P(x))$.

Especificación existencial

Si se ha demostrado la proposición $\exists x(P(x))$, entonces es posible deducir la proposición $P(a)$, *donde* $a \in \mathcal{D}$ *es un elemento arbitrario que no ha sido usado en la demostración de* $\exists x(P(x))$.

2.5. Teorías matemáticas

PENDIENTE.

2.6. Ejercicios

1. Para las siguientes proposiciones, el universo consiste en todos los enteros distintos de cero, y los significados de los símbolos de función y operaciones aritméticas es el usual. Determine el valor de verdad de cada proposición, y escriba la negación de cada una de ellas.

   a) $\exists x \exists y (x \cdot y = 1)$.
   b) $\exists x \forall y (x \cdot y = 1)$.
   c) $\forall x \exists y (x \cdot y = 1)$.
   d) $\forall x \forall y \left[\operatorname{sen}^2 x + \cos^2 x = \operatorname{sen}^2 y + \cos^2 y\right]$.
   e) $\exists x \exists y \left[(2x + y = 5) \wedge (x - 3y = -8\right]$.
   f) $\exists x \exists y \left[(3x - y = 7) \wedge (2x + 4y = 3\right]$.
   g) $\exists x \exists y \left[(2x + y = 5) \vee (x - 3y = -8\right]$.
   h) $\exists x \exists y \left[(3x - y = 7) \vee (2x + 4y = 3\right]$.

2. Repita el ejercicio anterior, ahora tomando como universo todos los números reales distintos de cero.

3. Repita el ejercicio anterior, ahora tomando como universo todos los números reales (incluyendo al cero).

4. Escriba las negaciones de las siguientes proposiciones:

   a) $\exists x \left[p(x) \vee q(x)\right]$.
   b) $\forall x \left[p(x) \wedge \neg q(x)\right]$.
   c) $\forall x \left[(p(x) \rightarrow q(x)\right]$.
   d) $\exists x \left[(p(x) \vee q(x)) \rightarrow \neg r(x)\right]$.

5. Demuestre que las proposiciones $\forall x(Q(x) \wedge R(x))$ y $\forall x(Q(x)) \wedge \forall x(R(x))$ son lógicamente equivalentes.

6. Demuestre que las proposiciones $\forall x(Q(x) \vee R(x))$ y $\forall x(Q(x)) \vee \forall x(R(x))$ no son lógicamente equivalentes.

7. Una de las dos proposiciones presentadas anteriormente es consecuencia lógica de la otra. Demuestre este hecho.

8. Demuestre las equivalencias lógicas

$$\begin{aligned} \neg \forall x(P(x)) &\Leftrightarrow \exists x(\neg P(x)), \\ \neg \exists x(P(x)) &\Leftrightarrow \forall x(\neg P(x)). \end{aligned}$$

9. Demuestre que, dada cualquier proposición en lógica de predicados, existe una proposición lógicamente equivalente a ella en que todos los cuantificadores están al principio de la proposición, y se aplican globalmente a ella (ésta es llamada la *Forma Normal Prenex*, y es utilizada en inteligencia artificial).

10. En cada uno de los siguientes casos, decida si la equivalencia lógica expresada es verdadera o no. En caso de que su respuesta sea negativa, indique si una de las implicaciones lógicas es correcta o si ambas son falsas. *Justifique sus respuestas*.

    a) $\forall x \left[p(x) \rightarrow q(x)\right] \Leftrightarrow \forall x(p(x)) \rightarrow \forall x(q(x))$.
    b) $\exists x \left[p(x) \rightarrow q(x)\right] \Leftrightarrow \exists x(p(x)) \rightarrow \exists x(q(x))$.
    c) $\forall x \left[p(x) \vee q(x)\right] \Leftrightarrow \forall x(p(x)) \vee \forall x(q(x))$.
    d) $\exists x \left[p(x) \vee q(x)\right] \Leftrightarrow \exists x(p(x)) \vee \exists x(q(x))$.
    e) $\forall x \left[p(x) \wedge q(x)\right] \Leftrightarrow \forall x(p(x)) \wedge \forall x(q(x))$.
    f) $\exists x \left[p(x) \wedge q(x)\right] \Leftrightarrow \exists x(p(x)) \wedge \exists x(q(x))$.

11. Dé un ejemplo de una interpretación que haga verdaderas $\forall x \left[p(x)\right]$ y $\forall x \left[q(x) \rightarrow p(x)\right]$ pero que no haga verdadera $\forall x \left[q(x)\right]$. ¿Qué se puede concluir de este ejemplo?

Capítulo 3

Teoría de Conjuntos

3.1. Definiciones básicas

3.1.1. Nociones primitivas

Consideramos como "primitivas" (i.e., no necesitan explicación) las siguientes nociones:

- elemento,
- conjunto,
- pertenencia ($\in$).

Definiremos los otros conceptos relacionados con conjuntos a partir de estas nociones básicas.

3.1.2. Subconjuntos, igualdad de conjuntos

Definición 7. Si A y B son conjuntos, decimos que A es *subconjunto* de B (en símbolos, $A \subseteq B$) sii

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

Definición 8. Si A y B son conjuntos, diremos que A y B son *iguales* sii $A \subseteq B$ y $B \subseteq A$.
En símbolos, $A = B \leftrightarrow (A \subseteq B \wedge B \subseteq A)$.

3.1.3. Maneras de definir un conjunto

Definiremos conjuntos de dos formas distintas:

- Por *extensión*, o sea, listando todos sus elementos.

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

- Por *comprensión*, o sea, dando una propiedad $\varphi(x)$ que caracterice a los elementos del conjunto (y sólo a dichos elementos).

  Así, si $A = \{x : \varphi(x)\}$, entonces

  $$\forall x(x \in A \leftrightarrow \varphi(x)).$$

  Ejemplo. $\mathbb{Z}_5 = \{x : x \in \mathbb{N} \wedge x < 5\}$.

  Note que consideramos que $\mathbb{N}$ contiene al cero. Discutiremos esto más adelante.

3.1.4. Conjuntos con elementos repetidos

Note que de la definición de igualdad se deduce que en un conjunto da lo mismo si "se repiten los elementos". Así, por ejemplo, $\{1, 1, 1, 1, 2, 2, 2\} = \{1, 2\}$.

A veces es necesario considerar "multiconjuntos": objetos similares a los conjuntos, pero donde es necesario tomar en cuenta la cantidad de veces que se repite cada elemento. En estos apuntes no discutiremos multiconjuntos en detalle.

3.1.5. El conjunto vacío

Definición 9. El *conjunto vacío* (denotado por $\emptyset$) es un conjunto que no contiene elementos.

Por extensión,

$$\emptyset = \{\}\,.$$

Por comprensión, quisiéramos definir

$$\emptyset = \{x : \varphi(x)\}$$

donde $\varphi(x)$ es una propiedad que es falsa sin importar el valor de x.

Una posible elección de $\varphi(x)$ es:

$$\varphi(x) : x \neq x.$$

Ejercicio. Demuestre que, dado cualquier conjunto A, se tiene:

1. $\emptyset \subseteq A$,
2. $A \subseteq A$.

3.2. La paradoja de Russell

¿Es posible usar cualquier propiedad $\varphi(x)$ al momento de definir un conjunto por comprensión?

Es necesario un poco de cuidado: en 19??, Bertrand Russell demostró que el ser demasiado permisivos con las propiedades usadas para definir conjuntos nos lleva a "paradojas" (contradicciones dentro de la teoría de conjuntos). La más famosa de estas paradojas es la siguiente, llamada "paradoja de Russell": si $\varphi(x)$ es la propiedad "$x \notin x$" entonces definimos el conjunto

$$A = \{x : x \notin x\}$$

y nos formulamos la pregunta:

¿Es A un elemento de A?

De la definición de A, tenemos que

$$A \in A \leftrightarrow A \notin A.$$

O sea, la única manera de que A sea un elemento de sí mismo es ... ¡que no sea un elemento de sí mismo!

Cualquier parecido entre esta paradoja (debida a Bertrand Russell) y la "paradoja del barbero" es absolutamente intencional.

3.2.1. Lidiando con las paradojas

¿Cómo evitar las paradojas en la teoría de conjuntos?

Esencialmente, no cualquier propiedad puede definir un conjunto, por lo cual debemos agregar restricciones.

Ejemplo. Una forma, bastante aceptada, de eliminar paradojas como la de Russell consiste en lo siguiente:

- Se distingue entre "clases" (colecciones arbitrarias de elementos) y "conjuntos" (clases que son elementos de otras clases).

  Las clases que no son conjuntos son llamadas "clases propias".

- Sólo se permiten fórmulas del tipo

$$\varphi(x) : x \text{ es un conjunto y } \ldots \psi(x).$$

Ejercicio. ¿Por qué previene esto la paradoja de Russell?

3.3. Operaciones

A partir de conjuntos dados, es posible construir nuevos conjuntos:

$$\begin{aligned} A \cup B &= \{x : x \in A \vee x \in B\}, \\ A \cap B &= \{x : x \in A \wedge x \in B\}, \\ A \setminus B &= \{x : x \in A \wedge x \notin B\}, \\ \mathcal{P}(A) &= \{x : x \subseteq A\}. \end{aligned}$$

En realidad, necesitamos axiomas que nos aseguren que las clases así definidas son efectivamente conjuntos.

3.4. Las Leyes de la Teoría de Conjuntos

Ley del *doble complemento*

$$(A^c)^c = A.$$

Leyes de *de Morgan*

$$\begin{aligned} (A \cup B)^c &= A^c \cap B^c. \\ (A \cap B)^c &= A^c \cup B^c. \end{aligned}$$

Propiedades *conmutativas*

$$\begin{aligned} A \cup B &= B \cup A. \\ A \cap B &= B \cap A. \end{aligned}$$

Propiedades *asociativas*

$$\begin{aligned} A \cup (B \cup C) &= (A \cup B) \cup C. \\ A \cap (B \cap C) &= (A \cap B) \cap C. \end{aligned}$$

Propiedades *distributivas*

$$\begin{aligned} A \cup (B \cap C) &= (A \cup B) \cap (A \cup C). \\ A \cap (B \cup C) &= (A \cap B) \cup (A \cap C). \end{aligned}$$

Propiedades *de idempotencia*

$$\begin{aligned} A \cup A &= A. \\ A \cap A &= A. \end{aligned}$$

Propiedades *de elemento neutro*

$$\begin{aligned} A \cup \emptyset &= A. \\ A \cap \mathcal{U} &= A. \end{aligned}$$

Propiedades *de elemento inverso*

$$\begin{aligned} A \cup A^c &= \mathcal{U}. \\ A \cap A^c &= \emptyset. \end{aligned}$$

Propiedades *de dominación*

$$\begin{aligned} A \cup \mathcal{U} &= \mathcal{U}. \\ A \cap \emptyset &= \emptyset. \end{aligned}$$

Propiedades *de absorción*

$$\begin{aligned} A \cup (A \cap B) &= A. \\ A \cap (A \cup B) &= A. \end{aligned}$$

3.5. Operaciones generalizadas

Las operaciones binarias definidas anteriormente (unión e intersección) pueden fácilmente ser generalizadas de modo que, en lugar de considerar dos conjuntos, consideren una cantidad (finita) mayor. La forma de hacer esto es, por ejemplo, la siguiente: si $A_1, A_2, \ldots, A_n$ son conjuntos, entonces definimos

$$\bigcup_{i=1}^{n} A_i = \begin{cases} A_1 & \text{si } n = 1, \\ \left(\bigcup_{i=1}^{n-1} A_i \right) \cup A_n & \text{si } n > 1. \end{cases}$$

Si se desea unir o intersectar una cantidad infinita de conjuntos, las definiciones anteriores no son adecuadas. Para definir adecuadamente uniones e intersecciones de una cantidad infinita de conjuntos, usamos la definición siguiente:

Definición 10. Sea A un conjunto cualquiera (del que supondremos que sus elememtos son, a su vez, conjuntos). Definimos dos nuevas clases (y agregamos axiomas que dicen que, si A es conjunto, entonces estas nuevas clases también lo son) como sigue:

- $\bigcup A = \{x : \exists y \in A(x \in y)\}$.
- $\bigcap A = \{x : \forall y \in A(x \in y)\}$.

Ejemplos.

- $\bigcup \emptyset = \emptyset$ (fácil).
- $\bigcap \emptyset = \mathcal{U}$ (no tan fácil).

Demostración. ▪ Dejamos esta demostración como ejercicio.

- Dado $x \in \mathcal{U}$, cualquiera, tenemos que

$$x \in \bigcap \emptyset \quad \text{sii} \quad \forall y \in \emptyset (x \in y) \quad \text{sii} \quad \forall y (y \in \emptyset \to x \in y).$$

O sea,

$$\begin{aligned} x \notin \bigcap \emptyset \quad & \text{sii} \quad \neg \forall y \in \emptyset (x \in y) \\ & \text{sii} \quad \exists y (\neg (y \in \emptyset \to x \in y) \\ & \text{sii} \quad \exists y (\neg (y \in \emptyset \to x \in y)) \\ & \text{sii} \quad \exists y (y \in \emptyset \wedge \neg (x \in y)) \\ & \text{sii} \quad \exists y (y \in \emptyset \wedge x \notin y). \end{aligned}$$

Como esto último es claramente falso, se tiene $x \in \bigcap \phi$, por lo que todo $x \in \mathcal{U}$ satisface $x \in \bigcap \emptyset$, o sea, $\bigcap \emptyset = \mathcal{U}$.

□

3.6. Operaciones con conjuntos de índices

Sea I un *conjunto de índices*, de modo que para cada $i \in I$ existe un único A_i.

Definición 11.

- $\bigcup_{i \in I} A_i = \{x : \exists i \in I (x \in A_i)\}$.
- $\bigcap_{i \in I} A_i = \{x : \forall i \in I (x \in A_i)\}$.

En particular, note que si $I = \{1, 2, \ldots, n\}$, entonces $\bigcup_{i \in I} A_i$ y $\bigcap_{i \in I} A_i$ corresponden a nuestras definiciones anteriores de $\bigcup_{i=1}^{n} A_i$ y $\bigcap_{i=1}^{n} A_i$ respectivamente.

Si $I = \mathbb{N}$, escribimos $\bigcup_{i=0}^{\infty} A_i$ y $\bigcap_{i=0}^{\infty} A_i$ en lugar de $\bigcup_{i \in I} A_i$ y $\bigcap_{i \in I} A_i$.

Notaciones como $\bigcup_{i=1}^{\infty} A_i$ y $\bigcap_{i=1}^{\infty} A_i$ se definen en forma similar.

3.7. Ejercicios

1. Demuestre, usando equivalencias lógicas, las leyes de la teoría de conjuntos (dadas en clases).
2. Repita el ejercicio anterior, ahora usando diagramas de Venn.
3. Demuestre que, si A, B y C son conjuntos arbitrarios ($A, B, C \subseteq \mathcal{U}$), entonces:
   a) $A \subseteq A \cup B$.
   b) $A \cap B \subseteq A$.
   c) $A \subseteq B \leftrightarrow A \cup B = B$.
   d) $A \subseteq B \leftrightarrow A \cap B = A$.
   e) $A \cup B \subseteq C \leftrightarrow (A \subseteq C) \wedge (B \subseteq C)$.
   f) $A \subseteq (B \cap C) \leftrightarrow (A \subseteq B) \wedge (A \subseteq C)$.

g) $A - B \subseteq C \leftrightarrow A - C \subseteq B$.

h) $A \subseteq B \leftrightarrow \forall D \subseteq \mathcal{U}(D \subseteq A \rightarrow D \subseteq B)$.

4. Demuestre las *leyes generalizadas de De Morgan*:

a) $\left(\bigcup_{i\in I} A_i\right)^c = \bigcap_{i\in I} A_i^c$.

b) $\left(\bigcap_{i\in I} A_i\right)^c = \bigcup_{i\in I} A_i^c$.

5. Demuestre las *leyes distributivas generalizadas*:

a) $A \cap \left(\bigcup_{i\in I} B_i\right) = \bigcup_{i\in I}(A \cap B_i)$.

b) $A \cup \left(\bigcap_{i\in I} B_i\right) = \bigcap_{i\in I}(A \cup B_i)$.

6. Dado un conjunto universal $\mathcal{U}$, se define $A \triangle B$ (la *diferencia simétrica* de A y B) como $A \triangle B = (A - B) \cup (B - A)$. Demuestre que:

a) $A \triangle B = B \triangle A$. (o sea, la operación $\triangle$ es conmutativa).

b) $A \triangle A^c = \mathcal{U}$.

c) $A \triangle \mathcal{U} = A^c$.

d) $A \triangle \emptyset = A$ (por lo que el neutro para $\triangle$ es $\emptyset$).

e) $A \triangle A = \emptyset$ (por lo que cada conjunto es su propio inverso respecto a $\triangle$).

f) $(A \triangle B) \triangle C = A \triangle (B \triangle C)$ (o sea, la operación $\triangle$ es asociativa).

g) ¿Qué estructura tiene el conjunto de subconjuntos de $\mathcal{U}$ con la operación $\triangle$?

7. Recuerde que es posible definir la unión e intersección *de una colección cualquiera de conjuntos*, como sigue:

Si S es una colección de conjuntos (todos ellos subconjuntos de un conjunto universal dado $\mathcal{U}$), entonces

$$\begin{aligned} \cap S &= \{x \in \mathcal{U} : \forall y(y \in S \rightarrow x \in y)\}, \\ \cup S &= \{x \in \mathcal{U} : \exists y(y \in S \wedge x \in y)\}. \end{aligned}$$

Demuestre las siguientes propiedades de la unión e intersección así definidas:

a) $\cup\emptyset = \emptyset$.

b) $\cup\{a\} = a$.

c) $\cup\{a, b\} = a \cup b$.

d) Si $A \subseteq B$ entonces $\cup A \subseteq \cup B$.

e) $\cup(A \cup B) = (\cup A) \cup (\cup B)$.

f) Si $x \in A$, entonces $x \subseteq \cup A$.

g) Si $\forall x(x \in A \rightarrow x \subseteq B)$, entonces $\cup A \subseteq B$.

h) $\cap\emptyset = \mathcal{U}$.

i) $\cap\{a\} = a$.

j) $\cap\{a, b\} = a \cap b$.

k) Si $A \subseteq B$ entonces $\cap B \subseteq \cap A$.

l) $\cap(A \cup B) = (\cap A) \cap (\cap B)$.

m) $(\cap A) \cup (\cap B) \subseteq \cap(A \cap B)$.

n) Si $x \in A$, entonces $\cap A \subseteq x$.

ñ) Si $\forall x(x \in A \rightarrow B \subseteq x)$, entonces $B \subseteq \cap A$.

3.8. Aplicación: definición formal de la aritmética

3.8.1. Definición axiomática de $\mathbb{N}$

Para un conjuntista, los números naturales se construyen a partir de la teoría de conjuntos: $0 = \emptyset$ y, dado un conjunto cualquiera x, definimos $\sigma(x) = x \cup \{x\}$ (el *sucesor* de x).

Así,

$$\begin{aligned} 0 &= \emptyset, \\ 1 &= \sigma(0) = \sigma(\emptyset) = \emptyset \cup \{\emptyset\} = \{\emptyset\}, \\ 2 &= \sigma(1) = \sigma(\{\emptyset\}) = \{\emptyset\} \cup \{\{\emptyset\}\} = \{\emptyset, \{\emptyset\}\}, \\ 3 &= \sigma(2) = \sigma(\{\emptyset, \{\emptyset\}\}) = \{\emptyset, \{\emptyset\}\} \cup \{\{\emptyset, \{\emptyset\}\}\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}, \\ &\vdots \end{aligned}$$

Más formalmente, los naturales satisfacen los *axiomas de Peano*:

1. $\emptyset \in \mathbb{N}$.
2. $\forall n(n \in \mathbb{N} \to \sigma n \in \mathbb{N})$.
3. $\forall m \forall n((m \in \mathbb{N} \wedge n \in \mathbb{N} \wedge \sigma m = \sigma n) \to m = n)$.
4. $\forall n(n \in \mathbb{N} \to \sigma n \neq \emptyset)$.
5. Dada cualquier clase S que satisfaga:
   - $\emptyset \in S$,
   - $\forall n(n \in S \to \sigma n \in S)$,
   - $\forall m \forall n((m \in S \wedge n \in S \wedge \sigma m = \sigma n) \to m = n)$ y
   - $\forall n(n \in S \to \sigma n \neq \emptyset)$,

   entonces $\mathbb{N} \subseteq S$.

3.8.2. Operaciones en $\mathbb{N}$

Es posible definir $+$, $\cdot$, etc., en términos de operaciones de conjuntos.

Ejemplo (la suma). Dado $n \in \mathbb{N}$, definimos $s(n, 0) = n$.

Dados $m, n \in \mathbb{N}$, definimos $s(m, \sigma(n)) = \sigma(s(m, n))$.

Ejercicios.

1. Demuestre, usando esta definición de suma, que $3 + 4 = 7$.
2. Defina multiplicación, y demuestre que $3 \cdot 4 = 12$.
3. Demuestre que la suma es conmutativa, asociativa, y tiene elemento neutro.

Capítulo 4

Relaciones

4.1. Definiciones básicas

4.1.1. Pares ordenados

Nos interesa definir formalmente la noción de *par ordenado*. Intuitivamente, queremos definir "par ordenado" como una agregación de dos elementos de modo que dos pares ordenados sean iguales si y sólo si sus elementos respectivos son iguales.

La definición clásica de par ordenado es la siguiente:

Definición 12. Sean $a, b \in \mathcal{U}$ (nuestro *conjunto universo*).

Definimos el *par ordenado* (a, b) como

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

Ejercicio. Demuestre que, si $a, b, c, d \in \mathcal{U}$, entonces

$$(a, b) = (c, d) \leftrightarrow ((a = c) \wedge (b = d)).$$

Ejercicio. ¿Se satisfaría la misma propiedad si hubiéramos definido (a, b) como

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

4.1.2. Producto cartesiano

Sean ahora A y B dos conjuntos cualesquiera. Definimos el *producto cartesiano* de A y B:

Definición 13.

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

4.1.3. Producto de más de dos conjuntos

Si se tienen n conjuntos $A_1, A_2, \ldots, A_n$, entonces definimos

$$A_1 \times A_2 \times A_3 \times \cdots \times A_n = (\ldots((A_1 \times A_2) \times A_3) \times \cdots) \times A_n.$$

4.1.4. Producto cartesiano generalizado

Así como es posible generalizar la unión y la intersección, también podemos generalizar la idea de producto cartesiano.

Definición 14. Sea I un *conjunto de índices*, de modo que para cada $i \in I$ existe un único A_i.

Definimos

$$\prod_{i \in I} A_i$$

como el conjunto de todas las funciones

$$f : I \to \bigcup_{i \in I} A_i$$

tales que, para cada $i \in I$, se tenga $f(i) \in A_i$.

O sea, un elemento de $\prod_{i \in I} A_i$ le asigna a cada elemento $i \in I$ un elemento $f(i)$.

Ejercicio. Explique por qué $A \times B$ y $(A \times B) \times C$ son casos particulares de esta definición.

4.1.5. Las funciones de proyección

En la situación descrita anteriormente, definimos, para cada $i \in I$, la función

$$\pi_i : \prod_{i \in I} A_i \to A_i$$

como

$$\pi_i(f) = f(i).$$

La función π_i es la *proyección sobre la i-ésima coordenada.*

Ejercicio. Explique la relación entre estas funciones de proyección y las del álgebra lineal.

4.1.6. Relaciones binarias

Definición 15.

- Una *relación (binaria) de A en B* es un subconjunto de $A \times B$.
- Una *relación (binaria) en A* es un subconjunto de $A \times A$.

En este curso estaremos interesados mayormente en relaciones binarias definidas en un conjunto dado (excepto cuando hablemos de funciones).

Notación. en vez de escribir $(x, y) \in R$, usualmente escribiremos xRy. En vez de escribir $(x, y) \notin R$, escribiremos $x \not R y$

4.1.7. Relaciones n-arias

Si en lugar de considerar $R \subseteq A \times B$ (o $R \subseteq A \times A$) consideramos $R \subseteq A_1 \times A_2 \times \cdots \times A_n$ (o $R \subseteq \underbrace{A \times A \times \ldots A}_{n \text{ veces}}$), diremos que R es una *relación n-aria*.

Ejemplo. Las tablas de una *base de datos relacional*.

4.1.8. Propiedades de las relaciones binarias

Sea $R \subseteq A \times A$. Dependiendo de las propiedades que satisfaga R, diremos que ésta es:

Refleja si $\forall x \in A(xRx)$.

Irrefleja si $\forall x \in A(x \not R x)$.

Simétrica si $\forall x, y \in A(xRy \to yRx)$.

Antisimétrica si $\forall x, y \in A((xRy \wedge yRx) \to x = y)$.

Transitiva si $\forall x, y, z \in A((xRy \wedge yRz) \to xRz)$.

4.2. Órdenes parciales

Definición 16. Sea A un conjunto. Un *orden parcial* en A es un par $(A, \preceq)$, donde $\preceq$ es una relación en A que es:

1. refleja en A,
2. antisimétrica, y
3. transitiva.

Definición 17. Dado un orden $(A, \preceq)$, el *orden inverso* es el orden $(A, \succeq)$ donde $\succeq$ es la relación inversa de $\preceq$, i.e.,

$$x \succeq y \leftrightarrow y \preceq x.$$

Ejemplos.

- Los órdenes naturales en $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$:

$$(\mathbb{N}, \leq), (\mathbb{Z}, \leq), (\mathbb{Q}, \leq), (\mathbb{R}, \leq).$$

- El orden $|$ (*divide a*) en $\mathbb{N}$:

$$(\mathbb{N}, |).$$

- El orden $\subseteq$ entre los subconjuntos de un conjunto dado $\mathcal{U}$:

$$(\mathcal{P}(\mathcal{U}), \subseteq).$$

A estos ejemplos debemos agregar sus órdenes inversos.

Ejercicio. ¿Es $(\mathbb{Z}, |)$ un orden parcial?

4.2.1. Órdenes estrictos

Sea A un conjunto. Un *orden estricto* en A es un par $(A, \prec)$, donde $\prec$ es una relación en A que es:

1. irrefleja en A,
2. antisimétrica, y
3. transitiva.

Los órdenes estrictos están relacionados con los órdenes parciales, de la siguiente manera:

Teorema.

- *Si $(A, \prec)$ es un orden estricto, entonces $(A, \preceq)$, donde $\preceq$ está definido por $x \preceq y \leftrightarrow (x \prec y \vee x = y)$, es un orden parcial.*
- *Si $(A, \preceq)$ es un orden parcial, entonces $(A, \prec)$, donde $\prec$ está definido por $x \prec y \leftrightarrow (x \preceq y \wedge x \neq y)$, es un orden estricto.*

4.2.2. Órdenes lineales o totales

Definición 18. Sea $(A, \preceq)$ un orden parcial.

Dos elementos $x, y \in A$ son *comparables bajo el orden* $\preceq$ si $x \preceq y$ o $y \preceq x$.

Decimos que $(A, \preceq)$ es un *orden total* o *lineal* si

$$\forall x, y \in A(x \preceq y \vee y \preceq x),$$

o sea, si todos los pares de elementos de A son comparables bajo el orden $\preceq$.

Ejercicio. Indique cuáles de los órdenes parciales dados como ejemplo son lineales.

4.2.3. Elementos maximales y máximos

Sea $(A, \preceq)$ un orden parcial, y sean $S \subseteq A$, $x \in S$.

Definición 19. Decimos que:

- x es un $\preceq$-*elemento maximal de* S si $\forall y \in S(x \preceq y \to x = y)$.
- x es un $\preceq$-*elemento máximo de* S si $\forall y \in S(y \preceq x)$.

Notas:

- Si la relación $\preceq$ es clara del contexto, la omitimos y hablamos simplemente de elementos maximales o máximos.
- De manera análoga se definen los conceptos de *elemento minimal* y *elemento mínimo*.

4.2.4. Cotas, supremos, ínfimos

Sea $(A, \preceq)$ un orden parcial, y sean $S \subseteq A$, $c \in A$.

Definición 20. Decimos que:

- c es una $\preceq$-*cota superior para* S si $\forall x \in S(x \preceq c)$.
- c es un $\preceq$-*supremo para* S si c es $\preceq$-cota superior para S y además, dada cualquier $\preceq$-cota superior c' para S, se tiene $c \preceq c'$.
- S es $\preceq$-*acotado superiormente* si existe una $\preceq$-cota superior para S.

Notas:

- Si la relación $\preceq$ es clara del contexto, la omitimos y hablamos simplemente de cotas superiores, supremos y conjuntos acotados superiormente.
- De manera análoga se definen los conceptos de cota inferior, ínfimo, y conjunto acotado inferiormente.

Teorema. Si $S \subseteq A$ tiene un supremo, éste es único.

Demostración. Ejercicio.

Este teorema nos autoriza a hablar de "*el* supremo de S" (siempre que S tenga al menos un supremo ...). Si éste es el caso, anotaremos $\sup(S)$. Si $S = \{x_1, x_2, \ldots, x_n\}$, entonces anotaremos $\sup\{x_1, x_2, \ldots, x_n\}$ o $\sup(x_1, x_2, \ldots, x_n)$.

Por supuesto, un teorema análogo respecto a ínfimos también es válido.

4.2.5. El axioma del supremo

Sea $(A, \preceq)$ un orden parcial. ¿Será verdad la siguiente afirmación?

Todo subconjunto de A, no vacío y acotado superiormente, tiene supremo.

A esta propiedad la llamamos *el axioma del supremo*.

Aquellos órdenes parciales que lo satisfacen serán llamados (por ahora) *órdenes superiormente completos*. De manera análoga definiremos el concepto de *orden inferiormente completo*.

4.2.6. Órdenes completos

El siguiente teorema nos dice que la distinción entre órdenes superior e inferiormente completos es superflua:

Teorema. Si un orden parcial es superiormente completo, entonces es inferiormente completo (y viceversa).

Demostración. Ejercicio. □

Gracias a este teorema, desde ahora en adelante podemos hablar simplemente de *órdenes completos*.

¿Qué órdenes parciales son completos?

Ejercicio. Demuestre que $(\mathbb{Z}, \leq)$ y $(\mathbb{N}, \leq)$ son órdenes completos.

Ejemplo. Demostraremos que $(\mathbb{Q}, \leq)$ no es un orden completo.

En efecto: sea A el subconjunto de $\mathbb{Q}$ dado por

$$A = \left\{q \in \mathbb{Q} : q^2 < 2\right\}.$$

Claramente, $0 \in A$, por lo que A no es vacío. Por otra parte, si $q \in A$, debe tenerse $q < 2$, por lo que A es acotado superiormente[1].

Para demostrar que $\mathbb{Q}$ no satisface el axioma del supremo, basta probar que no existe ningún racional s tal que $s = \sup A$. Demostraremos esto por contradicción.

Supongamos que existe $s \in \mathbb{Q}$ es tal que $s = \sup A$. En primer lugar, como $1 \in A$, debe tenerse $s > 0$. Por tricotomía, debe darse alguno de los tres casos siguientes: o $s^2 < 2$, o $s^2 > 2$, o $s^2 = 2$. Mostraremos que en los dos primeros casos es imposible que s sea el supremo de A.

Examinemos primero el caso en que $s^2 < 2$. Demostraremos que, en este caso, s no es cota superior de A; para ello, mostraremos que existe un número $s' \in A$ tal que $s < s'$.

En efecto: sea $s' = \dfrac{4}{s + \frac{2}{s}} = \dfrac{4s}{s^2 + 2}$. Para probar que $s' \in A$, vemos que

$$2 - s'^2 = 2 - \frac{16s^2}{(s^2+2)^2} = \frac{2s^4 + 8s^2 + 8 - 16s^2}{(s^2+2)^2} = \frac{2(s^2-2)^2}{(s^2+2)^2} > 0,$$

de donde $s'^2 < 2$, o sea, $s' \in A$. Para probar que $s < s'$, vemos que

$$s' - s = \frac{4s}{s^2+2} - s = \frac{4s - s(s^2+2)}{s^2+2} = \frac{2s - s^3}{s^2+2} = \frac{s(2-s^2)}{s^2+2} > 0,$$

de donde $s < s'$.

Supongamos ahora que $s^2 > 2$. Demostraremos que, en este caso, s no es la cota superior más pequeña de A, ya que existe una cota superior s' de A tal que $s' < s$.

En efecto: sea $s' = \dfrac{s + \frac{2}{s}}{2} = \dfrac{s^2+2}{2s}$. Como

$$s - s' = s - \frac{s^2+2}{2s} = \frac{2s^2 - (s^2+2)}{2s} = \frac{s^2-2}{2s} > 0,$$

[1] También es posible demostrar que todo $q \in A$ es $< 1{,}5$, o incluso $< 1{,}4143\ldots$

vemos que $s' < s$.

Para probar que s' es cota superior de S, basta probar que $s' > 0$ y $s'^2 > 2$ (¿por qué?). Que $s' > 0$ es obvio (¿por qué?). Como

$$s'^2 - 2 = \frac{(s^2+2)^2}{4s^2} - 2 = \frac{s^4 + 4s^2 + 4 - 8s^2}{4s^2} = \frac{s^2 - 4s^2 + 4}{4s^2} = \frac{(s^2-2)^2}{4s^2} > 0,$$

vemos que $s'^2 > 2$. Así, el caso $s^2 > 2$ también es imposible.

Hemos visto que los casos $s^2 < 2$ y $s^2 > 2$ son imposibles, por lo que la única posibilidad es que $s^2 = 2$. Pero este caso también es imposible, debido a la siguiente propiedad (ya conocida por los griegos):

No existe un número racional s tal que $s^2 = 2$.

La demostración de esta propiedad es por contradicción: si s es un racional, puede ser escrito como $s = \frac{m}{n}$ con m y n "primos entre sí" (o sea, sin factores comunes aparte del 1).

Si $s^2 = 2$, tendríamos $m^2 = 2n^2$, por lo que m^2 es par, y por ende m es par. Pero entonces m^2 sería divisible por 4, por lo que $2n^2$ también es divisible por 4, y por lo tanto n^2 sería par y n también sería par.

Pero el hecho de que m y n sean pares contradice la hipótesis de que m y n son primos entre sí. Esta contradicción muestra que m y n no pueden existir.

Así, hemos encontrado un ejemplo que prueba que el conjunto de los racionales no satisface el axioma del supremo.

4.2.7. Los reales y los racionales

Intuitivamente, $\mathbb{R}$ se obtiene a partir de $\mathbb{Q}$ "llenando los agujeros" que se forman en $\mathbb{Q}$ (los supremos que le faltan a algunos conjuntos de racionales).

Más adelante esbozaremos algunas formas de construir los reales a partir de los racionales, y de demostrar que $(\mathbb{R}, \leq)$ es un orden completo.

4.2.8. El teorema de Knaster-Tarski

Un teorema útil al tratar con órdenes completos es el siguiente:

Teorema (Knaster-Tarski). *Sea $(A, \preceq)$ un orden completo, con un elemento máximo y un elemento mínimo.*

Sea $\varphi : A \to A$ una función $\preceq$-monótona; en otras palabras, φ preserva el orden $\preceq$:

$$x \preceq y \to \varphi(x) \preceq \varphi(y).$$

Entonces existe $\tilde{a} \in A$ tal que $\varphi(\tilde{a}) = \tilde{a}$ (o sea, φ tiene un punto fijo).

Demostración. Ejercicio.

Ejercicio. Muestre por qué se necesitan las hipótesis de que A debe tener un elemento máximo y un elemento mínimo.

4.2.9. Formas de representar relaciones binarias

Una relación binaria definida en un conjunto finito A puede ser representada de varias maneras:

por extensión: listando los pares que la forman.

como una matriz 0-1: una matriz M cuyas filas y columnas están indexadas por los elementos de A, y donde

$$M_{xy} = \begin{cases} 1 & \text{si } xRy, \\ 0 & \text{si } x \not R y. \end{cases}$$

como un *grafo dirigido*: donde los elementos de A son los *vértices* y, para cada par $(x, y) \in R$, se tiene una *arista* que va desde x a y.

4.2.10. Ejemplo

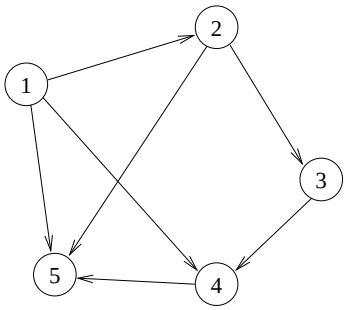
Sea $A = \{1, 2, 3, 4, 5\}$, y sea $R \subseteq A \times A$ dada por

$$R = \{(1,2), (1,4), (1,5), (2,3), (2,5), (3,4), (4,5)\}\,.$$

Esta relación puede ser representada por la matriz

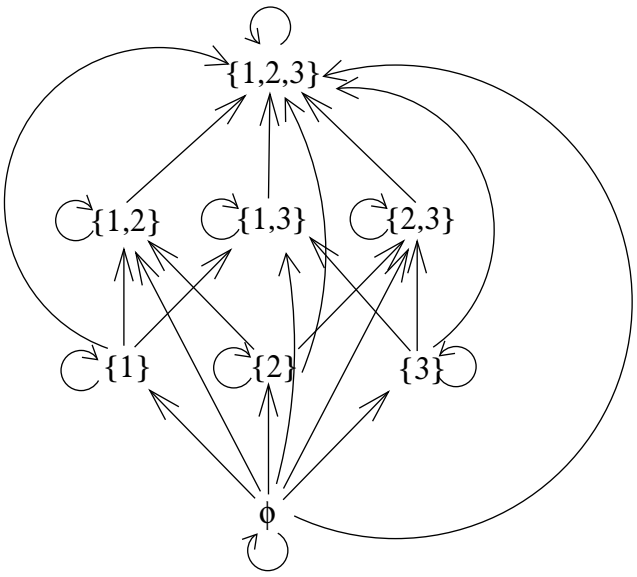
$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

o bien por el grafo dirigido



4.2.11. Diagramas de Hasse

Consideremos el orden $(\mathcal{P}(\{1,2,3\}), \subseteq)$. Un grafo dirigido que representa este orden es:



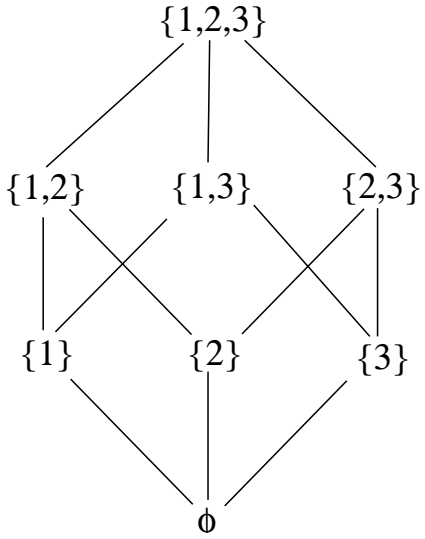
Muchas de las aristas del grafo anterior pueden ser deducidas de otras, usando el hecho de que la relación es un orden parcial.

Un orden parcial puede ser representado en forma gráfica de una forma simplificada, tomando el grafo dirigido que lo representa (como relación), y haciéndole los siguientes cambios:

- eliminar las aristas que pueden ser deducidas de otras por transitividad;
- eliminar los lazos (se sabe que todos los posibles lazos están, por reflexividad);

- ubicar los vértices de modo que todas las flechas vayan "hacia arriba", y eliminar las flechas.

La figura resultantes es llamada un *diagrama de Hasse*.
Por ejemplo, para el ejemplo anterior, el diagrama de Hasse sería:



4.2.12. Reticulados (lattices)

Definición 21. Un orden parcial $(A, \preceq)$ es un *reticulado* si todo subconjunto de A de cardinalidad 2 tiene un supremo y un í nfimo.

Ejemplos:

- $(\mathcal{P}(\mathcal{U}), \subseteq)$ es un reticulado.
- $(\mathbb{N} - \{0\}, |)$ es un reticulado.

Ejercicio. Dé definiciones explícitas de $\sup\{x, y\}$ e $\text{ínf}\{x, y\}$ para los ejemplos anteriores.

A futuro veremos otros ejemplos.

4.3. Relaciones de equivalencia

Definición 22. Sea A un conjunto. Una *relación de equivalencia* en A es una relación $\sim$ definida en A que es:

1. refleja en A,
2. simétrica, y
3. transitiva.

4.3.1. Ejemplos

- La igualdad es siempre una relación de equivalencia, en cualquier conjunto A.
- Si $f : A \to B$ es una función cualquiera, entonces la relación $\sim$ definida en A por $x \sim y \leftrightarrow f(x) = f(y)$ es una relación de equivalencia.
- Sea $n \in \mathbb{N}$, $n > 0$. La relación $\equiv_n$, definida en $\mathbb{Z}$ por $x \equiv_n y \leftrightarrow n \mid (x - y)$, es una relación de equivalencia.

- La relación $\uparrow$ definida en $\mathbb{Z} \times (\mathbb{N} - \{0\})$ por

$$(a, b) \uparrow (c, d) \leftrightarrow ad = bc$$

  es una relación de equivalencia.

- La relación "ser trasladado paralelo" entre trazos dirigidos (en el plano o el espacio) es una relación de equivalencia.

Ejercicio. Demuestre las afirmaciones contenidas en los ejemplos anteriores.

4.3.2. Clases de equivalencia

Sea $\sim$ una relación de equivalencia definida en A. Para cada $a \in A$, consideremos el conjunto

$$[a]_{\sim} = \{x \in A : x \sim a\}.$$

Llamamos al conjunto $[a]_{\sim}$ la *clase de equivalencia de a por* $\sim$. Cuando la relación sea clara del contexto, omitiremos el subíndice que la menciona.

El conjunto

$$A/\sim = \{[a]_{\sim} : a \in A\}$$

es llamado el *conjunto cuociente de A por* $\sim$.

4.3.3. Propiedades de las clases de equivalencia

Las clases de equivalencia de A por $\sim$ satisfacen lo siguiente:

- cada una de ellas es no vacía;
- dos cualesquiera distintas de ellas son disjuntas;
- la unión de todas ellas es A.

Ejercicio. Demuestre lo anterior.

4.3.4. Particiones

Definición 23. Dado un conjunto A, un conjunto $\Pi \subseteq \mathcal{P}(A)$ es una *partición de A* si satisface lo siguiente:

- $\forall S \in \Pi(S \neq \emptyset)$,
- $\forall S, T \in \Pi(S \neq T \rightarrow S \cap T = \emptyset)$,
- $\bigcup \Pi = A$.

Así, hemos probado que, si $\sim$ es una relación de equivalencia en A, entonces $A/\sim$ es una partición de A.

Viceversa: si Π es una partición de A, es posible definir una relación de equivalencia $\sim$ en A tal que $A/\sim = \Pi$. ¿Cómo?

4.3.5. Definiendo nuevos objetos con relaciones de equivalencia

Uno de los mayores usos de las relaciones de equivalencia es la definición de nuevos objetos, como conjunto cuociente de otro por una relación de equivalencia.

Ejemplo. Formalmente, los enteros son definidos como el conjunto cuociente de $\mathbb{N} \times \mathbb{N}$ por la relación

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

Ejercicio. Defina formalmente los racionales a partir de una relación de equivalencia en $\mathbb{Z} \times (\mathbb{N} - \{0\})$.

Ejercicio. Defina formalmente los números reales a partir de una relación de equivalencia entre las sucesiones de Cauchy de racionales.

4.3.6. Ejemplo: los enteros módulo n

Consideremos la relación $\equiv_n$, definida en $\mathbb{Z}$ por $x \equiv_n y \leftrightarrow n \mid (x - y)$. Llamamos *conjunto de los enteros módulo n* al conjunto

$$\mathbb{Z}_n = \{[i] : i \in \mathbb{Z}\}$$

(aquí estamos usando la convención de que si la relación de equivalencia es clara del contexto no la mencionamos explícitamente).

Es fácil ver que

$$\mathbb{Z}_n = \{[0], [1], \ldots, [n-1]\}.$$

Los enteros tienen una estructura dada por dos operaciones, $+$ y $\cdot$. ¿Será posible que $\mathbb{Z}_n$ "*herede*" esta estructura?

4.3.7. Operaciones en $\mathbb{Z}_n$

La manera "natural" de definir una suma y un producto en $\mathbb{Z}_n$ es como sigue:

$$\begin{aligned} [i] + [j] &= [i + j], \\ [i] \cdot [j] &= [i \cdot j], \end{aligned}$$

¿Cuál es el (posible) problema con esta definición?

4.3.8. Independencia de los representantes

El cuidado que hay que tener al definir operaciones o funciones en términos de miembros de clases de equivalencia de $A/\sim$ es que dicha definición sea *independiente de los representantes*.

En otras palabras, si por ejemplo definimos $f([a])$ en términos de a, debemos cerciorarnos de que, si en lugar de a elegimos $b \sim a$ como representante de $[a]$ (ya que $b \sim a \rightarrow [a] = [b]$), obtengamos el mismo resultado.

Supongamos que definimos $f([a])$ como $f([a]) = g(a)$. Entonces debe tenerse:

$$b \sim a \rightarrow g(b) = f([b]) = f([a]) = g(a).$$

Por ejemplo, en el caso de $\mathbb{Z}_n$ al definir la suma, tendrí amos que demostrar que, si $a \equiv_n c$ y $b \equiv_n d$, entonces $[a + b] = [c + d]$.

4.3.9. Otros objetos definidos por relaciones de equivalencia

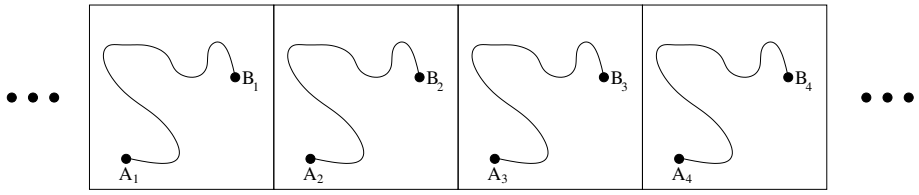
El cilindro

Consideremos la relación definida en $\mathbb{R} \times [0,1]$ como

$$(x,y)R_1(x',y') \leftrightarrow x - x' \in \mathbb{Z} \wedge y = y'.$$

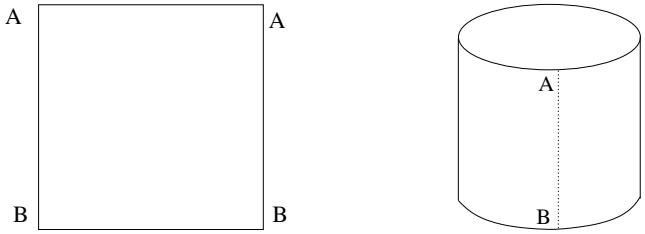
Ejercicio. Demuestre que ésta es una relación de equivalencia.

Una hormiga que viva en $\mathbb{R} \times [0,1]$ pero que "perciba" los puntos como equivalentes de acuerdo a esta relación no pensará que el mundo es "un plano", sino que considerará que cada uno de los cuadrados de la figura siguiente es "el mismo".



Así, la hormiga creerá que es lo mismo ir de A_1 a B_1 que de A_2 a B_2, de A_3 a B_3, etc.

Si nuestra hormiga hiciera un mapa de "el mundo",encontraría que los puntos marcados A en la figura de la izquierda serían "equivalentes (y lo mismo pasa con los puntos marcados B).



Identificando estos puntos (y los otros puntos equivalentes en los bordes del cuadrado), vemos que el mundo para esta hormiga tiene una apariencia similar a la de la figura de la derecha.

En otras palabras, hemos definido un *cilindro* como el conjunto cuociente de $\mathbb{R} \times [0,1]$ por la relación de equivalencia indicada más arriba (o, si se quiere, como el conjunto cuociente del cuadrado unitario $[0,1] \times [0,1]$ por la relación que iguala a los puntos correspondientes de los extremos derecho e izquierdo del cuadrado.

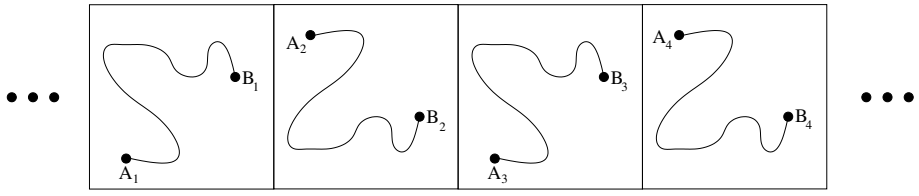
La cinta de Möbius

Supongamos que cambiamos la relación de equivalencia por la siguiente:

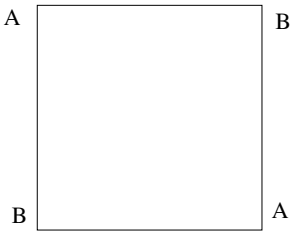
$$(x,y)R_2(x',y') \leftrightarrow x - x' \in \mathbb{Z} \wedge \begin{cases} y = y' & \text{si } x - x' \text{ es par,} \\ y + y' = 1 & \text{si } x - x' \text{ es impar.} \end{cases}.$$

Ejercicio. Demuestre que ésta es una relación de equivalencia.

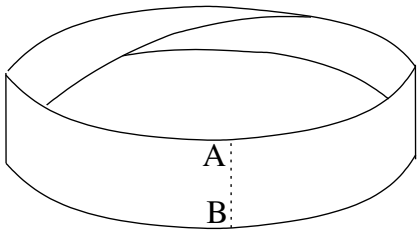
Así, la hormiga vería $\mathbb{R} \times [0,1]$ así:



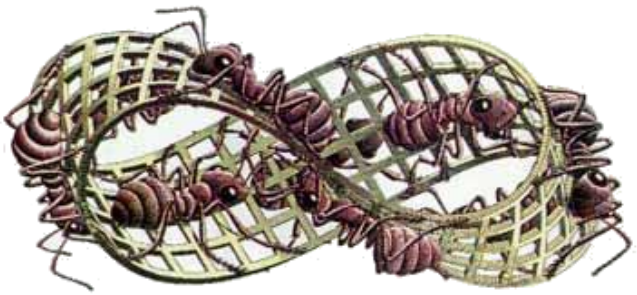
La hormiga del ejemplo considerará que los bordes derecho e izquierdo de cada uno de estos cuadrados son equivalentes, pero con los puntos "dados vuelta" (ver figura).



Así, al hacer el mapa, la hormiga descubrirá que el mundo tiene la forma de una *cinta de Möbius*:

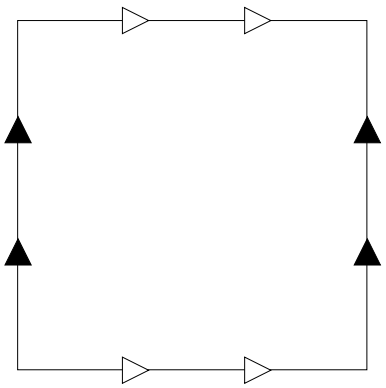


Quizás la cinta de Möbius más conocida es el "desfile de hormigas" pintado por M.C. Escher:

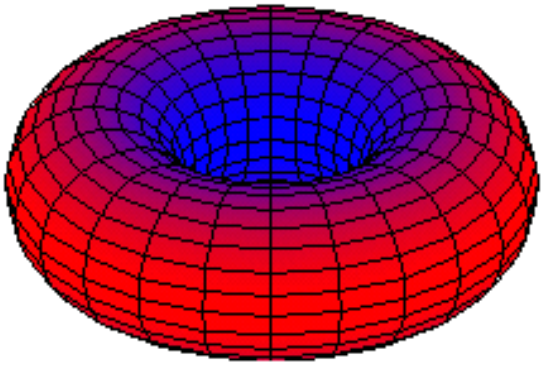


El toro

También conocido como "la dona" o "el Michelin", el toro se obtiene al tomar el cuociente de $[0,1]\times[0,1]$ por la relación que hace equivalente a cada punto del extremo izquierdo con el correspondiente punto del extremo derecho, y a cada punto del extremo superior con el correspondiente punto del extremo inferior (las flechas indican qué puntos se identifican con cuáles):

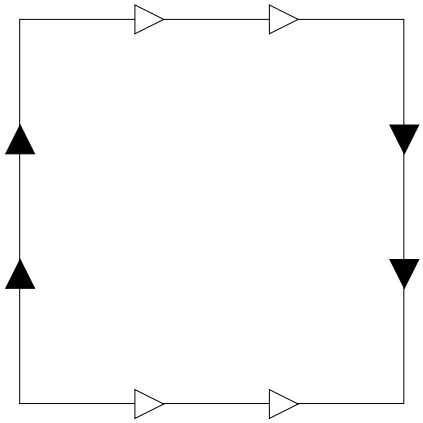


Una imagen del toro es



La botella de Klein

Otro ejemplo de superficie que se puede definir usando relaciones de equivalencia es la *botella de Klein*: para ello tomamos el cuociente de $[0,1] \times [0,1]$ por la relación que hace equivalente a cada punto del extremo superior con el correspondiente punto del extremo inferior, y a cada punto del extremo izquierdo con el punto *opuesto* del extremo derecho (las flechas indican qué puntos se identifican con cuáles):



Lamentablemente, en $\mathbb{R}^3$ es imposible "realizar" una botella de Klein (o sea, es imposible conseguir una botella de Klein sin que se "cruce" consigo misma).

Una imagen de la botella de Klein (con un cruce) es:

4.3.10. El volumen de la botella de Klein

Así como la cinta de Möbius tiene "un solo lado", la botella de Klein no tiene "interior y exterior". Si designamos una de las superficies como "adentro", es posible moverse sobre dicha superficie y llegar a estar "afuera".

Así, el interior de la botella tiene volumen 0, por lo que una graduación adecuada de su volumen está dado por la siguiente foto:

4.4. Ejercicios

1. Sea $A \neq \emptyset$, y sea $R \subseteq A \times A$ el conjunto vacío. ¿Cuáles de las siguientes propiedades tiene R?

   a) refleja,

   b) simétrica,

   c) transitiva,

   d) antisimétrica.

2. Dada una relación R definida en un conjunto A, definimos

$$R^{-1} = \{(x, y) \in A \times A : (y, x) \in R\}.$$

   Demuestre que R es simétrica sii $R = R^{-1}$.

3. Dada una relación R definida en un conjunto A, definimos la *clausura simétrica* de R como

$$\bar{R} = \bigcap \{S : R \subseteq S \subseteq A \times A \wedge S \text{ es simétrica}\}.$$

   Demuestre que $\bar{R} = R \cup R^{-1}$. Deduzca que $\bar{R}$ es simétrica.

4. Dadas dos relaciones $R \subseteq A \times B$ y $S \subseteq B \times C$, definimos la *composición* $S \circ R$ de R y S como

$$S \circ R = \{(x, z) \in A \times C : \exists y \in B(xRy \wedge ySz)\}.$$

   Demuestre que $R \subseteq A \times A$ es transitiva sii $R \circ R \subseteq R$.

5. Dada una relación R definida en un conjunto A, definimos la *clausura transitiva* de R como

$$R^+ = \bigcap \{S : R \subseteq S \subseteq A \times A \wedge S \text{ es transitiva}\}.$$

Demuestre que $R^+ = \bigcup_{n=1}^{\infty} R^n$, donde R^n está definido inductivamente por

$$\begin{aligned} R^1 &= R, \\ R^{n+1} &= R^n \circ R. \end{aligned}$$

Deduzca que R^+ es transitiva.

6. Sea $f : A \to B$ una función. Demuestre que la siguiente relación R es de equivalencia en A:

$$(x, y) \in R \leftrightarrow f(x) = f(y).$$

7. Sean R_1 y R_2 dos relaciones de equivalencia en A. Demuestre que $R_1 \cap R_2$ es relación de equivalencia en A. ¿Es necesariamente $R_1 \cup R_2$ una relación de equivalencia? Justifique.

8. Generalizando el problema anterior, sea $\{R_i : i \in I\}$ una familia (posiblemente infinita) de relaciones de equivalencia definidas en un conjunto A. Demuestre que

$$R = \bigcap_{i \in I} R_i$$

también es relación de equivalencia en A.

9. Demuestre que las siguientes relaciones definidas en $\mathbb{R}^2 \times \mathbb{R}^2$ son de equivalencia:

   - $(x, y)R_1(x', y') \leftrightarrow x - x' \in \mathbb{Z}$.
   - $(x, y)R_1(x', y') \leftrightarrow x - x' \in \mathbb{Z} \wedge y = y'$.

   En clases vimos (informalmente) que es posible definir un cilindro como el cuociente del conjunto $\mathbb{R} \times [0, 1]$ por la relación de equivalencia

$$(x, y)R_1(x', y') \leftrightarrow x - x' \in \mathbb{Z} \wedge y = y',$$

y que es posible definir la cinta de Möbius como el cuociente del mismo conjunto por la relación

$$(x, y)R_2(x', y') \leftrightarrow x - x' \in \mathbb{Z} \wedge \begin{cases} y = y' & \text{si } x - x' \text{ es par,} \\ y + y' = 1 & \text{si } x - x' \text{ es impar.} \end{cases}$$

Demuestre que las relaciones R_1 y R_2 definidas en el párrafo anterior, así como las relaciones R_3, R_4 y R_5 definidas en $\mathbb{R}^2$ que se dan a continuación son efectivamente relaciones de equivalencia.

$$\begin{aligned} (x, y)R_3(x', y') &\leftrightarrow x - x' \in \mathbb{Z} \wedge y - y' \in \mathbb{Z}; \\ (x, y)R_4(x', y') &\leftrightarrow x - x' \in \mathbb{Z} \wedge \begin{cases} y - y' \in \mathbb{Z} & \text{si } x - x' \text{ es par,} \\ y + y' \in \mathbb{Z} & \text{si } x - x' \text{ es impar;} \end{cases} \\ (x, y)R_5(x', y') &\leftrightarrow (x - x' \in \mathbb{P} \wedge y - y' \in \mathbb{P}) \\ &\quad \vee (x + x' \in \mathbb{P} \wedge y + y' \in \mathbb{P}) \\ &\quad \vee (x - x' \in \mathbb{I} \wedge y + y' \in \mathbb{I}) \\ &\quad \vee (x + x' \in \mathbb{I} \wedge y - y' \in \mathbb{I}) \\ &\quad \text{(donde } \mathbb{P} \text{ y } \mathbb{I} \text{ representan los conjuntos de enteros pares e} \\ &\quad \text{impares respectivamente).} \end{aligned}$$

10. Considere la siguiente relación definida en $\mathbb{N}^2$:

$$(m,n) \sim (m',n') \leftrightarrow m+n' = m'+n.$$

a) Demuestre que la relación $\sim$ así definida es de equivalencia.

b) Liste algunos elementos de las clases de equivalencia $[(0,3)]$, $[(4,1)]$, $[(5,0)]$, $[(2,7)]$.

c) Formalmente, $\mathbb{Z} = \mathbb{N}^2/\sim$. Intuitivamente, ¿a qué entero corresponde la clase $[(m,n)]$?

d) Demuestre que la operación entre clases de equivalencia dada por $[(m,n)] \oplus [(p,q)] = [(m+p, n+q)]$ está bien definida (en otras palabras, esta definición no depende de los representantes escogidos: si $(m',n') \in [(m,n)]$ y $(p',q') \in [(p,q)]$ entonces $[(m'+p', n'+q')] = [(m+p, n+q)]$).

e) Intuitivamente, la operación $\oplus$ corresponde a la suma de números enteros. Defina una operación $\otimes$ que corresponda al producto de números enteros.

11. Así como en el ejercicio anterior definimos formalmente $\mathbb{Z}$, en este ejercicio queremos definir $\mathbb{Q}$. Para ello, consideramos la siguiente relación definida en $\mathbb{Z} \times (\mathbb{N} - \{0\})$:

$$(k,n) \sim (k',n') \leftrightarrow kn' = k'n.$$

a) Demuestre que la relación $\sim$ así definida es de equivalencia.

b) Liste algunos elementos de las clases de equivalencia $[(0,2)]$, $[(3,1)]$, $[(2,3)]$.

c) Formalmente, $\mathbb{Q} = (\mathbb{Z} \times (\mathbb{N} - \{0\})/\sim$. Intuitivamente, ¿a qué número racional corresponde la clase $[(k,n)]$?

d) Demuestre que la operación entre clases de equivalencia dada por $[(k,n)] \otimes [(l,p)] = [(kl, np)]$ está bien definida (en otras palabras, esta definición no depende de los representantes escogidos: si $(k',n') \in [(k,n)]$ y $(l',p') \in [(l,p)]$ entonces $[(k'l', n'p')] = [(kl, np)]$).

e) Intuitivamente, la operación $\otimes$ corresponde a la multiplicación de números racionales. Defina una operación $\oplus$ que corresponda a la suma de números racionales.

12. Sea $n \in \mathbb{N}$. En $\mathbb{Z}$, definimos la relación $\equiv \pmod{n}$ como:

$$x \equiv y \pmod{n} \leftrightarrow n \text{ divide a } x-y.$$

a) Demuestre que $\equiv \pmod{n}$ es una relación de equivalencia.

b) El conjunto $\mathbb{Z}_n$ (los enteros módulo n) es definido como $\mathbb{Z}/\cong \pmod{n}$. Encuentre [16], [22], [28] y [39] en $\mathbb{Z}_7$.

c) Definimos la operación $+$ (suma módulo n) en $\mathbb{Z}_n$, como sigue:

$$[k] + [l] = [k+l].$$

Demuestre que esta operación está bien definida, es decir, es independiente de los representantes considerados.

d) Defina una operación $\cdot$ (multiplicación *módulo* n) en $\mathbb{Z}_n$. Demuestre que esta operación está bien definida.

e) Queremos definir la operación $[k] \star [l]$ en $\mathbb{Z}_n$, como sigue:

$$[k] \star [l] = \left[\left\lfloor \frac{k+l}{2} \right\rfloor \right].$$

Aquí, $\lfloor x \rfloor$ representa la parte entera de x (o sea, el mayor entero que no excede x). Demuestre que esta operación no está bien definida, es decir, si $x, y \in \mathbb{Z}_n$ entonces el valor de $x \star y$ depende de los representantes considerados.

Capítulo 5

Funciones

5.1. Definiciones básicas

Definición 24. Una relación $f \subseteq A \times B$ es llamada una *función de A en B* (en símbolos, $f : A \to B$) si , dado cualquier $x \in A$, existe un único $y \in B$ tal que $(x, y) \in f$.

Notación. Dado $x \in A$, denotamos por $f(x)$ al único $y \in B$ tal que $(x, y) \in f$.

5.1.1. Tipos de funciones

Una función $f : A \to B$ es:

- *Inyectiva* (o 1-1) si, dados $x, y \in A$,

$$f(x) = f(y) \to x = y.$$

- *Epiyectiva* (o sobreyectiva, o simplemente *sobre*) si, dado cualquier $y \in B$, existe algún $x \in A$ tal que $y = f(x)$.

- *Biyectiva* (o biyección, o *correspondencia biunívoca*) si es inyectiva y epiyectiva.

Estamos especialmente interesado en las biyecciones.

5.2. Cardinalidad

Intuitivamente, *contar* los elementos de un conjunto A significa poner en correspondencia el conjunto A con los elementos de un conjunto de referencia.

Formalmente, diremos lo siguiente:

Definición 25. Dados dos conjuntos A y B, éstos *tienen la misma cantidad de elementos* si existe una biyección $f : A \to B$.

En este caso también se dice que A y B son *equinumerosos*, o *equipotentes*, lo que denotaremos por $A \sim B$.

Teorema. La relación $\sim$ es una relación de equivalencia.

Definición 26. Sea A un conjunto cualquiera. El *cardinal* de A (que anotamos $|A|$) es su clase de equivalencia por la relación de equinumerosidad.

Típicamente, usamos algún miembro de $|A|$ para denotar la clase entera.

5.2.1. Conjuntos finitos e infinitos

Recordemos que un elemento de $\mathbb{N}$ es de la forma

$$n = \{0, 1, 2, \dots, n-1\}.$$

Definición 27. Un conjunto se dice *finito* si es equinumeroso con algún $n \in \mathbb{N}$. En este caso, diremos que la *cardinalidad* de A es n: $|A| = n$.

Un conjunto que no es finito se dice (sorpresa) *infinito*.

5.2.2. Caracterizando los conjuntos finitos

Sea A un conjunto finito, y sea $f : A \to A$.
Es posible demostrar que:

- si f es 1-1 entonces es sobre.
- si f es sobre entonces es 1-1.

Más aún: si A es un conjunto cualquiera, tal que toda función 1-1 $f : A \to A$ es sobre (o que toda función sobre $f : A \to A$ es 1-1), entonces A es finito.

5.2.3. Conjunto numerables

Un conjunto equinumeroso con $\mathbb{N}$ se dice *numerable*.
Claramente, todo conjunto numerable es infinito. ¿Será verdad el recíproco?

5.2.4. Ejemplos de conjuntos numerables

Algunos de los conjuntos que pueden ser puestos en correspondencia biunívoca con $\mathbb{N}$:

- $\mathbb{P}$, el conjunto de los naturales *pares*:

$$\mathbb{P} = \{2n : n \in \mathbb{N}\}.$$

- $\mathbb{I}$, el conjunto de los naturales *impares*:

$$\mathbb{P} = \{2n + 1 : n \in \mathbb{N}\}.$$

- Π, el conjunto de los números primos:

$$\Pi = \{2, 3, 5, 7, 11, 13, 17, 19, 23, \dots\}.$$

Estos ejemplos pueden aparecer anti-intuitivos a primera vista, ya que estos conjuntos son todos *subconjuntos propios* de $\mathbb{N}$. Esperaríamos que tuvieran cardinalidad "menor" que la de $\mathbb{N}$.

5.3. Caracterizaciones de numerabilidad

Una manera de caracterizar los conjuntos numerables es la siguiente:

> Un conjunto A es numerable si y sólo si es posible escribir A como una *lista infinita*:
>
> $$A = \{a_1, a_2, a_3, \dots\}.$$

En otras palabras, existe una *sucesión*

$$(a_1, a_2, a_3, \dots)$$

de elementos de A con la propiedad de que *todo elemento de A aparece en algún momento en dicha sucesión*.

En términos algorítmicos, podemos pensar esto como que existe un "algoritmo"[1] que genera un elemento de A, luego otro, y así sucesivamente, de modo que *todo elemento de A es generado en algún momento por este algoritmo*.

[1] Aunque este "algoritmo" nunca termina ...

5.4. Los racionales

¿Es $\mathbb{Q}$ numerable?

Argumento (intuitivo) en contra:

Parece haber "demasiados" racionales más que naturales. ¡Entre dos naturales consecutivos siempre hay una cantidad infinita de racionales!

Sin embargo, es posible hacer una lista (infinita) en que cada racional aparezca (exactamente) una vez:

$$\mathbb{Q} = \left\{ \frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{3}{1}, \frac{1}{4}, \frac{2}{3}, \frac{3}{2}, \frac{4}{1}, \dots \right\}.$$

O sea, $\mathbb{Q}$ es numerable.

5.5. Los reales

¿Será verdad que $\mathbb{R}$ es numerable?

Se puede demostrar que ni siquiera el intervalo cerrado $[0,1]$ es numerable ...

Demostración. Supongamos que $[0,1]$ es numerable. Entonces, es posible escribir

$$[0,1] = \{r_0, r_1, r_2, \dots\}$$

donde:

$$\begin{aligned} r_0 &= 0.c_{00}c_{01}c_{02}c_{03}c_{04}\cdots \\ r_1 &= 0.c_{10}c_{11}c_{12}c_{13}c_{14}\cdots \\ r_2 &= 0.c_{20}c_{21}c_{22}c_{23}c_{24}\cdots \\ r_3 &= 0.c_{30}c_{31}c_{32}c_{33}c_{34}\cdots \\ &\vdots \end{aligned}$$

donde $c_{ij} \in \{0, \dots, 9\}$.

Pero entonces el real

$$r := 0.d_0d_1d_2d_3d_4\cdots$$

donde

$$d_i := (c_{ii} + 5) \mod 10$$

no aparece en la lista ...

□

Ejercicio. ¿Dónde falla el argumento aquí presentado al tratar de demostrar —de la misma manera— que $\mathbb{Q} \cap [0,1]$ no es numerable?

5.6. El argumento de Cantor

El argumento presentado antes es llamado la *diagonalización de Cantor* o *argumento diagonal de Cantor*. El teorema presentado es un caso particular del siguiente:

Teorema (Cantor). *Sea A un conjunto cualquiera. Entonces no hay ninguna biyección entre A y $\mathcal{P}(A)$.*

5.6.1. El problema de la detención

El argumento diagonal de Cantor tiene aplicaciones a Ciencia de la Computación. Por ejemplo, se le usa para probar que es imposible escribir un programa cuya entrada sea un par (P, A) (donde P es un programa y A un archivo de datos) y decida si el programa P se detendrá al ser ejecutado con entrada A.

5.7. Orden entre cardinalidades

El hecho de que no hay una biyección entre $\mathbb{N}$ y el intervalo $[0,1]$, o entre A y $\mathcal{P}(A)$ (y sin embargo sí hay una función 1-1 $f : \mathbb{N} \to [0,1]$, y una de A en $\mathcal{P}(A)$) nos sugiere la siguiente definición:

Definición 28. Dados dos conjuntos A y B, decimos que B tiene *al menos tantos elementos como* A (o que A *no tiene más elementos que* B) si existe una función 1-1 $f : A \to B$.

Si este es el caso, anotamos $A \preceq B$.

5.7.1. Propiedades de $\preceq$

La relación $\preceq$ tiene las siguientes propiedades:

1. es refleja: $A \preceq A$;
2. es transitiva: si $A \preceq B$ y $B \preceq C$ entonces $A \preceq C$;
3. es "casi" antisimétrica: si $A \preceq B$ y $B \preceq A$ entonces $A \sim B$ (teorema de Cantor-Schröder-Bernstein (CSB)).

En realidad, $\preceq$ no es un orden parcial pero determina un orden parcial en el conjunto[2] de clases de equivalencia por la relación $\sim$.

5.8. El teorema de Cantor-Schröder-Bernstein (CSB)

Enunciamos el

Teorema (Cantor-Schröder-Bernstein). *Si $A \preceq B$ y $B \preceq A$ entonces $A \sim B$.*

O sea: si existe una función inyectiva $g : A \to B$ y una función inyectiva $h : B \to A$, entonces existe una biyección $f : A \to B$.

Antes de ver su demostración, estudiaremos un caso particular del teorema.

5.8.1. Prolegómeno:

Demostraremos primero que $[0,1] \sim [0,1)$.

Para hacer esto, la primera (estúpida) idea que se nos ocurre es considerar la "función" $f_0(x) = x$ definida en $[0,1]$.

¿Qué problema tenemos?

Como $f_0(1)$ ni siquiera está en el conjunto de llegada, debemos asignarle a 1 una imagen en $[0,1)$. Digamos por ejemplo que a 1 le asignamos como imagen $1/2$. Así, definimos $f_1 : [0,1] \to [0,1)$ como sigue:

$$f_1(x) = \begin{cases} x & \text{si } x \neq 1, \\ 1/2 & \text{si } x = 1. \end{cases}$$

Ahora tenemos que $1/2$ tiene dos pre-imágenes: 1 y $1/2$. Ya que el 1 acaba de llegar a $1/2$, está cansado, por lo que el que tiene que reubicarse es $1/2$.

Digamos, por ejemplo, que $1/2$ se va a $1/4$ (podría ser $1/3$, o cualquier otro $\neq 1/2$). Así, definimos $f_2 : [0,1] \to [0,1)$ como sigue:

$$f_2(x) = \begin{cases} x & \text{si } x \neq 1, x \neq 1/2, \\ 1/2 & \text{si } x = 1, \\ 1/4 & \text{si } x = 1/2. \end{cases}$$

Ahora tenemos un problema con $1/4$ ... ¿Qué hacemos?

[2] ¿La clase propia?

Ahora mandemos 1/4 a 1/8. Así, definimos $f_3 : [0,1] \to [0,1)$ como sigue:

$$f_2(x) = \begin{cases} x & \text{si } x \notin \left\{1, \frac{1}{2}, \frac{1}{4}\right\}, \\ 1/2 & \text{si } x = 1, \\ 1/4 & \text{si } x = 1/2, \\ 1/8 & \text{si } x = 1/4. \end{cases}$$

Y ahora el problema se produce en 1/8 ... ¿Qué hemos ganado?

Hemos ganado mucho: si consideramos la función

$$f(x) = \lim_{n\to\infty} f_n(x) = \begin{cases} x & \text{si } x \notin \left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots\right\}, \\ x/2 & \text{si } x \in \left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots\right\}, \end{cases}$$

entonces es posible demostrar que, efectivamente, f es una biyección entre $[0,1]$ y $[0,1)$.

Demostración. Ejercicio. □

5.8.2. Demostración de C-S-B

Retomemos la demostración del Teorema de Cantor, Schröder y Bernstein.

Sean A y B dos conjuntos tales que hay dos funciones inyectivas, $g : A \to B$ y $h : B \to A$.

Sea $B' = h(B) \subseteq A$. Claramente, B y B' son equinumerosos ($h : B \to B'$ es una biyección). Así, si logramos establecer una biyección entre A y B' tendremos la equinumerosidad deseada entre A y B.

Claramente, $h \circ g : A \to B'$ es inyectiva.

Así, nos bastará con demostrar la siguiente versión simplificada del teorema:

> *Dados un conjunto A y un subconjunto A' de A, si existe una función inyectiva $j : A \to A'$, entonces A y A' son equinumerosos.*

En el caso que demostramos anteriormente, $A = [0,1]$, $A' = [0,1)$, $j : [0,1] \to [0,1)$ está dada por $j(x) = x/2$.

La manera de construir la biyección entre A y A' recuerda a la de la demostración anterior.

Primera (estúpida) idea:

$$f_0 : A \to A'$$

dada por $f_0(x) = x$ no resulta (se escapa de A').

5.8.3. Solución (temporal) del problema

Definimos $f_1 : A \to A'$ como sigue:

$$f_1(x) = \begin{cases} x & \text{si } x \notin A - A', \\ j(x) & \text{si } x \in A - A'. \end{cases}$$

Ahora el problema lo tenemos con los elementos de $j(A - A')$ (lo que pasaba con 1/2 en el ejemplo).

Si $x \in j(A - A')$, en lugar de mandar x a x, los mandamos a $j(x)$. O sea:

$$f_2(x) = \begin{cases} x & \text{si } x \notin A - A' \cup j(A - A'), \\ j(x) & \text{si } x \in A - A' \cup j(A - A'). \end{cases}$$

Se ve que vamos a tener problemas con $j(j(A - A'))$, $j(j(j(A - A')))$, etc.

5.8.4. Solución final

Consideremos los conjuntos $C_0 = A - A'$, $C_1 = j(C_0)$, $C_2 = j(C_1)$, $C_3 = j(C_2)$, etc. Claramente, necesitamos aplicar j a los elementos de estos conjuntos. Así, sea

$$C = \bigcup_{i=0}^{\infty} C_i.$$

Definimos

$$f(x) = \begin{cases} x & \text{si } x \notin C, \\ j(x) & \text{si } x \in C. \end{cases}$$

Ejercicio. Demuestre que la función $f : A \to A'$ recién definida es efectivamente una biyección (tarea, 2'2002).

5.9. Ejercicios

1. Demuestre que si A es un conjunto finito y $B \subseteq A$ entonces B es finito.
2. Demuestre que si A es un conjunto finito entonces toda función $f : A \to A$ es 1-1 si y sólo si es sobre.
3. Demuestre que un conjunto A es infinito si y sólo si A es equinumeroso con algún subconjunto propio $A' \subset A$.
4. Demuestre que el conjunto $\mathbb{N} \times \mathbb{N}$ es numerable. Construya explícitamente una biyección

   $$f : \mathbb{N} \to \mathbb{N} \times \mathbb{N}.$$

   Determine el valor de $f(17)$ y de $f^{-1}(5, 6)$.
5. Dé biyecciones explícitas entre:

   a) El conjunto de los naturales pares y el de los enteros impares.

   b) El conjunto de los naturales pares y el de los enteros.

   c) El conjunto de las potencias de 2 y el de los números enteros pares.
6. Demuestre que si dos conjuntos A y B son numerables entonces $A \cup B$ es numerable.

   Ayuda: ¡Cuidado con los elementos comunes a A y a B!
7. Demuestre que si dos conjuntos A y B son finitos entonces $A \cup B$ y $A \cap B$ son conjuntos finitos.
8. Demuestre que si A es un conjunto finito y B es un conjunto numerable entonces $A \cup B$ es numerable y $A \cap B$ es finito.
9. Demuestre que si $A \subseteq \mathbb{N}$ entonces

   $$A \text{ es finito } \leftrightarrow A \text{ tiene un máximo.}$$

10. Dé ejemplos de dos conjuntos infinitos no numerables cuya intersección sea:

    a) finita,

    b) infinita numerable,

    c) infinita no numerable.
11. Sean A, B y C conjuntos tales que $C \subseteq A$, $A \cap B = \emptyset$ y $B \cong C$. Demuestre que $A \cup B \cong A$.

12. Demuestre que el conjunto de todas las sucesiones de números enteros no es numerable.

13. Demuestre que el conjunto de todas las sucesiones *finitas* de números racionales es numerable.

14. Una sucesión $(a_0, a_1, a_2, \dots)$ de enteros se dice *eventualmente periódica* si existen dos números naturales n_0 y $p > 0$ tales que, para todo $n \geq n_0$, $a_{n+p} = a_n$.

    Demuestre que el conjunto de todas las sucesiones eventualmente periódicas de enteros es numerable.

15. Una sucesión $(a_0, a_1, \dots)$ de enteros se dice *progresión aritmética* si para todo $n \in \mathbb{N}$ se tiene $a_{n+2} - a_{n+1} = a_{n+1} - a_n$.

    Demuestre que el conjunto de todas las progresiones aritméticas de enteros es numerable.

16. Un número real se dice *algebraico* si es raíz de algún polinomio con coeficientes enteros. Si un número real no es algebraico entonces es *trascendente*.

    a) Demuestre que el conjunto de todos los números reales algebraicos es numerable.

    b) Demuestre que el conjunto de todos los números reales trascendentes no es numerable.

17. Demuestre que el conjunto $\mathcal{P}(\mathbb{N})$ (el conjunto potencia de $\mathbb{N}$) tiene la misma cardinalidad que $\mathbb{R}$.

18. Demuestre que el conjunto de todas las secuencias finitas de enteros positivos es numerable.

    Ayuda: A la secuencia $(r_0, r_1, r_2, \dots, r_k)$ asóciele el número $2^{r_0}3^{r_1}\cdots p_k^{r_k}$, donde p_k es el k-ésimo primo (contando desde cero).

19. Demuestre que el conjunto de todas las secuencias finitas de racionales es numerable.

20. Demuestre que el conjunto de todas las rectas del plano que pasan por (al menos) dos puntos con coordenadas racionales, es numerable.

21. Demuestre que cualquier conjunto infinito de círculos disjuntos en el plano es numerable. ¿Por qué es importante aquí la hipótesis de "disjuntos"?

22. Demuestre que $\mathbb{N}$ puede ser escrito como la unión de una familia numerable de conjuntos numerables disjuntos.

23. Demuestre que si D es un conjunto numerable de puntos del plano cartesiano, entonces es posible escribir D como $D = D_x \cup D_y$, donde $D_x \cap \ell$ es finito para cada recta ℓ paralela al eje X, y $D_y \cap \ell$ es finito para cada recta ℓ paralela al eje Y.

24. Demuestre que en el conjunto $\mathcal{P}(\mathbb{N})$ (el *conjunto potencia* de los naturales) es posible encontrar una cadena NO NUMERABLE de subconjuntos, o sea, una familia no numerable

$$\mathcal{S} \subseteq \mathcal{P}(\mathbb{N})$$

    tal que $\forall A, B \in \mathcal{S}(A \subseteq B \vee B \subseteq A)$.

Capítulo 6

Inducción y clausuras

6.1. Inducción (sobre los naturales)

Usamos inducción sobre los naturales para:

- Demostrar que todos los números naturales tienen una cierta propiedad.

  Ejemplo típico: $\sum_{k=1}^{n} k \ = \frac{n(n+1)}{2}$.

- Definir diversos objetos asociados a los números naturales: definiciones inductivas/recursivas de: funciones, relaciones, etc.

  Ejemplo típico: definición de la función factorial para todo natural.

  $$0! = 1, \quad (n+1)! = (n+1) \cdot n!.$$

  Suponemos que el factorial está definido para n y lo definimos para $n+1$. La inducción nos permite demostrar que existe una única función $! : \mathbb{N} \to \mathbb{N}$ que satisface las ecuaciones de arriba.

6.1.1. Otros puntos de partida

Hemos adoptado la convención de que el primer número natural es cero.

¿Qué pasa si queremos demostrar que una propiedad se cumple a partir de $n = 17$? ¿o, incluso, a partir de $n = -13$?

En realidad, dado $n_o \in \mathbb{Z}$, podemos considerar, en lugar de $\mathbb{N}$, el conjunto $\{n \in \mathbb{Z} : n \geq n_o\}$, y demostrar la propiedad deseada (o definir el nuevo concepto) para n en dicho conjunto ($\mathbb{N}$ corresponde a la elección $n_o = 0$).

Los principios de inducción que veremos a continuación pueden ser adaptados a cualquier elección de n_o.

6.1.2. Principios de Inducción

Hay (al menos) tres principios de inducción para naturales que son *equivalentes*:

Principio Simple de Inducción (PSI)

Dado un subconjunto S de $\mathbb{N}$ ($S \subseteq \mathbb{N}$), si se cumple que:

(I) $0 \in S$;

(II) dado cualquier $n \in \mathbb{N}$, si $n \in S$ entonces $n + 1 \in S$;

entonces $S = \mathbb{N}$.

Al realizar demostraciones basadas en este principio le llamamos a la parte correspondiente a (i) "base" de la inducción, y a (ii) el "paso inductivo". Dentro del paso inductivo, la parte $n \in S$ recibe el nombre de "hipótesis de inducción" (HI) y la parte $n + 1 \in S$ recibe el nombre de "tesis de inducción" (TI).

Principio de Inducción "por curso de valores"

(también llamado "segundo principio de inducción" o "principio fuerte de inducción").

Dado un subconjunto S de $\mathbb{N}$ ($S \subseteq \mathbb{N}$), si se cumple que, para todo $n \in \mathbb{N}$:

$$\{k \in \mathbb{N} : k < n\} \subseteq S \to n \in S, \qquad (\star)$$

entonces $S = \mathbb{N}$.

Aquí vemos que no hay una "base" explícita: sólo hay un paso inductivo, y en éste la HI es $\{k \in \mathbb{N} : k < n\} \subseteq S$ y la TI es $n \in S$.

Principio del buen orden

Todo subconjunto no vacío, $S \neq \emptyset$, de $\mathbb{N}$, tiene un "primer elemento", es decir, existe $y \in S$, tal que, para todo $x \in S$, $y \leq x$.

Como dijimos al principio, todos estos pricipios son equivalentes: cada uno se puede demostrar a partir de cualquiera de los otros.

A veces, uno es más apropiado que otro, según el problema donde se quiera aplicar.

PICV es muy poderoso, la hipótesis es más facíl de usar, ya que lo que se quiere demostrar se supone para todos los predecesores de n, y no sólo para $n - 1$.

6.1.3. Ejercicios

Ejercicio. ¿Cómo puede usar el principio de inducción simple (y justificar el uso) para demostrar que todos los números naturales mayores que, digamos 10, tienen una cierta propiedad?

Ejercicio. Lo mismo que antes, pero ahora se desea demostrar que todos los números *enteros* mayores que, digamos -5, tienen una cierta propiedad.

Ejercicio. Lo mismo que en los dos ejercicios anteriores, pero ahora usando PICV.

Ejercicio. Demuestre la equivalencia entre los tres principios de inducción.
Ayuda: Encuentre una cadena cíclica de implicaciones.

6.1.4. Una formulación equivalente

Los dos primeros principios de inducción pueden ser formulados de manera equivalente usando predicados en lugar de conjuntos. Sea $P(n)$ un predicado con una variable n (entera o natural). Entonces podemos formular los dos primeros principios de inducción como sigue:

Principio Simple de Inducción (PSI)

Si se cumple que:

(I) $P(0)$; y

(II) para todo $n \in \mathbb{N}$, si se cumple $P(n)$ entonces se cumple $P(n + 1)$;

entonces $\forall n \in \mathbb{N}(P(n))$.

Principio de Inducción "por curso de valores"

Si, para todo $n \in \mathbb{N}$ se cumple que:

$$\forall k \in \mathbb{N}(k < n \to P(k)) \to P(n), \qquad (\star\star)$$

entonces $\forall n \in \mathbb{N}(P(n))$.

6.1.5. Casos base en en PICV

Nótese que, para usar PICV, hay que demostrar ($\star$) —o, equivalentemente, ($\star\star$)— *para todo* $n \in \mathbb{N}$.

Esto incluye al 0, por lo que debemos demostrar que:

$$\{k \in \mathbb{N} : k < 0\} \subseteq S \to 0 \in S.$$

Como la hipótesis es siempre verdadera, no aporta nada, e igual hay que demostrar que $0 \in S$, a partir de nada, tal como en PSI.

En otros casos, el hecho de que $\{k \in \mathbb{N} : k < n\} \subseteq S$ no aporta nada *porque en la demostración de que* $n \in S$ *no se ocupa la hipótesis*.

A estos casos los llamaremos *casos base* de las demostraciones por PICV (y, típicamente, 0 es un caso base de estas demostraciones, pero no es necesariamente el único).

Ejemplo. Demostremos que, si tenemos una cantidad infinita de estampillas de 4 y 7 pesos, podemos formar cualquier franqueo de 18 pesos o más.

Sea $S = \{n \in \mathbb{N} : n \geq 18 \to$ existen $x, y \in \mathbb{N}$ tales que $n = 4x + 7y\}$.

Los casos base en nuestra demostración corresponden a $0 \leq n < 18$ (que trivialmente pertenecen a S), $n = 18$, $n = 19$, $n = 20$ y $n = 21$. Por ejemplo, 21 es caso base porque la demostración de que $21 \in S$ no usa el hecho de que $\{0, 1, \ldots, 20\} \subseteq S$.

Así, en la demostración de que todo número natural n pertenece a S, hay 22 casos "base".

Ejemplo. Considérese la demostración de que todo número natural ≥ 2 tiene un factor primo.

Sea $S = \{n \in \mathbb{N} : n \geq 2 \to n$ tiene un factor primo $\}$.

Si n es compuesto, usamos la HI. Pero si n es primo, no la necesitamos para comprobar que $n \in S$.

Así, en la demostración de que todo número natural ≥ 2 tiene un factor primo, *hay una cantidad infinita de casos base*: todos los n primos, más 0 y 1 (que trivialmente pertenecen a S).

6.1.6. Aplicaciones de inducción en $\mathbb{N}$

En lo que sigue veremos aplicaciones no usuales en cursos básicos, pero importantes y útiles:

- Acotación de soluciones de ecuaciones de recurrencia. No sólo hay que demostrar que la solución (usualmente no disponible explícitamente) está acotada por una expresión algebraica que contiene constantes, sino que hay que demostrar en el proceso que tales constantes existen ("inducción constructiva").
- Demostración de Corrección y Término de programas computacionales. Se trata de demostrar que el programa para y que entrega en la salida el resultado esperado.
- Demostración de principios combinatorios (el Principio de los Cajones).

6.1.7. Ejercicios

Primer principio de inducción

1. Demuestre que para todo $n \in \mathbb{N}$, $7^n - 2^n$ es divisible por 5.
2. Demuestre que para todo $n \in \mathbb{N}$, $11^{n+1} + 2^{2n-1}$ es divisible por 133.
3. Demuestre que, si se tiene un conjunto de n líneas rectas en el plano, tal que entre ellas no hay dos paralelas ni tres concurrentes, entonces ellas dan lugar a $\frac{n^2+n+2}{2}$ regiones.
4. Considere la siguiente afirmación:

   "Podemos comer una cantidad infinita de comida."

   Analice el siguiente intento de demostración de esta afirmación, usando inducción:

- Podemos comer una pequeña porción de comida (caso base).
- Si podemos comer una cierta cantidad de comida, podemos comer un poco más (paso inductivo). Por inducción hemos demostrado que podemos comer una cantidad infinita de comida.

Segundo principio de inducción

5. Demuestre que, si se tiene un conjunto de n líneas rectas en el plano, tal que entre ellas no hay dos paralelas ni tres concurrentes, entonces entre las regiones a las que ellas dan lugar hay exactamente $n-2$ triángulos.

6. Demuestre que, si $\mathfrak{C}$ es un conjunto de n círculos en el plano, tal que cualesquiera 3 de dichos círculos tienen intersección no vacía, entonces la intersección de todos los círculos de $\mathfrak{C}$ es no vacía.

   Ayuda: Ocupe el hecho de que, dados dos puntos x, y en un un cí rculo C, el segmento $\overline{xy}$ está completamente contenido en C (o sea, C es convexo).

7. Suponga que $T(n)$ es una función que satisface:

$$\begin{aligned} T(1) &= 1, \\ T(n) &= 2T(\lfloor n/2 \rfloor) + 3T(\lceil n/2 \rceil) + 4 \quad \text{para } n > 1. \end{aligned}$$

   Demuestre que T es una función no decreciente, o sea, $T(n) \leq T(n+1)$ para todo $n \in \mathbb{N}$.

8. Suponga que $T(n)$ es una función que satisface:

$$\begin{aligned} T(0) &= 0, \\ T(n) &= T(\lfloor n/3 \rfloor) + T(\lceil n/5 \rceil) + T(\lceil n/7 \rceil) + n \quad \text{para } n > 0. \end{aligned}$$

   Demuestre que $T(n) < 4n$ para todo $n \in \mathbb{N}$.

9. El siguiente juego se juega entre dos jugadores con un montón de pasas (nueces, caramelos, almendras, M&M's también son admisibles). El juego comienza con el montón de pasas dividido en dos pilas (no necesariamente del mismo tamaño).

   Una *movida* consiste en comerse todas las pasas de una pila y dividir la otra en dos pilas no vacías, no necesariamente iguales.

   Los jugadores se van turnando en hacer sus movidas, y gana el último que puede hacer una movida.

   Demuestre que, si el número inicial de pasas en al menos una de las pilas es par, entonces el primer jugador puede ganar sin importar cómo juegue su adversario (o sea, tiene una *estrategia perfecta para ganar*).

10. Recuerde que los números de Fibonacci se definen como:

$$f_n = \begin{cases} 0 & \text{si } n = 0, \\ 1 & \text{si } n = 1, \\ f_{n-1} + f_{n-2} & \text{si } n > 1, \end{cases}$$

    Demuestre que, dado $n \in \mathbb{N}$, $f_n \leq (7/4)^{n-1}$.

Principio del menor entero

11. Demuestre que, dados $a \in \mathbb{Z}$ y $b \in \mathbb{N}$, $b > 0$, existe un único $q \in \mathbb{Z}$ y un único $r \in \mathbb{Z}$, $0 \leq r < b$ tal que $a = bq + r$.

    Ayuda: Considere el conjunto $S = \{n \in \mathbb{N} : \exists q \in \mathbb{Z}(n = a - bq)\}$, y aplique el Principio del Menor Entero.

12. Demuestre que, dados dos naturales $a, b > 0$, existen enteros x, y tales que $ax + by = \text{mcd}(a, b)$.

    Ayuda: Considere el conjunto $S = \{n \in \mathbb{Z}^+ : \exists x, y \in \mathbb{Z}(ax + by = n)\}$. Sea d el menor elemento de S, y demuestre que $d = \text{mcd}(a, b)$.

13. En un torneo de fútbol juegan n equipos, todos contra todos. No se permiten empates. No hay revanchas.

    Demuestre que, si existe un ciclo de la forma "A le gana a B, B le gana a C, C le gana a D,..., Z le gana a A" entonces hay un ciclo de largo 3.

6.2. Clausuras

6.2.1. Funciones n-arias

Definición 29. Sea A un conjunto cualquiera, y sea $n \in \mathbb{N}$. Una función $f : A^n \to A$ es llamada una *operación n-aria definida en A*.

Note que toda operación n-aria es esencialmente una relación $(n+1)$-aria R_f:

$$f(x_1, x_2, \ldots, x_n) = y \leftrightarrow (x_1, x_2, \ldots, x_n, y) \in R_f.$$

6.2.2. Conjuntos cerrados

Definición 30. Sean A un conjunto cualquiera, $n \in \mathbb{N}$, y $f : A^n \to A$ una operación n-aria definida en A.

Un subconjunto $S \subseteq A$ se dice *cerrado bajo* f si, dados $x_1, x_2, \ldots, x_n \in S$ se tiene necesariamente $f(x_1, x_2, \ldots, x_n) \in S$.

Ejemplos.

1. Sean $A = \mathbb{Z}$, $n = 1$, y sea $f : \mathbb{Z} \to \mathbb{Z}$ dada por $f(x) = -x$.

   - El conjunto $\mathbb{P}$ de enteros pares es cerrado bajo f.
   - El conjunto $\mathbb{I}$ de enteros impares es cerrado bajo f.
   - El conjunto $\mathbb{N}$ de los números naturales no es cerrado bajo f.

2. Sean $A = \mathbb{Z}$, $n = 2$, y sea $g : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ dada por $g(x, y) = x + y$.

   - El conjunto $\mathbb{P}$ de enteros pares es cerrado bajo g.
   - El conjunto $\mathbb{I}$ de enteros impares no es cerrado bajo g.

6.2.3. Conjuntos cerrados bajo una relación

Generalicemos las ideas anteriores:

Definición 31. Si A es un conjunto no vacío, $n \in \mathbb{N}$, y $R \subseteq A^{n+1}$ es una relación $(n+1)$-aria cualquiera, entonces un conjunto $S \subseteq A$ se dice *cerrado bajo* R si, dados $x_1, x_2, \ldots, x_n, \in S$, $x_{n+1} \in A$ tales que $(x_1, x_2, \ldots, x_n, x_{n+1}) \in R$, se tiene necesariamente $x_{n+1} \in S$.

En particular, note que la noción anteriormente definida de conjunto cerrado bajo una operación, es un caso particular del concepto de conjunto cerrado bajo una relación.

6.2.4. El *menor conjunto* que satisface ψ

Sea $\mathcal{U}$ un "conjunto universal" dado, sea $\psi(S)$ una propiedad que tienen algunos subconjuntos S de $\mathcal{U}$, y sea $A \subseteq \mathcal{U}$ un conjunto dado, fijo.

Nos interesa encontrar un conjunto $C \subseteq \mathcal{U}$ que satisfaga lo siguiente:

1. $A \subseteq C$.
2. Se cumple $\psi(C)$.
3. Dado cualquier $D \subseteq \mathcal{U}$ tal que $A \subseteq D$ y $\psi(D)$, debe tenerse $C \subseteq D$.

Si existe C que cumpla estas tres propiedades, diremos que C es *el menor subconjunto de $\mathcal{U}$ que contiene a A y satisface ψ*. Note que éste es un elemento mínimo (en el orden de la inclusión) entre los conjuntos que contienen a A y satisfacen ψ.

Note que de existir C que cumpla estas tres propiedades, es único.

Ejemplos. Sea $\mathcal{U} = \mathbb{N}$, y $\psi(S)$ definida por "S es cerrado bajo adición".

- Si $A = \{3\}$, entonces el menor conjunto que contiene a A y satisface ψ (o sea, el menor conjunto que contiene a A y es cerrado bajo adición) es el de los múltiplos positivos de 3.
- Si $A = \{4, 7\}$, entonces el menor conjunto que contiene a A y es cerrado bajo adición es

$$C = \mathbb{N} - \{0, 1, 2, 3, 5, 6, 9, 10, 13, 17\}.$$

  Esto puede ser interpretado como sigue: si n es cualquier entero positivo *excepto 1,2,3,5,6,9,10,13 o 17*, entonces es posible franquear una carta por $\$n$ usando sólo estampillas de \$4 y \$7.

6.2.5. Un problema

Dependiendo de cuál sea la propiedad ψ, es posible que no exista "el menor conjunto que contiene a A y satisface ψ".

Ejemplo. Sean $\mathcal{U} = \mathbb{N}$, $A = \{0\}$, y $\psi(S)$ la propiedad "$S \cap \{1, 2\} \neq \emptyset$".

Entonces no existe un único elemento minimal (en el orden de la inclusión) entre los subconjuntos de $\mathcal{U}$ que contienen a A y satisfacen ψ, y por ende no existe entre ellos un elemento mínimo.

6.2.6. Una definición alternativa

Teorema. Sea $\mathcal{U}$ un "conjunto universal" dado, sea $\psi(S)$ una propiedad que tienen algunos subconjuntos S de $\mathcal{U}$, y sea $A \subseteq \mathcal{U}$ un conjunto dado, fijo.

Si existe un "menor subconjunto C de $\mathcal{U}$ que contiene a A y satisface ψ" entonces

$$C = \bigcap \{S \subseteq \mathcal{U} : A \subseteq S \wedge \psi(S)\}.$$

Demostración. Ejercicio.

Este teorema tiene una suerte de recíproco: si

$$B = \bigcap \{S \subseteq \mathcal{U} : A \subseteq S \wedge \psi(S)\}$$

satisface ψ, entonces B es el menor subconjunto buscado.

6.2.7. Propiedades de clausura

Definición 32. Diremos que una propiedad $\psi(S)$ definida sobre los subconjuntos de un conjunto dado A es una *propiedad de clausura* si $\psi(S)$ es de la forma "S es cerrado bajo cada una de las relaciones $R_1, R_2, \ldots, R_k$", donde k es un entero positivo y $R_1, R_2, \ldots, R_k$ son relaciones definidas en A.

6.2.8. Clausura bajo una relación

Teorema. Sean $\mathcal{U}$ un conjunto universal dado, $A \subseteq \mathcal{U}$ un subconjunto fijo de $\mathcal{U}$, y sea ψ una propiedad de clausura que es satisfecha por algunos de los subconjuntos de $\mathcal{U}$.

Entonces existe un menor subconjunto de $\mathcal{U}$ que contiene a A y satisface ψ.

Demostración. Ejercicio.

Ayuda: aprovéchese del recíproco del teorema anterior: demuestre que

$$B = \bigcap \{S \subseteq \mathcal{U} : A \subseteq S \wedge \psi(S)\}$$

satisface ψ.

Notación. Denotaremos la clausura de A bajo las relaciones $R_1, R_2, \ldots, R_n$ por

$$\mathcal{C}(A; R_1, R_2, \ldots, R_n).$$

Ejemplos.

La clausura aditiva

La mayoría de los ejemplos de clausuras pueden ser presentados como "clausuras bajo relaciones". Por ejemplo:

Sea $S \subseteq \mathbb{Z}$. La "clausura aditiva" de S (el menor conjunto cerrado bajo suma que contiene a S) es la clausura de S bajo la relación

$$\left\{(x, y, z) \in \mathbb{Z}^3 : x + y = z\right\}.$$

Expresiones aritméticas

El conjunto de todas las expresiones aritméticas que se pueden formar con constantes y variables en un lenguaje de programación pueden ser vistas como la clausura del conjunto

$$\{X : X \text{ es una constante o variable}\}$$

bajo las relaciones:

$$\begin{array}{rcl} R_1 & : & \{(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3) : \mathcal{E}_3 = (\mathcal{E}_1 + \mathcal{E}_2)\}, \\ R_2 & : & \{(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3) : \mathcal{E}_3 = (\mathcal{E}_1 \times \mathcal{E}_2)\}, \\ R_3 & : & \{(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3) : \mathcal{E}_3 = (\mathcal{E}_1 \div \mathcal{E}_2)\}, \\ R_4 & : & \{(\mathcal{E}_1, \mathcal{E}_2) : \mathcal{E}_2 = (-\mathcal{E}_1)\}, \\ & \vdots & \end{array}$$

El conjunto $\mathbb{N}$

El conjunto $\mathbb{N}$ puede ser visto como la clausura de $\{\emptyset\}$ bajo la relación "sucesor":

$$\sigma = \{(x, y) : y = x \cup \{x\}\}.$$

6.2.9. Clausura simétrica de una relación

Sea $R \subseteq A \times A$ una relación.

Definimos la *clausura simétrica de* R como

$$R^s = \cap \{S : R \subseteq S \subseteq A \times A \wedge S \text{ es simétrica}\}.$$

R^s es la menor relación simétrica que contiene a R.

Podemos definir R^s como la clausura de R bajo la relación $\mathcal{S} \subseteq (A \times A)^2 = (A \times A) \times (A \times A)$ definida por

$$\mathcal{S} = \{((a,b),(b,a)) : a, b \in A\}.$$

Ejercicio. Defina de las dos maneras anteriores la clausura refleja, la clausura transitiva, la clausura transitiva-refleja, etc., de una relación.

Ejercicio. Defina de las dos maneras anteriores la menor relación de equivalencia que contiene a R.

6.2.10. Otra forma de ver las clausuras

Sean $A \subseteq \mathcal{U}$ y $R \subseteq A^{n+1}$. Definimos:

$$\begin{aligned} S_0 &= A, \\ S_{i+1} &= S_i \cup \{y \in \mathcal{U} : \exists x_1, \ldots, x_n \in S_i (x_1, \ldots, x_n, y) \in R\} \\ &\qquad \text{para } i \geq 0. \end{aligned}$$

Teorema. La clausura de A bajo la relación R es

$$\mathcal{C}(A; R) = \bigcup_{i=0}^{\infty} S_i.$$

Análogamente podemos definir

$$\mathcal{C}(A; R_1, R_2, \ldots, R_n).$$

6.2.11. Capas

Definimos $C_0 = S_0 = A$. Para $i \geq 0$, definimos $C_{i+1} = S_{i+1} - S_i$. Como

$$\mathcal{C}(A; R_1, R_2, \ldots, R_n) = \bigcup_{i=0}^{\infty} C_i,$$

(donde la unión es disjunta), decimos que los C_i son las *capas* de $\mathcal{C}(A; R_1, R_2, \ldots, R_n)$.

6.3. Inducción Estructural

¿Cómo demostrar que todos los elementos de $\mathcal{C}(A; R_1, R_2, \ldots, R_n)$ satisfacen una cierta propiedad?

Podemos usar una variante de inducción, que esencialmente se reduce a hacer inducción sobre los S_i (o sobre los C_i).

Vimos que los números naturales son la clausura del conjunto $\{\emptyset\}$ bajo la función sucesor.

Esto nos da la idea de presentar el principio de inducción en términos de las capas de la construcción de clausura de los naturales, y de adaptar esto a otros objetos definidos como clausuras.

Así, podemos formular el siguiente "Principio de inducción estructural":

Sea $\mathcal{U}$ un conjunto construído como

$$\mathcal{U} = \mathcal{C}(A; R_1, R_2, \ldots, R_n),$$

donde A es un *conjunto base*, y las relaciones R_i son relaciones en A, con "ariedades" $n_i + 1$ respectivamente.

Si $P(x)$ es un predicado que:

1. es verdadero para todo $x \in A$, y
2. cada vez que es verdadero para $x_1, x_2, \ldots, x_{n_i} \in \mathcal{U}$, es verdadero para todo $y \in \mathcal{U}$ tal que $(x_1, x_2, \ldots, x_{n_i}, y) \in R_i$;

entonces $P(x)$ es verdadero para todo $x \in \mathcal{U}$.

Ejercicio. Demuestre este principio de inducción a partir del principio de inducción simple.

6.3.1. Ejemplo: lógica proposicional

Consideremos el lenguaje de todas las fórmulas de la lógica proposicional.

Este lenguaje puede ser considerado como la clausura del conjunto de proposiciones atómicas bajo las funciones $\varphi_\neg$, $\varphi_\wedge$, $\varphi_\vee$, $\varphi_\to$, $\varphi_\leftrightarrow$, donde $\varphi_\neg(P) = (\neg P)$, y —para cada conectivo binario $\star$— se tiene $\varphi_\star(P, Q) = (P \star Q)$.

¿Cómo aprovechar esta definición para demostrar propiedades de las proposiciónes lógicas?

Ejemplo. Demuestre que toda proposición lógica tiene la misma cantidad de paréntesis izquierdos que derechos.

6.3.2. Conjuntos completos de conectivos

Dado un conjunto C de conectivos, decimos que éste es *completo* si para toda proposición φ existe una proposición $\varphi' \Leftrightarrow \varphi$, y que contiene sólo conectivos de C.

Ejemplo. El conjunto $\{\neg, \wedge, \vee\}$ es completo

Demostración. Sea F_0 el conjunto de fórmulas proposicionales atómicas, y sea $\Sigma = F_0 \cup \{(,), \neg, \wedge, \vee, \to, \leftrightarrow, \ldots\}$ el conjunto de todos los símbolos que pueden aparecer en una fórmula proposicional (podríamos considerar otros conectivos binarios).

Sea $E_\neg : \Sigma^* \to \Sigma^*$ dada por $E_\neg(\varphi) = (\neg\varphi)$, y para cada conectivo binario $\star \in \{\wedge, \vee, \to, \leftrightarrow, \ldots\}$, sea $E_\star : \Sigma^* \times \Sigma^* \to \Sigma^*$ dada por $E_\star(\varphi, \psi) = (\varphi \star \psi)$.

Sea $\mathcal{F} = \mathcal{C}(F_0; E_\neg, E_\wedge, E_\vee, E_\to, E_\leftrightarrow, \ldots)$ el conjunto de todas las fórmulas proposicionales que se pueden formar con conectivos tomados de $\{\neg, \wedge, \vee, \to, \leftrightarrow, \ldots\}$.

Definiremos ahora el conjunto $\mathcal{F}'$ de todas las fórmulas proposicionales que se pueden formar con conectivos tomados de $\{\neg, \wedge, \vee\}$.

Formalmente,

$$\mathcal{F}' = \mathcal{C}(F_0; E_\neg, E_\wedge, E_\vee).$$

Probaremos por inducción estructural que, para toda fórmula proposicional $\varphi \in \mathcal{F}$, se cumple el siguiente predicado:

$$P(\varphi): \text{ existe una fórmula proposicional } \varphi' \in \mathcal{F}' \text{ tal que } \varphi' \Leftrightarrow \varphi.$$

Base: si $\varphi \in F_0$ (o sea, si φ es atómica), entonces claramente, tomando $\varphi' = \varphi$ se tiene $\varphi' \in \mathcal{F}'$ y $\varphi' \Leftrightarrow \varphi$. O sea, $P(\varphi)$ se cumple para $\varphi \in F_0$.

Paso inductivo

Supongamos que se tienen dos fórmulas $\varphi_1, \varphi_2 \in \mathcal{F}$ tales que se cumple $P(\varphi_1)$ y $P(\varphi_2)$. Debemos demostrar que se cumple $P((\neg\varphi_1))$, $P((\varphi_1 \wedge \varphi_2))$, $P((\varphi_1 \vee \varphi_2))$, $P((\varphi_1 \to \varphi_2))$, $P((\varphi_1 \leftrightarrow \varphi_2))$, etc.

En efecto: probaremos que se cumple $P(\varphi)$, en cada uno de los siguientes casos:

$\varphi = (\neg\varphi_1)$**:** por HI, existe $\varphi'_1 \in \mathcal{F}'$ tal que $\varphi'_1 \Leftrightarrow \varphi_1$. Así,

$$\varphi' = (\neg\varphi'_1) \Leftrightarrow (\neg\varphi_1) = \varphi.$$

$\varphi = (\varphi_1 \wedge \varphi_2)$**:** por HI, existen $\varphi'_1, \varphi'_2 \in \mathcal{F}'$ tales que $\varphi'_1 \Leftrightarrow \varphi_1$, $\varphi'_2 \Leftrightarrow \varphi_2$. Así,

$$\varphi' = (\varphi'_1 \wedge \varphi'_2) \Leftrightarrow (\varphi_1 \wedge \varphi_2) = \varphi.$$

$\varphi = (\varphi_1 \vee \varphi_2)$**:** por HI, existen $\varphi'_1, \varphi'_2 \in \mathcal{F}'$ tales que $\varphi'_1 \Leftrightarrow \varphi_1$, $\varphi'_2 \Leftrightarrow \varphi_2$. Así,

$$\varphi' = (\varphi'_1 \vee \varphi'_2) \Leftrightarrow (\varphi_1 \vee \varphi_2) = \varphi.$$

$\varphi = (\varphi_1 \to \varphi_2)$**:** por HI, existen $\varphi'_1, \varphi'_2 \in \mathcal{F}'$ tales que $\varphi'_1 \Leftrightarrow \varphi_1$, $\varphi'_2 \Leftrightarrow \varphi_2$. Así,

$$\varphi' = (\neg\varphi'_1 \vee \varphi'_2) \Leftrightarrow (\varphi'_1 \to \varphi'_2) \Leftrightarrow (\varphi_1 \to \varphi_2) = \varphi,$$

y así para cada conectivo binario.

Ejercicio. Complete la demostración con cada uno de los posibles conectivos binarios faltantes.

6.3.3. Otro conjunto completo

Demostraremos que $\{\neg, \wedge\}$ es un conjunto completo.

Para ello, definiremos el conjunto $\mathcal{F}''$ como el conjunto de todas las fórmulas proposicionales que se pueden formar con conectivos tomados de $\{\neg, \wedge\}$.

Formalmente,

$$\mathcal{F}'' = \mathcal{C}(F_0; E_\neg, E_\wedge).$$

Probaremos por inducción estructural que, para toda fórmula proposicional $\varphi' \in \mathcal{F}'$, se cumple el siguiente predicado:

$$P(\varphi') : \text{ existe una fórmula proposicional } \varphi'' \in \mathcal{F}'' \text{ tal que } \varphi'' \Leftrightarrow \varphi'.$$

Pregunta: ¿Por qué basta probar $P(\varphi')$ para todo $\varphi' \in \mathcal{F}'$? (probar $P(\varphi)$ para todo $\varphi \in \mathcal{F}$ sería más trabajo).

Demostración de que $\{\neg, \wedge\}$ es completo

Base: si $\varphi' \in F_0$ (o sea, si φ' es atómica), entonces claramente, tomando $\varphi'' = \varphi'$ se tiene $\varphi'' \in \mathcal{F}''$ y $\varphi'' \Leftrightarrow \varphi'$. O sea, $P(\varphi')$ se cumple para $\varphi' \in F_0$.

Paso inductivo: Supongamos que se tienen dos fórmulas $\varphi'_1, \varphi'_2 \in \mathcal{F}'$ tales que se cumple $P(\varphi'_1)$ y $P(\varphi'_2)$. Debemos demostrar que se cumple $P((\neg\varphi'_1))$, $P((\varphi'_1 \wedge \varphi'_2))$ y $P((\varphi'_1 \vee \varphi'_2))$.

En efecto: probaremos que se cumple $P(\varphi')$, en cada uno de los siguientes casos:

$\varphi' = (\neg\varphi'_1)$**:** por HI, existe $\varphi''_1 \in \mathcal{F}''$ tal que $\varphi''_1 \Leftrightarrow \varphi'_1$. Así,

$$\varphi'' = (\neg\varphi''_1) \Leftrightarrow (\neg\varphi'_1) = \varphi'.$$

$\varphi' = (\varphi'_1 \wedge \varphi'_2)$**:** por HI, existen $\varphi''_1, \varphi''_2 \in \mathcal{F}''$ tales que $\varphi''_1 \Leftrightarrow \varphi'_1$, $\varphi''_2 \Leftrightarrow \varphi'_2$. Así,

$$\varphi'' = (\varphi''_1 \wedge \varphi''_2) \Leftrightarrow (\varphi'_1 \wedge \varphi'_2) = \varphi'.$$

$\varphi' = (\varphi'_1 \vee \varphi'_2)$**:** por HI, existen $\varphi''_1, \varphi''_2 \in \mathcal{F}''$ tales que $\varphi''_1 \Leftrightarrow \varphi'_1$, $\varphi''_2 \Leftrightarrow \varphi'_2$. Así,

$$\varphi' = (\neg((\neg\varphi''_1) \wedge (\neg\varphi''_2))) \Leftrightarrow (\varphi''_1 \vee \varphi''_2) \Leftrightarrow (\varphi'_1 \vee \varphi'_2) = \varphi'.$$

Ejercicios.

1. Demuestre que $\{\neg, \wedge\}$ es un conjunto completo de conectivos.
2. Demuestre que $\{\neg, \vee\}$ es un conjunto completo de conectivos.
3. Demuestre que $\{\neg, \to\}$ es un conjunto completo de conectivos.
4. Demuestre que $\{\wedge, \vee, \to\}$ no es un conjunto de conectivos.

6.3.4. Conjuntos no completos

¿Cómo demostrar que un conjunto de conectivos *no es completo*? Una posibilidad es probar que todas las proposiciones que se pueden formar con ese conjunto satisfacen alguna propiedad común, pero que existen proposiciones que no la satisfacen.

Ilustraremos esto con un ejemplo.

Ejemplo. El conjunto $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$ no es completo.

Para demostrar esto, definiremos el conjunto $\mathcal{H}$ como el conjunto de todas las fórmulas proposicionales que se pueden formar con conectivos tomados de $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$, y probaremos que toda fórmula de $\mathcal{H}$ se hace verdadera en la asignación de verdad que hace verdaderas a todas las proposiciones atómicas.

Formalmente,

$$\mathcal{H} = \mathcal{C}(F_0; E_\wedge, E_\vee, E_\rightarrow, E_\leftrightarrow),$$

y definimos σ_0 como la asignación de verdad que hace verdaderas a todas las fórmulas atómicas.

Es posible probar por inducción estructural que, para toda fórmula proposicional $\varphi \in \mathcal{H}$, se cumple el siguiente predicado:

$$P(\varphi) : \sigma_0(\varphi) = 1.$$

También es posible probar que existe una fórmula $\psi \in \mathcal{F}$ que no satisface $P(\psi)$. Así, ninguna fórmula $\varphi \in \mathcal{H}$ es equivalente a ψ.

Ejercicio. Complete la demostración.

Ejercicio. Demuestre que el conjunto de conectivos $\{\neg, \oplus\}$ no es completo (el conectivo $\oplus$ es "o excluyente").

6.3.5. Ejercicios

1. Considere el conjunto $\mathbb{P}$ de proposiciones definido como

   $$\mathcal{C}(\{X : X \text{ es proposición atómica}\}; f_1, f_2, f_3, f_4, f_5)$$

   donde

   $$\begin{aligned} f_1(x) &= (\neg x), \\ f_2(x, y) &= (x \wedge y), \\ f_3(x, y) &= (x \vee y), \\ f_4(x, y) &= (x \rightarrow y), \\ f_5(x, y) &= (x \leftrightarrow y). \end{aligned}$$

   Sea σ una asignación de verdad, vale decir, una función

   $$\sigma : \{X : X \text{ es proposición atómica}\} \rightarrow \{0, 1\}.$$

   Muestre cómo definir recursivamente una función $\bar{\sigma} : \mathbb{P} \rightarrow \{0, 1\}$ que extienda la idea de asignación de verdad a proposiciones arbitrarias.

2. Para cada $P \in \mathbb{P}$ definimos:

   $$\begin{aligned} a(P) &: \text{número de proposiciones atómicas en } P, \\ n(P) &: \text{número de veces que el conectivo } \neg \text{ aparece en } P, \\ o(P) &: \text{número de otros conectivos en } P, \\ i(P) &: \text{número de (en } P, \\ d(P) &: \text{número de) en } P. \end{aligned}$$

   Demuestre que para toda proposición $P \in \mathbb{P}$ se cumple que:

a) $i(P) = d(P)$.

b) $i(P) = n(P) + o(P)$.

c) $a(P) = o(P) + 1$.

3. Demuestre que el conjunto de conectivos $\{\neg, \rightarrow\}$ es completo.

4. Definimos el conectivo $\mid$ (conocido como *la raya de Sheffer*, y que en computación también es llamado *nand*) por la equivalencia $p \mid q \Leftrightarrow \neg(p \wedge q)$. Demuestre que el conjunto de conectivos $\{\mid\}$ es completo.

5. Definimos el conectivo $\downarrow$ (ni) por la equivalencia $p \downarrow q \Leftrightarrow (\neg p \wedge \neg q)$. Demuestre que el conjunto de conectivos $\{\downarrow\}$ es completo.

6. Demuestre que el conjunto de conectivos $\{\wedge, \vee\}$ no es completo.

   Indicación: Defina adecuadamente el conjunto de fórmulas que sólo usan conectivos de $\{\wedge, \vee\}$, y demuestre que toda fórmula P de dicho conjunto satisface $\bar{\sigma}(P) = 1$ donde $\bar{\sigma}$ es la extensión de la asignación de verdad que a toda fórmula atómica le asigna 1 (ver ejercicio 1 para la definición de $\bar{\sigma}$).

7. Definimos el conectivo $\oplus$ (ó excluyente) por la equivalencia $p \oplus q \Leftrightarrow (p \leftrightarrow \neg q)$. Demuestre que el conjunto de conectivos $\{\oplus, \rightarrow\}$ es completo.

8. Demuestre que el conjunto de conectivos $\{\oplus, \leftrightarrow, \neg\}$ no es completo.

   Indicación: Defina adecuadamente el conjunto de fórmulas que sólo usan conectivos de $\{\oplus, \leftrightarrow, \neg\}$, y demuestre que, dada cualquier fórmula P de dicho conjunto que contenga al menos dos proposiciones atómicas distintas, la cantidad de asignaciones de verdad que satisfacen a P es par.

9. Deduzca del ejercicio anterior que ninguno de los siguientes conjuntos de conectivos es completo:

   a) $\{\oplus\}$.

   b) $\{\oplus, \leftrightarrow\}$.

   c) $\{\leftrightarrow, \neg\}$.

10. ¿Es el conjunto de conectivos $\{\oplus, \wedge, \vee\}$ completo? Justifique su respuesta.

Capítulo 7

Corrección de programas

Dado un programa o algoritmo P, nos interesa probar que éste es *correcto*, es decir, que satisface ciertas *especificaciones*.

En particular, nos interesa probar que, si se satisfacen ciertas *precondiciones* (proposiciones que involucran algunas variables del programa) entonces el programa termina y se satisfacen ciertas *postcondiciones*.

Estudiamos dos tipos de demostración de corrección de programas: programas *iterativos* y programas *recursivos*.

7.1. Corrección de programas iterativos

Supongamos que queremos demostrar que un programa no recursivo (o sea, uno en que no hay llamadas recursivas de un algoritmo a sí mismo) es correcto. La única dificultad proviene de la posibilidad de que el programa contenga *loops* (iteraciones), por lo que nos centramos en este caso.

Generalmente, dividimos la demostración de que un programa iterativo es correcto en dos tareas independientes, que llamamos *corrección parcial* y *terminación*:

Corrección Parcial: si el programa termina, entonces se satisfacen las postcondiciones.

Terminación: el programa se detiene.

Para demostrar que un algoritmo iterativo es parcialmente correcto, generalmente se demuestra que una cierta condición (el llamado *invariante del loop* o *invariante de los loops*) se cumple siempre en los distintos momentos en que la ejecución del algoritmo se encuentra en un mismo punto del loop (generalmente al principio o al final, pero no necesariamente).

Para ello, para cada variable v que aparece en el algoritmo, denotamos por v_i al valor de dicha variable en el momento en que se alcanza el punto designado en el loop durante (o después de) la iteración i-ésima, y en el invariante del loop enunciamos relaciones entre los distintos valores de las variables en dicho punto.

Al lema que afirma que esta condición efectivamente es un invariante para todo número de iteraciones, la llamamos el *lema del invariante*, y como consecuencia de éste deducimos la corrección parcial.

Para demostrar terminación de un algoritmo iterativo, generalmente consideramos una expresión entera E cuyo valor va cambiando con cada iteración del algoritmo. Llamando E_i al valor de E tras i iteraciones del algoritmo, si logramos demostrar que $E_{i+1} < E_i$ para todo i (o sea, que E *decrece estrictamente* con cada iteración, y que existe $k \in \mathbb{Z}$ tal que $\forall i \in \mathbb{N}(E_i \geq k)$ (o sea, que $\{E_i : i \in \mathbb{N}\}$ es un conjunto acotado inferiormente) entonces podremos concluir que efectivamente el algoritmo debe terminar.

7.1.1. Ejemplo: mezcla de dos archivos

Consideremos el siguiente algoritmo, que toma dos archivos f_1 y f_2, ordenados crecientemente, y produce un nuevo archivo f que contiene todos los elementos de f_1 y f_2, en orden.

Algorithm 1 MEZCLA(f_1, f_2, f)

```
1: while ¬eof(f1) ∧ ¬eof(f2) do
2:    if primer elemento de f1 < primer elemento de f2 then
3:       copiar el primer elemento de f1 al final de f, eliminar el primer elemento de f1
4:    else
5:       copiar el primer elemento de f2 al final de f, eliminar el primer elemento de f2
6:    end if
7: end while
8: while ¬eof(f1) do
9:    Copiar el primer elemento de f1 al final de f, eliminar el primer elemento de f1
10: end while
11: while ¬eof(f2) do
12:    Copiar el primer elemento de f2 al final de f, eliminar el primer elemento de f2
13: end while
```

Demostraremos primero que el algoritmo de mezcla es *parcialmente correcto* respecto al siguiente par de condiciones:

Precondición: f_1 y f_2 están ordenados crecientemente, y $f = \emptyset$

Postcondición: f contiene los mismos elementos que originalmente contení an f_1 y f_2, ordenados crecientemente

Una vez demostrada la corrección parcial, probaremos que el algoritmo efectivamente termina.

Demostración de la corrección parcial

Para cada $i \geq 0$, sean f_i, f_{1_i} y f_{2_i} los conjuntos de valores presentes en los distintos archivos tras haberse producido en total i iteraciones de los distintos loops.

Para demostrar que este algoritmo es parcialmente correcto respecto a este par de condiciones, demostramos que, después de i iteraciones:

- $f_i \cup f_{1_i} \cup f_{2_i} = f_{1_0} \cup f_{2_0}$,
- f_i está ordenado en orden creciente, y
- dados $x \in f_i$ e $y \in f_{1_i} \cup f_{2_i}$, se tiene $x \leq y$.

Más formalmente: demostramos el siguiente lema:

Lema. Para todo $i \in \mathbb{N}$, si en total los loops se ejecutan al menos i veces, entonces

$$(f_i \cup f_{1_i} \cup f_{2_i} = f_{1_0} \cup f_{2_0}) \wedge (f_i \text{ está ordenado en orden creciente}) \wedge \forall x \in f_i, y \in f_{1_i} \cup f_{2_i} (x \leq y). \tag{7.1}$$

La condición (7.1) es nuestro "invariante de los loops". Note que tenemos un solo invariante para los tres loops.

Demostración del lema del invariante. Hacemos esta demostración por inducción sobre i:

Base: que el invariante es verdadero para $i = 0$ es claro.

Supongamos que el invariante se cumple para $i = n$, y tratemos de demostralo para $i = n+1$. Si no hay una iteración $n + 1$, el resultado es obvio, por lo que supondremos que sí la hay. Hay cuatro casos:

- Si f_{1_i} y f_{2_i} son no vacíos, entonces se debe realizar una iteración más del primer loop. Sea x el menor entre los primeros elementos de f_{1_i} y f_{2_i}.

  Si x es el primer elemento de f_{1_i}, entonces $f_{i+1} = f_i \cup \{x\}$, $f_{1_{i+1}} = f_{1_i} - \{x\}$, $f_{2_{i+1}} = f_{2_i}$, por lo que $f_{i+1} \cup f_{1_{i+1}} \cup f_{2_{i+1}} = f_i \cup f_{1_i} \cup f_{2_i} = f_{1_0} \cup f_{2_0}$.

  Por la tercera parte de la hipótesis de inducción, todos los elementos de f_i son $\leq x$, y como f_i está ordenado en forma ascendente y x es agregado al final de $f_i$para obtener f_{i+1}, se tiene que f_{i+1} está ordenado en forma creciente.

  Finalmente, todo elemento de f_i es $\leq$ que todo elemento de $f_{1_i} \cup f_{2_i}$ (y por lo tanto, que todo elemento de $f_{1_{i+1}} \cup f_{2_{i+1}}$). Como el único otro elemento de f_{i+1} es x, y x es el menor elemento de $f_{1_i} \cup f_{2_i}$, se tiene que todos los elementos de f_{i+1} son $\leq$ que todos los elementos de $f_{1_i} \cup f_{2_i}$ (y por lo tanto, que todo elemento de $f_{1_{i+1}} \cup f_{2_{i+1}}$).

- Si f_{1_i} es no vacío, pero f_{2_i} lo es, la demostración es similar a la anterior.

  Ejercicio. Complete los detalles.

- Si f_{2_i} es no vacío, pero f_{1_i} lo es, la demostración es idéntica a la anterior, cambiando 1 por 2.

□

Debido al Lema del invariante, si el algoritmo termina después de k iteraciones, al terminar se tiene $f_k \cup f_{1_k} \cup f_{2_k} = f_{1_0} \cup f_{2_0}$.

Como $f_{1_k} = f_{2_k} = \emptyset$ (debido a las condiciones de término de los loops), se tiene $f_k = f_{1_0} \cup f_{2_0}$.

Por la segunda parte del lema del invariante, f_k está ordenado.

Esto concluye la demostración de la corrección parcial.

Terminación

Para demostrar que el algoritmo termina, consideramos la expresión $E_i = |f_{1_i}| + |f_{2_i}|$. Claramente, ésta es siempre ≥ 0.

Además, esta expresión es estrictamente decreciente: como en cada iteración uno de los archivos pierde un elemento, $E_{i+1} < E_i$.

Así, cada uno de los loops debe terminar en algún momento.

7.1.2. Otro ejemplo: búsqueda binaria

Consideremos un tipo de dato T con un orden total $\preceq$ (por ejemplo, si el tipo de dato es "*string*", $\preceq$ podría ser orden lexicográfico).

Queremos escribir un programa que tome como argumentos un arreglo A (cuyos elementos son de tipo T) y un elemento x de tipo T, de modo que, cada vez que se cumpla la siguiente *precondición*, termine satisfaciendo la *postcondición* indicada:

Precondición: El arreglo $A[1..n]$ está *ordenado*, o sea, $A[i] \preceq A[i+1]$ para todo $i \in \{1, \ldots, n-1\}$.

Postcondición: El valor retornado por el programa es algún $t \in \mathbb{N}$ tal que: o bien $1 \leq t \leq n \wedge A[t] = x$ (si existe al menos un t que satisfaga esta condición), o bien $t = 0$ (si no hay tal t).

Probaremos que el siguiente programa es correcto respecto a este par de pre- y post-condición.

Algorithm 2 BinSearch(A, n, x)

```
1: f ← 1
2: l ← n
3: while f ≠ l do
4:    m ← ⌊(f+l)/2⌋
5:    if A[m] ≥ x then
6:       l ← m
7:    else
8:       f ← m + 1
9:    end if
10: end while
11: if A[f] = x then
12:    return f
13: else
14:    return 0
15: end if
```

Demostración de la corrección de BinSearch

Demostramos:

1. **Corrección parcial:** si A es un arreglo ordenado de largo $n \geq 1$, entonces si BinSearch(A, n, x) termina, retorna un entero t tal que $1 \leq t \leq n$ y $A[t] = x$ si un tal t existe; si no, BinSearch(A, n, x) retorna 0.
2. **Terminación:** si A es un arreglo ordenado de largo $n \geq 1$, entonces BinSearch(A, n, x) termina.

Corrección parcial

Demostraremos que, si las precondiciones se cumplen antes de que el programa comience, entonces al final de cada iteración se cumple lo siguiente:

$$1 \leq f \leq l \leq n, \text{ y si } x \in \{A[1], \ldots, A[n]\} \text{ entonces } x \in \{A[f], \ldots, A[l]\}.$$

Previamente, necesitamos el siguiente lema:

Lema. Dados dos enteros f, l tales que $f < l$, se tiene

$$f \leq \left\lfloor \frac{f+l}{2} \right\rfloor < l.$$

Demostración. Si $l - f$ es par, digamos $l - f = 2t > 0$, entonces

$$\left\lfloor \tfrac{f+l}{2} \right\rfloor = \left\lfloor f + \tfrac{l-f}{2} \right\rfloor = \lfloor f + t \rfloor = f + t.$$

Como $f \leq f + t < f + 2t = l$, se obtiene el resulrado deseado.

Si $l - f$ es impar, digamos $l - f = 2t + 1$, entonces

$$\left\lfloor \tfrac{f+l}{2} \right\rfloor = \left\lfloor f + \tfrac{l-f}{2} \right\rfloor = \lfloor f + t \rfloor = f + t$$

con la misma conclusión. □

Note que este lema no es nuestro *lema del invariante*.

Lema. Invariante del loop*:*

Para cada variable v del programa y cada $i \in \mathbb{N}$, sea v_i el valor de v después de i iteraciones del loop **WHILE**.

Sea $P(i)$ el siguiente predicado:

$P(i)$: si el loop tiene (al menos) i iteraciones, entonces:

1. $1 \leq f_i \leq l_i \leq n$, *y*
2. *si* $x \in \{A[1], \ldots, A[n]\}$, *entonces* $x \in \{A[f_i], \ldots, A[l_i]\}$

$P(n)$ es verdadera para todo $n \in \mathbb{N}$.

Demostración. La haremos por inducción en n.

Base: $i = 0$. Tenemos $f_0 = 1$, $l_0 = n$. La primera parte de $P(0)$ es verdadera gracias a que $n \geq 1$ (precondición).

La segunda parte de $P(0)$ es trivialmente verdadera.

Paso inductivo:

Supongamos que, tras j iteraciones del loop se tiene $P(j)$, o sea,

1. $1 \leq f_j \leq l_j \leq n$, y
2. si $x \in \{A[1], \ldots, A[n]\}$, entonces $x \in \{A[f_j], \ldots, A[l_j]\}$

Si el programa termina tras j iteraciones, entonces $P(j+1)$ es trivialmente cierto.

Supongamos que el loop tiene al menos $j+1$ iteraciones. Entonces (por la condición de término del loop) $f_j < l_j$ y por lo tanto (gracias al Lema)

$$f_j \leq m_{j+1} < l_j.$$

Debido al programa, debe tenerse $f_{j+1} = f_j$ y $l_{j+1} = m_{j+1}$, o bien $f_{j+1} = m_{j+1} + 1$ y $l_{j+1} = l_j$. En cualquiera de los dos casos, junto con la primera parte de la HI tenemos $1 \leq f_{j+1} \leq l_{j+1} \leq n$. Ésta es la primera parte de $P(j+1)$.

Supongamos ahora que $x \in \{A[1], \ldots, A[n]\}$. Por la primera parte de $P(j)$, $x \in \{A[f_j], \ldots, A[l_j]\}$. Debemos probar que $x \in \{A[f_{j+1}], \ldots, A[l_{j+1}]\}$.

Hay tres casos:

- $A[m_{j+1}] = x$. En este caso, por el programa, $f_{j+1} = f_j$ y $l_{j+1} = m_{j+1}$, y así, obviamente $x \in \{A[f_{j+1}], \ldots, A[l_{j+1}]\}$.
- $A[m_{j+1}] > x$. En este caso, ya que A está ordenado, $A[t] > x$ para todo t tal que $m_{j+1} \leq t \leq l_j$. Ya que x está en $\{A[f_j], \ldots, A[l_j]\}$ pero no en $\{A[m_{j+1}], \ldots, A[l_j]\}$, debe tenerse $x \in \{A[f_j], \ldots, A[m_{j+1}]\}$. Como (por el programa) $f_{j+1} = f_j$ y $l_{j+1} = m_{j+1}$, se tiene $x \in \{A[f_{j+1}], \ldots, A[l_{j+1}]\}$, que es lo que queríamos.
- $A[m_{j+1}] < x$. Ya que A está ordenado, $A[t] < x$ para todo t tal que $f_j \leq t \leq m_{j+1}$. Ya que x está en $\{A[f_j], \ldots, A[l_j]\}$ pero no en $\{A[f_j], \ldots, A[m_{j+1}]\}$, debe tenerse $x \in \{A[m_{j+1}+1], \ldots, A[l_j]\}$. Como (por el programa) $f_{j+1} = m_{j+1} + 1$ y $l_{j+1} = l_j$, se tiene $x \in \{A[f_{j+1}], \ldots, A[l_{j+1}]\}$, que es lo que queríamos.

□

Hemos probado el lema del invariante del loop.

Conclusión:

Nos falta probar que si se cumplen las precondiciones y el programa termina, entonces al terminar se cumple la postcondición.

Supongamos que el programa termina. Así, el loop se ejecuta una cantidad finita de veces, digamos k. Por la condición de término del loop, $f_k = l_k$. Por el lema del invariante del loop, $1 \leq f_k \leq n$.

Hay dos casos:

1. Hay algún t tal que $1 \leq t \leq n$ tal que $A[t] = x$. Por lo tanto $x \in \{A[1], \ldots, A[n]\}$, y por la primera parte de $P(k)$, $x \in \{A[f_k], \ldots, A[l_k]\}$, o sea, $x = A[f_k]$. Pero el valor retornado es precisamente f_k, que es lo que requiere la postcondición en este caso.

2. Para todo t tal que $1 \leq t \leq n$, se tiene $A[t] \neq x$. Por la primera parte de $P(k)$, $1 \leq f_k \leq n$, y por lo tanto $A[f_k] \neq x$. Así, el programa retorna 0, que es lo que requiere la postcondición en este caso.

Terminación

En general, para demostrar que un loop termina, buscamos una expresión entera E en términos de las variables del programa, que no es nunca negativa y que decrece en cada iteración.

Por ejemplo, para demostrar que BinSearch(A, n, x) termina, consideramos $E = l - f$. Claramente:

- $E_j = l_j - f_j \geq 0$, gracias al lema del invariante del loop, y
- $E_{j+1} < E_j$, ya que o bien $E_{j+1} = \left\lceil \frac{f_j + l_j}{2} \right\rceil - f_j < l_j - f_j$, o bien $E_{j+1} = l_j - \left(\left\lceil \frac{f_j + l_j}{2} \right\rceil + 1 \right) < l_j - f_j$).

Note que, en el fondo, estamos usando el *Principio del Buen Orden*: el conjunto $S = \{E_0, E_1, E_2, \dots\}$ es un subconjunto no vacío de los naturales, y por lo tanto tiene un "primer elemento" E_k. Como E_i decrece estrictamente en cada pasada, el loop se ejecuta exactamente k veces (si no, existiría $E_{k+1} < E_k$ lo que sería una contradicción).

7.2. Corrección de programas recursivos

Hemos aprendido a demostrar que un programa basado en un loop es correcto.

¿Qué hacer para demostrar que un programa recursivo es correcto?

Ejemplo: Mergesort

Considere el algoritmo RecMergeSort que se presenta a continuación.

RecMergeSort ordena los elementos del arreglo A entre las posiciones 1 y n. Para esto llama al algoritmo recursivo AuxRecMergeSort, que ordena el arreglo entre los valores $A[f]$ y $A[l]$.

Algorithm 3 RecMergeSort(A, f, l)

AuxRecMergeSort(A, f, l)

```
1: if f < l then
2:     m ← ⌊(f+l)/2⌋
3:     AuxRecMergeSort(A, f, m)
4:     AuxRecMergeSort(A, m + 1, l)
5:     Mezclar(A[f ... m], A[m + 1 ... l], A[f ... l])
6: end if
```

RecMergeSort(A, n)

```
1: return AuxRecMergeSort(A, 1, n)
```

Corrección del algoritmo

Para demostrar que RecMergeSort correctamente ordena A entre los índices 1 y n, demostramos que para todo $k \in \mathbb{N}$ se satisface la siguiente afirmación:

$$P(k): \text{ si } l - f = k \quad \text{entonces} \quad \begin{array}{l} AuxRecMergeSort(A, f, l) \\ \text{ordena correctamente } A[f] \dots A[l]. \end{array}$$

Demostración. La demostración de que $\forall k \in \mathbb{N}(P(k))$ es por inducción sobre k (segundo principio).

Nuestra hipótesis de de inducción es que $\forall t \in \mathbb{N}(0 \leq t < k \to P(t))$.

En efecto:

Base: si $k = 0$ entonces $\{A[f] \dots A[l]\}$ tiene un solo elemento, y como el algoritmo no hace nada en este caso, deja dicha porción del arreglo ordenado.

Paso inductivo

Sea $k > 0$. Entonces $t = m - f = \left\lfloor \frac{f+l}{2} \right\rfloor - m$ y $t' = f - (m+1)$ son tales que $0 \leq t, t' < k$, por lo que (HI) tanto AuxRecMergeSort(A, f, m) como AuxRecMergeSort$(A, m+1, f)$ correctamente ordenan los sub-arreglos $A[f] \dots A[m]$ y $A[m+1] \dots A[l]$.

Así, AuxRecMergeSort(A, f, m), al realizar las dos llamadas recursivas, deja ordenados los sub-arreglos $A[f] \dots A[m]$ y $A[m+1] \dots A[l]$, e inmediatamente después llama a $Mezclar$ con parámetros adecuados para dejar ordenada la porción de arreglo $A[f] \dots A[l]$ (recuerde que demostramos que $Mezclar$ correctamente mezcla dos archivos ordenados generando uno nuevo; es trivial adaptar dicho algoritmo, y dicha demostración, al caso en que se tienen arreglos en lugar de archivos).

□

Ejemplo: Versión recursiva de búsqueda binaria

El siguiente algoritmo es una versión recursiva del algoritmo de búsqueda binaria visto en la sección anterior. De manera similar al ejemplo anterior, el algoritmo principal llama a un algoritmo auxiliar que hace el trabajo.

Algorithm 4 BinSearch(A, n, x)

```
AuxRecBinSearch(A, f, l, x)
 1: if f = l then
 2:    if A[f] = x then
 3:       return f
 4:    end if
 5:    return 0
 6: end if
 7: m ← ⌊(f+l)/2⌋
 8: if A[m] ≥ x then
 9:    return AuxRecBinSearch(A, f, m, x)
10: else
11:    return AuxRecBinSearch(A, m+1, l, x)
12: end if
RecBinSearch(A, n, x)
 1: return AuxRecBinSearch(A, 1, n, x)
```

Demostración de la corrección de RecBinSearch

Demostramos que, para todo $i \in \mathbb{N}$, se cumple:

$P(i)$: Si A satisface la precondición y $l - f = i$ entonces AuxRecBinSearch(A, f, l, x) termina y retorna algún t tal que $f \leq t \leq l$ y $A[t] = x$ (si existe tal t) o 0, en caso contrario.

Esto lo demostraremos por inducción en i, usando PICV.

Sea $n \in \mathbb{N}$. Supongamos que se cumple $P(k)$ para todo $k \in \mathbb{N}, k < n$.

Hay dos casos:

Base: Si $n = l - f = 0$, entonces $l = f$ y el algoritmo termina. Si $A[f] = x$ entonces existe el t buscado ($t = f$) y el algoritmo retorna f; si $A[f] \neq x$ entonces no existe el t buscado y el algoritmo retorna 0.

Paso inductivo: Si $n = l - f > 0$ entonces el algoritmo no termina en el primer **if**, y debe realizar una llamada recursiva.

Si $A[m] \geq x$ entonces hay dos subcasos:

- Si $A[m] = x$ entonces claramente hay un t entre f y m tal que $A[t] = x$ (a saber, $t = m$); como $k = m - f < l - f = n$ se cumple $P(k)$, y por lo tanto la llamada recursiva AuxRecBinSearch(A, f, m, x) retornará un t como el buscado (aunque no necesariamente $t = m$).
- Si $A[m] > x$ entonces todo $A[t]$ con $m \leq t \leq l$ es $> x$, y por lo tanto, si existe t tal que $f \leq t \leq l$ y $A[t] = x$, entonces debe tenerse $f \leq t \leq m$. Por lo tanto, si existe dicho t, es retornado por AuxRecBinSearch(A, f, m, x); y si no existe dicho t entonces esta llamada recursiva retorna 0. En cualquier caso, AuxRecBinSearch(A, f, l, x) retorna lo que debe para cumplir la postcondición.

Supongamos ahora que $A[m] < x$. Entonces (como A está ordenado) todos los $A[t]$ con $f \leq t \leq m$ son $< x$. Por lo tanto, si existe t tal que $f \leq t \leq l$ y $A[t] = x$, entonces debe tenerse $m < t \leq l$. Por lo tanto, si existe dicho t, es retornado por AuxRecBinSearch$(A, m + 1, l, x)$; y si no existe dicho t entonces esta llamada recursiva retorna 0. En cualquier caso, AuxRecBinSearch(A, f, l, x) retorna lo que debe para cumplir la postcondición.

Conclusión

¿Hemos demostrado que la versión recursiva de *BinSearch* es correcta? En realidad, falta demostrar que tras la llamada original AuxRecBinSearch$(A, 1, n, x)$ se satisface la postcondición.

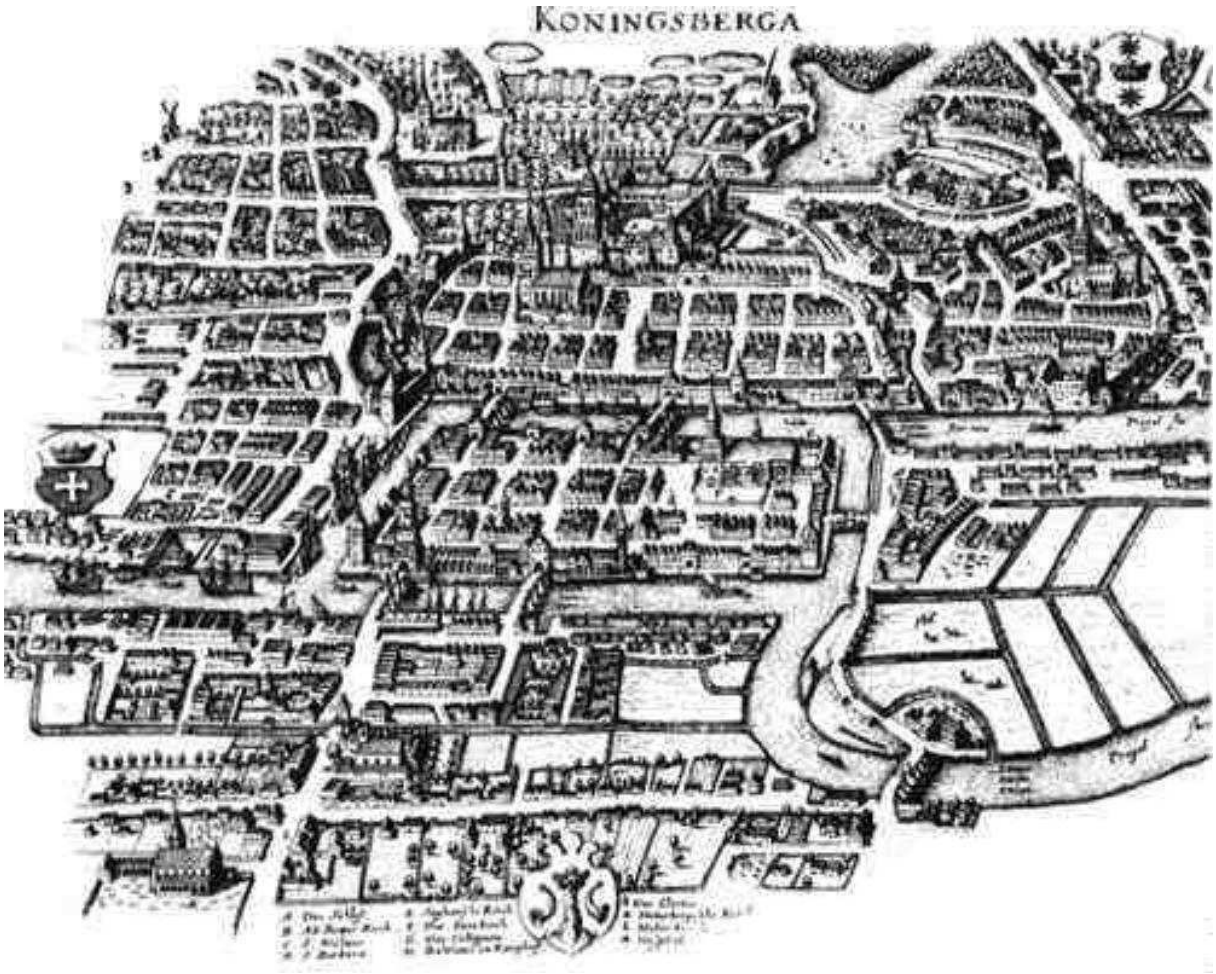
Pero esto es consecuencia de la propiedad demostrada, ya que es equivalente a $P(n - 1)$.

Capítulo 8

Grafos

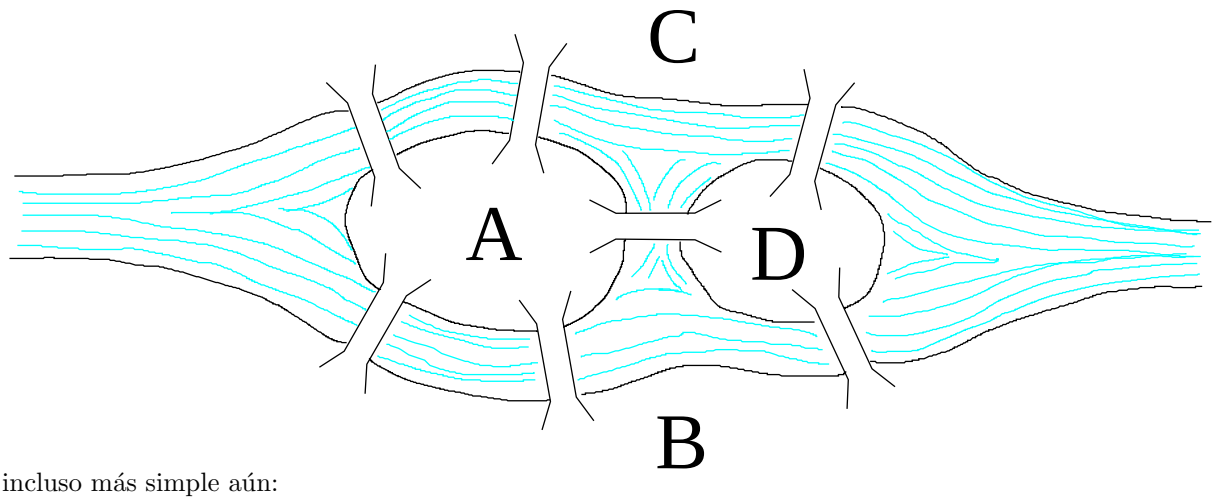
8.1. Motivación: los puentes de Königsberg

La Teoría de Grafos nació el año 1736, en la ciudad de Königsberg (hoy Khaliningrado). La ciudad era atravesada por el río Pregel, el que daba lugar a dos islas, conectadas entre ellas —y con las orillas— por siete puentes.

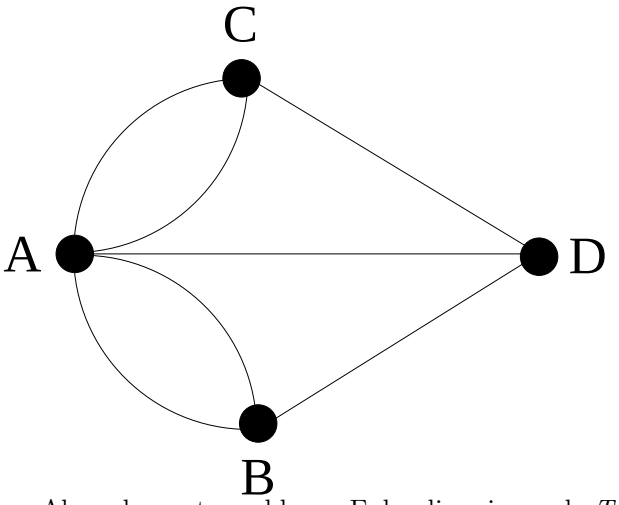


Los habitantes de Königsberg se preguntaban si era posible recorrer todos los puentes de la ciudad, sin repetir ninguno.

En 1736, Leonhardt Euler resolvió este problema, considerando la siguiente versión simplificada de éste:



O, incluso más simple aún:



Al resolver este problema, Euler dio origen a la *Teoría de Grafos*. Antes de entrar a estudiar en detalle éste y otros problemas, definiremos algunos conceptos básicos del área.

8.2. Definiciones básicas

Definición 33. Un *grafo* G es un par (V, E) donde V es un conjunto (cuyos elementos son llamados *vértices*) y E es otro conjunto (o *multiconjunto*), cuyos elementos son llamados *aristas*, y donde cada arista es i de la forma $e = \{x, y\}$ con $x, y \in V$.

Si $e = \{x, y\}$ con $x = y$ (o sea, si $e = \{x\}$) diremos que e es un *lazo* o *rizo* (en inglés, *loop*[1]).

Los elementos de una arista e son llamados sus *extremos*. Una arista se dice *incidente* en sus extremos.

Generalmente un grafo es dibujado en forma tal que cada vértice queda representado por un punto en el plano, y cada arista por una curva que une los representantes de sus extremos.

8.2.1. Multigrafos, grafos simples

Si consideramos un grafo $G = (V, E)$ tal que E es un multiconjunto (o sea, puede tener elementos repetidos), diremos que G es un *multigrafo*. Las aristas que tienen los mismos extremos son llamadas *aristas múltiples*.

[1] En general, despueés de dar el nombre de un concepto, trataremos de dar el nombre equivalente en inglés, a menos que éste sea obvio.

Si $G = (V, E)$ no tiene lazos ni aristas múltiples, diremos que es un grafo *simple*.

En un grafo simple, si los extremos de una arista e son u y v, anotamos $e = uv$ (o $e = vu$).

Desde ahora, supondremos que todos los grafos son simples, a menos que especifiquemos lo contrario.

8.2.2. El grafo nulo y los grafos triviales

El grafo $G = (\emptyset, \emptyset)$ es llamado el grafo nulo.

Comentario. Desde ahora, todos mención a un grafo supondrá implícitamente la condición de que dicho grafo no es nulo.

Un grafo no nulo G se dice *trivial* si no tiene aristas, es decir, si $G = (V, \emptyset)$, con $V \neq \emptyset$.

8.2.3. Grafos finitos

Definición 34. Un grafo G se dice *finito* si tanto $V(G)$ como $E(G)$ son finitos.

Convención: en este curso todos los grafos son finitos, salvo que explícitamente se indique lo contrario.

8.3. Adyacencia, grados, vértices aislados

Definición 35. Dos vértices u y v de un grafo son *adyacentes* (o cada uno es *vecino* del otro) sii ambos son extremos de una misma arista.

En este caso escribimos $u \leftrightarrow v$.

La cantidad de aristas incidentes a un vértice v es llamada el *grado* de v, y lo denotamos grado(v).

Nota: los lazos de la forma $\{v\}$ son contados como dos aristas al calcular grado(v).x

Teorema. En todo grafo,

$$\sum_{v \in V} \text{grado}(v) = 2\,|E|\,.$$

Teorema. En todo grafo, la cantidad de vértices de grado impar es par.

Definición 36. Un vértice que no tiene vecinos se dice *aislado*. Así, por ejemplo, un grafo es trivial si y sólo si todos sus vértices son aislados.

8.3.1. Matrices de adyacencia e incidencia

Definición 37. Sea $G = (V, E)$ un grafo sin lazos[2], y sean $V = \{v_1, \ldots, v_n\}$, $E = \{e_1, \ldots, e_m\}$ sus conjuntos de vértices y aristas.

La *matriz de adyacencia* de G es la matriz de $n \times n$ $A(G)$ donde a_{ij} es el número de aristas que tienen a v_i y a v_j por extremos.

La *matriz de incidencia* de G es la matriz de $n \times m$ $M(G)$ donde m_{ij} es 1 si v_i es un extremo de e_j, 0 si no.

Típicamente, en un programa un grafo es representado en memoria usando o su matriz de adyacencia o un arreglo de *listas de adyacencia* (donde cada vértice tiene una lista ligada con cada uno de los vértices vecinos).

En general, las matrices de incidencia no son usadas computacionalmente, pero sirven como ayuda conceptual.

[2]Pero posiblemente con aristas múltiples.

8.3.2. Complemento de un grafo. Cliques y conjuntos independientes.

- Dado un grafo simple $G = (V, E)$, su *complemento* es el grafo $\overline{G} = (V, E')$ donde $uv \in E'$ sii $uv \notin E$.
- Un *clique*[3] en un grafo es un conjunto de vértices mutuamente adyacentes.
- Un *conjunto independiente* (o *conjunto estable*) en un grafo es un conjunto de vértices mutuamente no adyacentes.

Note que $U \subseteq V$ es un conjunto independiente en G si y sólo si U es un clique en $\overline{G}$.

8.4. Subgrafos, subgrafos inducidos

Definición 38.

- Un *subgrafo* de un grafo $G = (V, E)$ es un grafo $H = (V', E')$ tal que $V' \subseteq V$ y $E' \subseteq E$.
  Si H es un subgrafo de G, decimos que "G *contiene* a H", y anotamos $H \subseteq G$.
- Si $H = (V', E')$ es un subgrafo de $G = (V, E)$ tal que $\forall x, y \in V'(\{x, y\} \in E' \leftrightarrow \{x, y\} \in E)$ (o sea, si H tiene *todas* las aristas que se forman en G con vértices de V') entonces se dice que H es el subgrafo de G *inducido* por V'.

8.5. Grafos conexos

Definición 39.

- Un grafo es *conexo* si cada par de vértices en G pertenece a un camino en G. Si G no es conexo se dice *disconexo*.
- Una *componente conexa* de $G = (V, E)$ es un subconjunto X de V tal que:
  - el subgrafo de G inducido por X es conexo, y
  - si $X' \subseteq V$ es tal que $X \subsetneq X'$ entonces el subgrafo de G inducido por X' no es conexo.

  O sea, una componente conexa de G es el conjunto de vértices de algún subgrafo conexo *maximal* de G.

8.6. Propiedades estructurales, isomorfismo

Las matrices de incidencia y adyacencia definidas anteriormente dependen del orden en que tomemos los vértices y las aristas; en otras palabras, dependen de los nombres que les demos a los vértices y aristas.

Nos interesa estudiar las *propiedades estructurales* de los grafos, i.e., aquellas que no cambian si cambiamos los nombres de sus vértices y sus aristas. El concepto central aquí es el de *isomorfismo*.

Definición 40. Un *isomorfismo* entre dos grafos simples $G = (V, E)$ y $H = (V', E')$ es una biyección $f : V \to V'$ tal que $uv \in E \leftrightarrow f(u)f(v) \in E'$.

Si existe un isomorfismo entre G y H decimos que ellos son *isomorfos* (lo que escribimos $G \cong H$).

Definición 41. Un isomorfismo entre un grafo simple y sí mismo es llamado un *automorfismo*.

[3]Posibles traducciones al castellano serían "banda" o "pandilla".

¿Cómo chequear isomorfismo?

Para mostrar isomorfismo entre dos grafos, generalmente es necesario dar explícitamente la biyección que preserva incidencia.

Par mostrar que dos grafos no son isomorfos, a veces es posible mostrar una propiedad estructural no compartida entre los dos grafos (¡o entre sus complementos!).

8.6.1. Clases de isomorfismo

Teorema. La relación de isomorfismo (entre los grafos simples) es una relación de equivalencia.

Las clases de equivalencia determinadas por la relación de isomorfismo son llamadas *clases de isomorfismo*.

Informalmente, las clases de isomorfismo corresponden a la idea de grafos con vértices "anónimos". Cuando se dibuja un miembro particular de la clase de isomorfismo (para enfatizar algún aspecto estructural), simplemente se está eligiendo un representante más conveniente de la clase, pero todavía se está discutendo el mismo "grafo con vértices anónimos".

Caminos y ciclos

Definición 42.

Un *camino* (path) es un grafo simple, cuyos vértices pueden ser ordenados de modo que dos vértices son adyacentes sii son consecutivos en la lista.

Para cada $n \geq 2$, el único camino[4] con n vértices es denominado P_n.

Definición 43. Un *ciclo* es un grafo con el mismo número de aristas que de vértices, cuyos vértices pueden ser puestos alrededor de un cí rculo de modo que dos vértices son adyacentes sii son aparecen consecutivamente a lo largo del círculo.

Para cada $n \geq 3$, el único ciclo[5] con n vértices es denominado C_n.

8.6.2. Algunas clases importantes

Le daremos nombres a los miembros de algunas clases de isomorfismo que aparecen comúnmente:

- El camino y el ciclo con n vértices son denotados por P_n y C_n respectivamente.
- Un *grafo completo* es un grafo simple cuyos vértices son todos adyacentes entre sí. Denotamos este grafo por K_n.
- Un *grafo bipartito completo* o *biclique* es un grafo simple bipartito donde dos vértices son adyacentes sii están en diferentes partes. Si los tamaños de las partes son r y s, denotamos este grafo por $K_{r,s}$.

Nota: cuando se menciona un grafo sin nombrar explícitamente sus vértices, en general nos referimos a su clase de isomorfismo.

8.7. Subgrafos

Técnicamente, decir que "H es un subgrafo de G" significa que algún subgrafo de G es isomorfo a H (también se dice que G contiene *una copia* de H).

Ejemplos.

Un camino *en un grafo* G es un subgrafo de G isomorfo a algún P_n.

Un ciclo *en un grafo* G es un subgrafo de G isomorfo a algún C_n.

Un grafo sin ciclos es llamado (¡sorpresa!) acíclico. Un grafo acíclico conexo es llamado un *árbol*.

[4]Salvo isomorfismo.

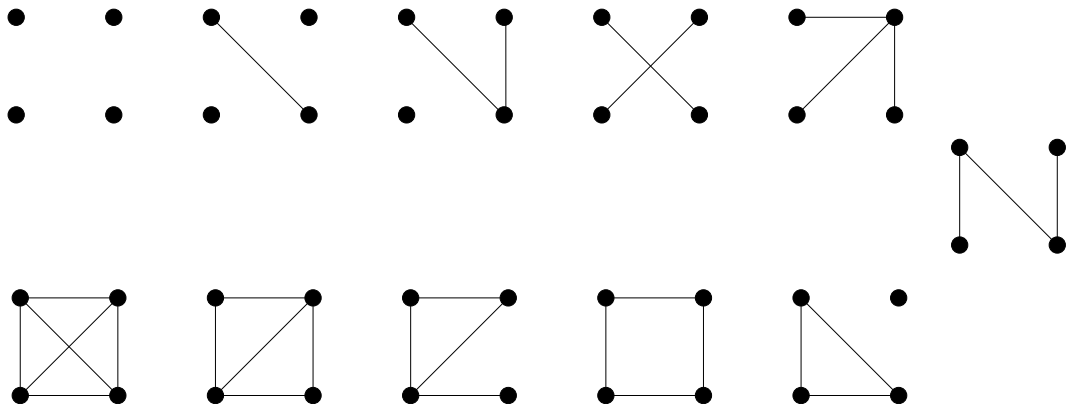
[5]Salvo isomorfismo.

Ejemplo. C_3 es subgrafo de K_5 pero no de $K_{2,3}$.

8.8. Los grafos con 4 vértices

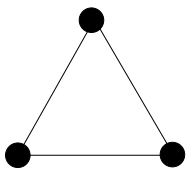
Hay $2^{\binom{n}{2}}$ grafos simples con n vértices. Así, con $n = 4$ vemos que hay 64 grafos simples con 4 vértices.

Estos grafos forman 11 clases de isomorfismos, de los cuales una (P_4) corresponde a grafos *auto-complementarios*, o sea, isomorfos a su propio complemento. La siguiente figura muestra estas 11 clases, .

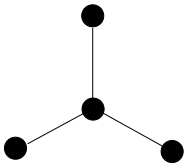


8.9. Otros grafos comunes

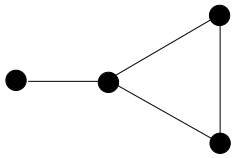
- el triángulo,



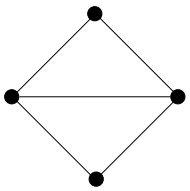
- la garra (claw),



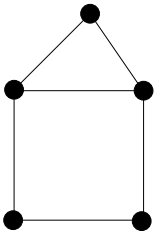
- la pata (paw),



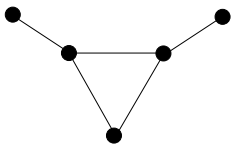
- el volantín (kite),



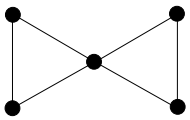
- la casa,



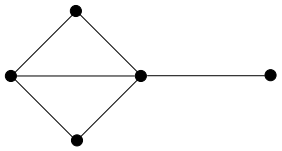
- el toro (bull),



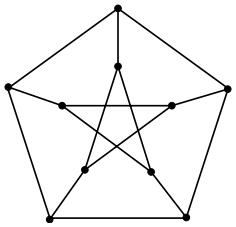
- la humita (bowtie),



- el dardo,



- el grafo de Petersen.



8.10. Grafos como modelos

Los grafos sirven para modelar diversas situaciones, por ejemplo:

- relaciones, como por ejemplo "*ser conocido de*";
- calificación de empleados para la realización de distintas tareas;
- programación de reuniones de comisiones del Senado (o de los exámenes en una universidad pequeña);
- coloración de mapas;
- rutas en una red de caminos;

8.10.1. Conocidos mutuos y desconocidos mutuos

Se puede demostrar (¡ejercicicio!) que en todo grupo de seis personas hay, o bien tres conocidos mutuos, o bien tres desconocidos mutuos.

Éste es un caso particular de lo que se conoce con el nombre de *Teorema de Ramsey*, que afirma lo siguiente:

Teorema (Ramsey). *Dados $k, l \in \mathbb{N}$, existe un natural n_0 tal que, dado cualquier $n \geq n_0$, todo grafo con n vértices tiene, o bien un clique de tamaño k, o bien un conjunto independiente de tamaño k.*

Demostración. Ejercicio. □

8.10.2. Asignación de tareas a distintos empleados

Se tiene un conjunto de tareas y un conjunto de empleados. Cada empleado puede realizar algunas (o incluso todas) las tareas.

¿Será posible asignar tareas a los empleados de modo que cada uno realice a lo más una tarea, y todas las tareas sean hechas por un empleado calificado?

Idea: Fórmese un grafo en que los empleados y las tareas son los vértices, y donde una arista une al empleado x con la tarea t sii el empleado x está calificado para desarrollar la tarea t.

Un concepto de teoría de grafos relacionado con esta situación es el siguiente:

Definición 44. Un grafo G es *bipartito* si V es la unión de dos conjuntos independientes (llamados las *partes* de G).

Así, el grafo formado por los empleados y las tareas es bipartito, con el conjunto de empleados como una parte y las tareas como la otra.

8.10.3. Reuniones de comisiones del Senado

Se desea programar las reuniones de varias comisiones del senado, de modo que dos comisiones que tienen un miembro en común no se reunan simultáneamente. Para simplificar, supongamos que cada comisión sesionará exactamente una hora.

¿Cuál es el número mínimo de períodos de una hora que deben ser usadas en la programación?

Idea: Esta situación puede ser modelada con un grafo en que los vértices son las comisiones, y las aristas unen las comisiones que comparten miembros.

La misma idea se aplica si se desea programar los exámenes de los cursos de una universidad pequeña, de modo que dos cursos no tengan exámenes simultáneos si tienen alumnos comunes.

Este ejemplo está relacionado con los concepto de *coloración* y *número cromático* de un grafo:

Definición 45.

- Una *coloración* de un grafo $G = (V, E)$ es una función $c : V \to \{1, \ldots, n\}$ (donde $n \in \mathbb{N}$). Si esta función es tal que $\forall x, y \in V(\{x, y\} \in E \to c(x) \neq c(y))$, decimos que esta coloración es *propia*. Los números $\{1, \ldots, n\}$ son llamados los "*colores*" de la coloración, y decimos que G ha sido "coloreado" o "pintado" con los colores $\{1, \ldots, n\}$.
- El *número cromático* $\chi(G)$ de un grafo G es qel menor número de colores que puede tener una coloración propia.

8.10.4. Grafos multipartitos y coloración

Definición 46. Dado $k > 1$, decimos que un grafo $G = (V, E)$ es *k-partito* si V es la unión de k conjuntos independientes.

Vemos que, si un grafo G es k-partito entonces puede ser pintado con k (o más) colores.

Coloración de mapas

Un caso particular del problema de coloración de grafos es el siguiente:

> ¿Cuántos colores se necesitan para pintar un mapa (dividido en *regiones* o *países*), de modo que no haya dos regiones con frontera común pintadas del mismo color?

Idea: Un mapa puede ser representado como un grafo, con las regiones como vértices y donde regiones adyacentes son las que comparten un trozo de frontera.

Los grafos correspondientes a los mapas satisfacen la siguiente definición:

Definición 47. Un grafo es *planar* si puede ser dibujado en el plano de modo que sus aristas no se crucen. Cada dibujo (con estas características) de un grafo es llamado *grafo plano*.

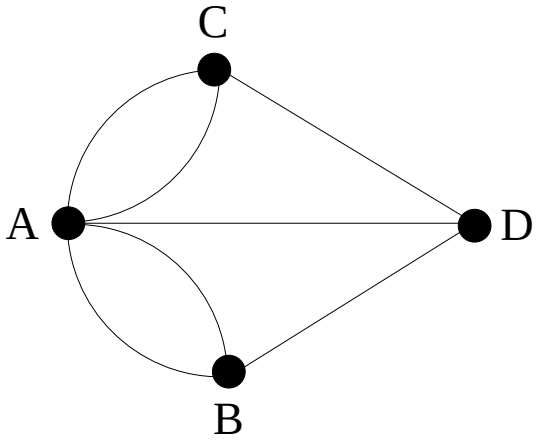
8.10.5. Rutas en una red de caminos

Un grafo puede ser usado para modelar caminos entre distintos puntos de una red caminera (o eléctrica, o hidráulica, etc.). Los vértices son las intersecciones de caminos y las aristas son los tramos de caminos entre intersecciones.

Un problema importante (y con mucha aplicación) en Teoría de Grafos es el de hallar el *camino más corto* entre dos puntos. Por ejemplo: en Santiago, ¿cuál es el camino más corto entre el Apumanque y el Museo Interactivo Mirador? Un sitio web que resuelve este problema es `http://www.mapcity.cl`.

8.11. Análisis del problema de Königsberg (Euler)

El problema de los puentes de Königsberg se puede expresar como sigue: ¿Tiene el grafo de la figura un camino (o circuito) que pase por cada arista *exactamente una vez*?



Un circuito con esta característica se llama *circuito Euleriano* (en honor a Leonard Euler).

Un grafo con un circuito Euleriano se dice *grafo Euleriano*.

Supongamos que podemos recorrer las aristas de un grafo (los puentes de Königsberg) en la forma pedida, y ordenemos el camino como sigue:

En cada vértice que no es ni el inicial ni el final, el camino debe *salir* una vez por cada vez que entra.

Así, el grado de cualquier vértice *excepto, posiblemente, los extremos del camino* debe ser par.

¿Qué pasa con los vértices extremos?

Si los extremos no coinciden (o sea, el camino no es un circuito) entonces la primera salida desde el vértice inicial no es compensada por ninguna entrada. Asimismo, la última entrada en el vértice final no es compensada por ninguna salida.

Así, si el camino no es un circuito, los vértices inicial y final deben tener grado impar.

Si el camino es un circuito, la primera salida y la última entrada se compensan mutuamente, y el vértice inicial/final tiene grado par, igual que los otros vértices.

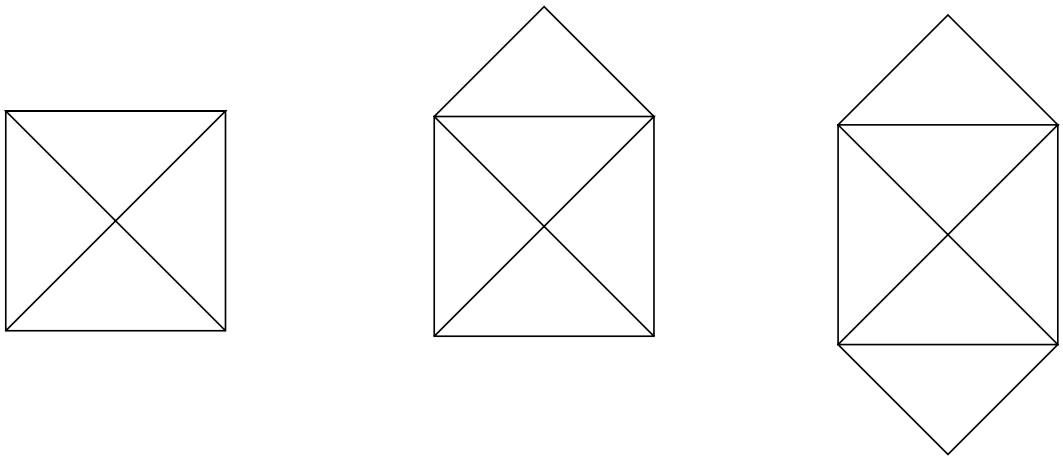
8.11.1. Análisis del problema (Resumen)

- Para que un grafo sea Euleriano, no puede haber más que dos vértices de grado impar en el grafo.
- Si el grafo tiene dos vértices de grado impar, todo camino Euleriano debe comenzar en uno de ellos y terminar en el otro.
- Si el grafo no tiene vértices de grado impar, todo camino Euleriano debe ser un circuito, y puede comenzar en cualquier vértice.

8.11.2. Dibujos sin levantar el lápiz

Una variante muy conocida del problema de determinar si un grafo es Euleriano es el de dibujar una figura de un solo trazo, o sea, sin levantar el lápiz del papel.

Ejemplo: ¿cuáles de las siguientes figuras pueden ser dibujadas con sólo un trazo?



Más aún: si no es posible dibujar una figura dada con un solo trazo, ¿cuántos trazos son necesarios?

Si en una figura hay $2n$ vértices de grado impar[6] entonces se necesitan exactamente n trazos para dibujarla.

8.12. Ciclos y caminos Hamiltonianos

Dado un grafo G, un ciclo de G que pasa por todos los vértices de G es llamado un *ciclo Hamiltoniano*.

[6]Se puede demostrar que el número de vértices de grado impar en un grafo siempre es par.

Análogamente, un camino de G que pasa por todos los vértices de G es llamado un *camino Hamiltoniano*.

Un grafo con un ciclo Hamiltoniano se dice Hamiltoniano.

Dado un grafo Hamiltoniano en el que cada arista tiene asociado un costo, el problema de hallar el ciclo Hamiltoniano de menor costo total es conocido como el *problema del vendedor viajero* (un problema clásico de optimización).

8.13. Grafos autocomplementarios

Definición 48. Un grafo es *autocomplementario* si es isomorfo a su propio complemento.

Teorema. Un grafo simple G de n vértices es auto-complementario sii K_n se descompone en dos copias de G.

Ejemplos de grafos autocomplementarios:

- C_5.
- P_4.

Ejercicio. Demuestre que si un grafo de n vértices es autocomplementario, entonces $n \cong 4 \pmod n$ o $n-1 \cong 4 \pmod n$.

Más aún: demuestre que para todo $n \in \mathbb{Z}^+$ tal que $n \cong 4 \pmod n$ o $n-1 \cong 4 \pmod n$, existe un grafo autocomplementario con n vértices.

8.14. Problemas computacionales relacionados con cliques y conjuntos independientes

Vimos que un *clique* en un grafo es un subgrafo isomorfo a algún grafo completo K_n, y que un conjunto independiente es un subgrafo isomorfo a algún grafo trivial $\overline{K_n}$.

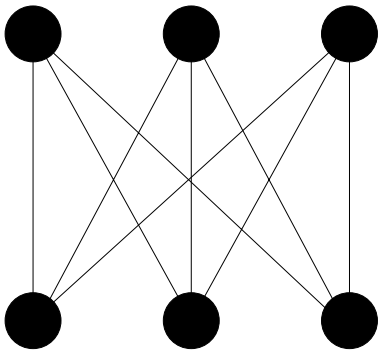
Los siguientes problemas son importantes desde el punto de vista computacional:

- Dado un grafo G y un entero positivo n, ¿tiene G un clique (o un conjunto independiente) de tamaño n?
- Dado un grafo G, ¿cuál es el máximo tamaño posible de un clique (o conjunto independiente) en G?
- Dado un grafo G, hallar un clique (o conjunto independiente) de tamaño máximo posible en G.
- Dado un grafo G, hallar *todos* los cliques (o conjuntos independientes) de tamaño máximo posible en G.

8.15. Planaridad

¿Puede un grafo ser dibujado en el plano, sin que se crucen las aristas?

Ejemplo. ¿Es posible unir tres casas a las distribuidoras de TV Cable, teléfono e Internet *sin que se crucen los cables*?



8.16. La característica de Euler

Teorema (Euler, 1758). *Si G es un grafo plano conexo con n vértices, e aristas y f caras, entonces $n - e + f = 2$.*

Demostración. Por inducción en n. □

8.16.1. Comentarios

Las siguientes son algunas consecuencias de la fórmula de Euler:

- Debido a la fórmula de Euler, todas las representaciones planas de un grafo G tienen el mismo número de caras.
- Si G es un grafo plano disconexo, la fórmula de Euler tal como está no es válida. Si G tiene k componentes, entonces hay que agregar $k-1$ aristas para conectarlas todas y aplicar la fórmula original. Pero el agregar estas $k-1$ aristas no cambia la cantidad de caras en G, por lo que en este caso $n - e + f = k + 1$.

Una consecuencia particularmente útil es la siguiente:

Teorema. Si G es un grafo plano que además es simple y tiene más de dos vértices, entonces $e(G) \leq 3n(G) - 6$. Si además G no tiene triángulos, entonces $e(G) \leq 2n(G) - 4$.

Demostración. Basta considerar el caso en que G es conexo (si no lo es, agréguense aristas hasta que lo sea; el nuevo número de aristas satisface la cota y por lo tanto el original también).

Sea G simple y con al menos 3 vértices, y sean $n = n(G)$, $e = e(G)$ y $f = f(G)$.

Por ser G es simple y tener al menos 3 vértices, cada cara tiene largo ≥ 3, de donde

$$2e = \sum_{i=1}^{f} l(F_i) \geq 3f = 3(e - n + 2),$$

de donde $e \leq 3n - 6$.

Análogamente, si G no tiene triángulos, $l(F_i) \geq 4$, de donde $2e \geq 4(e-n+2)$, o sea $e \leq 2n-4$.

Ejemplos. La no planaridad de K_5 y de $K_{3,3}$ puede ser demostrada usando el teorema recién visto: para K_5, $e = 10 > 9 = 3n - 6$. Como $K_{3,3}$ no tiene triángulos y $e = 9 > 8 = 2n - 4$, estos grafos tienen demasiadas aristas para ser planares.

8.17. Ejercicios

1. Sea G un grafo simple. Demuestre que, si G es disconexo, entonces $\bar{G}$ es conexo.

2. El *diámetro* de un grafo es la máxima distancia (medida en número de aristas) entre sus vértices.

   Demuestre que, si el diámetro de G es ≥ 4 entonces el diámetro de $\bar{G}$ es ≤ 2.

3. Demuestre que un grafo G es bipartito si y sólo si G no tiene ciclos de largo impar.

4. La *cintura* de un grafo es el tamaño del menor ciclo inducido (o sea, del menor ciclo sin diagonales).

   Sea G un grafo con cintura 5. Demuestre que si todo vértice de G tiene grado $\geq k$, entonces G tiene por lo menos $k^2 + 1$ vértices. Para $k = 2$ y $k = 3$, encuentre un grafo de cintura 5 y exactamente $k^2 + 1$ vértices.

5. Un vértice de un grafo es *aislado* si no tiene vecinos. Un vértice se dice *de corte* si su eliminación aumentaría la cantidad de componentes conexas del grafo.

   Demuestre o refute la siguiente afirmación:

   > Si un grafo simple con diámetro 2 tiene un vértice de corte, entonces su complemento tiene un vértice aislado.

6. Demuestre, o refute dando un contraejemplo, la siguiente afirmación:

   > Si G es un grafo Euleriano en que las aristas e y f tienen un extremo común, entonces G tiene un circuito Euleriano en que e y f aparecen en forma consecutiva.

7. Ordene 7 ceros y 7 unos en forma cíclica de modo que las 14 secuencias de 4 bits consecutivos que se forman sean todos las secuencias binarias de largo 4, excepto por 0101 y 1010.

   Ayuda: Forme un grafo y construya un ciclo Euleriano.

8. Demuestre que el grafo de Petersen no tiene ciclos de largo 10.

   Ayuda: suponga que existe un ciclo de largo 10, y use las propiedades del grafo de Petersen para obtener una contradicción.

9. Demuestre, o dé un contraejemplo, para cada una de las siguientes afirmaciones:

   a) Todo grafo Euleriano bipartito tiene un número par de aristas.

   b) Todo grafo Euleriano simple con un número par de vértices tiene un número par de aristas.

   c) Si G es un grafo Euleriano en que las aristas e y f tienen un extremo común, entonces G tiene un circuito Euleriano en que e y f aparecen en forma consecutiva.

10. Si un grafo no es conexo, ¿qué forma tendrá su matriz de adyacencia?

11. Muestre todas las posibles matrices de adyacencia para P_3 (un *camino* con 3 vértices).

12. Escriba una matriz de incidencia para P_6 (un camino con 6 vértices) y para C_6 (un ciclo con 6 vértices).

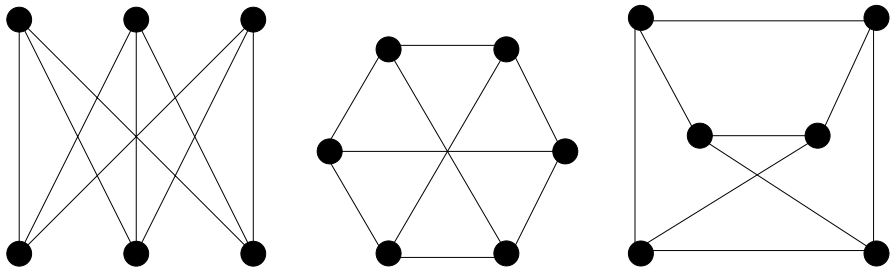
13. Demuestre que la relación $\cong$ (definida como "$G_1 \cong G_2$ sii G_1 es isomorfo a G_2") es una relación de equivalencia entre grafos.

14. Sean G_1 y G_2 dos grafos. Demuestre que $G_1 \cong G_2$ sii $\overline{G_1} \cong \overline{G_2}$.

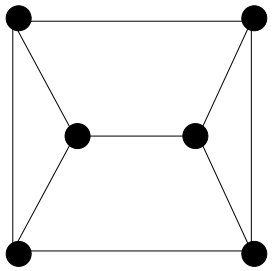
15. Demuestre que $C_5 \cong \overline{C_5}$.

16. Demuestre que $P_4 \cong \overline{P_4}$.

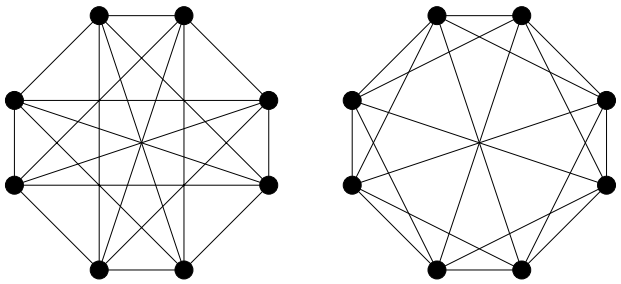
17. Demuestre que los siguientes grafos son isomorfos (de hecho, todos son isomorfos a $K_{3,3}$):



18. Demuestre que el siguiente grafo no es isomorfo a $K_{3,3}$:



19. Demuestre que los siguientes grafos no son isomorfos:



Ayuda: Considere los complementos de los grafos dados. ¿Qué propiedad es satisfecha por sólo uno de los complementos?

20. Sea G un grafo simple, conexo, no completo (o sea, $G \ncong K_n$ para ningún n). Demuestre que todo vértice de G pertenece a algún subgrafo inducido de G con 3 vértices que es isomorfo a P_3.

21. Demuestre que K_n tiene $n!$ automorfismos, y que $K_{m,n}$ tiene $m!n!$ automorfismos si $m \neq n$, y $2(n!)$ si $m = n$.

22. ¿Cuántos automorfismos tiene P_n? ¿Cuántos automorfismos tiene C_n?

23. Demuestre o refute mediante un contraejemplo: si G es un grafo finito simple donde todo vértice tiene grado 2, entonces G es un ciclo.

24. Demuestre que un grafo es bipartito si y sólo si su conjunto de vértices puede ser partido en dos conjuntos independientes.

25. Demuestre que un grafo es bipartito si y sólo si es 2-coloreable.

26. Demuestre que un grafo es bipartito completo si y sólo si es bipartito y la adición de cualquier arista haría que dejara de serlo.

27. Demuestre que $\overline{K_{m,n}} \cong K_m + K_n$.

28. Dado un grafo $G = (V, E)$, la identidad es un automorfismo de G, pero no necesariamente el único. Demuestre que el conjunto $Aut(G)$ de automorfismos de G forma un grupo con la composición.

29. Sea $G = (V, E)$ un grafo. Demuestre que la relación $\rightsquigarrow$ definida en V por "$x \rightsquigarrow y \leftrightarrow$ hay un camino en G entre x e y" es una relación de equivalencia en V.

    Nota: las clases de equivalencia definidas en V por $\rightsquigarrow$ son llamadas *componentes conexas* de G.

30. Demuestre que las componentes conexas de G son subconjuntos maximales de vértices que inducen subgrafos conexos (en otras palabras, $S \subseteq V$ es una componente conexa sii S induce un subgrafo conexo y todo S' tal que $S \subseteq S' \subseteq V$ induce un subgrafo disconexo).

31. Demuestre que las siguientes propiedades son invariantes bajo isomorfismos (o sea, si $G \cong G'$ entonces G' tiene la propiedad sii G la tiene):

    a) ser conexo,

    b) ser bipartito,

    c) ser completo,

    d) ser k-coloreable (para $k \in \mathbb{N}$).

32. Demuestre que las siguientes funciones son invariantes bajo isomorfismos (o sea, si $G \cong G'$ entonces $f(G') = f(G)$ para cada una de las siguientes funciones f):

    a) $\chi(G)$,

    b) $\kappa(G)$.

33. Demuestre el Teorema de Euler: dada una inmersión plana de $G = (V, E)$, y siendo R el conjunto de las regiones determinadas por ella en el plano, se tiene que $|V| - |E| + |R| = 2$.

Capítulo 9

Complejidad de algoritmos y problemas

9.1. Complejidad de un algoritmo y de un problema

Problemas e instancias

Cuando hablamos de "resolver un problema" no nos referimos a, por ejemplo, "decidir si en la lista de números $\{12, 18, 28, 45, 52, 61, 73, 84, 89, 97, 104\}$ hay dos números cuya suma también sea elemento de la lista".

A esto lo llamamos "resolver una instancia" de un problema. Por un problema, entendemos todas las posibles instancias, más una "pregunta" asociada.

Por *resolver un problema* nos referimos a encontrar un *algoritmo* que, dada una instancia del problema, conteste correctamente la pregunta asociada. La forma de escribir un algoritmo de modo que un computador pueda ejecutarlos es un *programa*.

Un mismo problema puede ser resuelto de distintas maneras (pueden utilizarse distintos algoritmos para resolverlo). Algunas son mejores que otras ...

Ejemplo. Tome una guía telefónica de Santiago y busque a todos los Dissett en ella. Probablemente Ud. puede pensar en una manera más astuta de resolver este problema que comenzar por Aaby, Aarón, Aazos, Abalón, Abarca, Abaroa, Abarzúa, ...

Eficiencia: tiempo *vs* "tamaño de la instancia"

Tomemos como ejemplo el problema de decidir, dada una lista de números enteros, si es posible dividirla en dos sub-listas tales que las sumas de ambas sublistas sean iguales.

Suponga que tiene dos programas que resuelven este problema, y que usa uno de ellos para resolver una instancia A y el otro para resolver una instancia B.

El primer algoritmo tomó 1 minuto en encontrar la respuesta óptima para A, y el segundo demoró 12 horas en encontrar la respuesta óptima para B.

¿Qué algoritmo es más "eficiente"?

Nos falta una parte crucial de la información para responder esta pregunta:

¿Cuáles son los *tamaños* de las instancias A y B que resuelven estos algoritmos?

Más aún: aunque conociéramos la respuesta a esta pregunta, puede ser difícil decidir "cuál de los dos algoritmos es más eficiente". Incluso si ambos tamaños coinciden (o si ambos algoritmos fueron utilizados separadamente para resolver *la misma instancia*), decir cuál algoritmo es "más eficiente" requiere más de un punto de comparación.

Complejidad de un algoritmo

Sea $\mathcal{A}$ un algoritmo que resuelve un problema dado π. Nos interesa estudiar la función $T_{\mathcal{A}} : \mathbb{N} \to \mathbb{R}_0^+$ que indica el tiempo máximo que puede tomar $\mathcal{A}$ en resolver instancias de tamaño n.

En realidad, el "tiempo" que tome resolver una instancia de π depende de demasiados factores, por ejemplo la arquitectura de la máquina en que ejecutamos el programa (mi otro computador es un Athlon XP 64 con 8 CPUs de 4 GHz cada una :-P)

Así que en realidad no nos interesa medir el tiempo exacto, sino una "aproximación".

Por ejemplo, nos puede interesar medir cuántas operaciones se realizan al resolver una instancia dada.

O cuántas operaciones de algún tipo dado (típicamente, las más costosas en tiempo). Por ejemplo, al buscar datos en una base de datos, da lo mismo si se hacen 100 o 1000 o 10000 cálculos usando datos de la RAM ... una sola lectura desde el disco duro toma más tiempo que miles de cálculos hechos por la CPU.

Es importante elegir bien qué se va a contar. Esto permite aproximar —de alguna manera, y salvo por algún factor constante por determinar— cuánto tiempo toma el algoritmo en resolver una determinada instancia.

Complejidad de un problema

Dado un problema π, llamamos la *complejidad* de π a la función $T_\pi : \mathbb{N} \to \mathbb{R}_0^+$ dada por

$$T_\pi(n) = \text{mín}\left\{T_{\mathcal{A}}(n) : \mathcal{A} \text{ es un algoritmo que resuelve } \pi\right\}.$$

Ejemplo. Se puede demostrar (típicamente visto, por ejemplo, en un curso de Estructuras de Datos) que, dado cualquier algoritmo de ordenación basado en comparaciones, su complejidad es *por lo menos* $c \cdot n \log n$ para alguna constante $c > 0$.

Así, como existen algoritmos para ordenar basados en comparaciones con complejidad $\leq C \cdot n \log n$ con C constante (por ejemplo, *Heapsort* o *Mergesort*), decimos que el problema de ordenar un arreglo en base a comparaciones tiene complejidad "esencialmente $n \log n$ (salvo por algún factor constante)".

9.2. Notación asintótica

La notación "O"

Ya que los factores constantes "casi no nos interesan" al estudiar complejidad, queremos reflejar este hecho en las funciones que representan complejidad de problemas y algoritmos. De hecho, esto es útil no sólo al estudiar el tiempo utilizado por un algoritmo, sino también al considerar otros recursos, por ejemplo, memoria o acceso a algún dispositivo específico.

Una notación útil para esto es la llamada notación "O grande".

Definición 49. Sea $f : \mathbb{N} \to \mathbb{R}_o^+$ una función cualquiera. Definimos $O(f(n))$ como el *conjunto*[1]

$$\left\{t : \mathbb{N} \to \mathbb{R}_0^+ \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N})(\forall n \in \mathbb{N})(n \geq n_0 \to t(n) \leq c \cdot f(n))\right\}.$$

Al valor $n_0 \in \mathbb{N}$ lo llamaremos el *umbral*.

Ejemplo. Veamos que $3n^2 + 5n + 6 \in O(n^2)$. En efecto, vemos que, si $n \geq 6$, tenemos

$$3n^2 + 5n + 6 \leq 3n^2 + n \cdot n + n \cdot n = 5n^2,$$

por lo que tomando $c = 5$ y $n_0 = 6$ se satisface la definición de $O(n^2)$.

Pero también podemos satisfacerla tomando $c = 14$, $n_0 = 1$. O también $c = 7$, $n_0 = 2$.

Ejercicio. Demuestre que siempre es posible usar un umbral $n_0 = 1$.

[1] Note que, por ejemplo, decimos $g(n) \in O(f(n))$, no $g(n) = O(f(n))$.

Ejercicio. Demuestre que, dados a y b dos números reales cualesquiera mayores que 1, y $f : \mathbb{N} \to \mathbb{R}_0^+$ una función cualquiera, se tiene:

$$O(f(n) \log_a n) = O(f(n) \log_b n)$$

(y lo mismo vale para Ω y Θ).

Así, cuando tengamos una función $g(n) \in O(f(n) \log_a n)$ escribiremos simplemente $g(n) \in O(f(n) \log n)$, ya que la base es irrelevante.

Las notaciones Ω y Θ

Sea $f : \mathbb{N} \to \mathbb{R}_o^+$ una función cualquiera. Definimos $\Omega(f(n))$ como el conjunto

$$\left\{t : \mathbb{N} \to \mathbb{R}_0^+ \mid (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N})(\forall n \in \mathbb{N})(n \geq n_0 \to t(n) \geq c \cdot f(n))\right\}.$$

Finalmente, definimos $\Theta(f(n))$ como:

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

Ejercicio. Demuestre que $\Theta(f(n))$ es el conjunto de todas las funciones $t : \mathbb{N} \to \mathbb{R}_0^+$ tales que

$$(\exists c_1, c_2 \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N})(\forall n \in \mathbb{N})(n \geq n_0 \to c_1 f(n) \leq t(n) \leq c_2 f(n)).$$

Generalmente, no nos interesa encontrar la función específica sino hallar una función "simple" $f(n)$ tal que $T_{\mathcal{A}}(n) \in O(f(n))$ (o, de preferencia, tal que $T_{\mathcal{A}}(n) \in \Theta(f(n))$).

Por "función simple" entendemos una función "elemental" (potencia, logaritmo, parte entera) o que sea expresable como combinación de una pequeña cantidad de éstas.

9.3. Complejidad de algoritmos iterativos

FALTA ...

9.4. Complejidad de algoritmos recursivos

En esta sección estudiaremos la complejidad de algoritmos recursivos. Típicamente, un algoritmo recursivo se "llama a sí mismo" con instancias de tamaño menor, excepto cuando el tamaño de la instancia es menor que un cierto valor límite (el *caso base* de la recursión).

Así, la complejidad de un algoritmo recursivo satisface una *ecuación de recurrencia* o *inecuación de recurrencia*, en que $T(n)$ (el *tiempo* o la cantidad de operaciones que toma resolver una instancia de tamaño n) está dado en términos de los valores de $T(k)$ para valores de $k < n$.

Ejemplo (la complejidad de *mergesort*). Nos interesa estudiar el tiempo que toma ordenar un archivo usando OM (ordenamiento por mezcla).

Sea

$$\begin{aligned} T_{\text{OM}}(n) \quad := \quad & \text{tiempo de ejecución del algoritmo } OM \\ & \text{con un archivo de largo } n \text{ (en el peor caso).} \end{aligned}$$

No tenemos una representación explícita para la función de complejidad, pero sí dos restricciones que debe satisfacer:

$$T_{\text{OM}}(1) \leq c \tag{9.1}$$

$$T_{\text{OM}}(n) \leq T_{\text{OM}}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T_{\text{OM}}\left(\left\lceil \frac{n}{2} \right\rceil\right) + T_M(n), \quad n \geq 2, \tag{9.2}$$

donde $T_M(n)$ es el tiempo que toma, en el peor caso, el algoritmo de mezcla M en mezclar dos archivos de largos que sumen n, y c es una constante que sirve de cota para el tiempo que *mergesort* se demora en ordenar una un archivo de largo 1.

En la inecuación anterior, podemos suponer que $T_M(n) \leq dn$, donde $d \geq 0$ es *alguna* constante.

Así, nuestra "inecuación de recurrencia" queda

$$T_{\text{OM}}(1) \leq c \tag{9.3}$$

$$T_{\text{OM}}(n) \leq T_{\text{OM}}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T_{\text{OM}}\left(\left\lceil \frac{n}{2} \right\rceil\right) + dn, \quad n \geq 2, \tag{9.4}$$

Es imposible resolver exactamente esta ecuación, pero sí es posible demostrar que la solución de la siguiente ecuación de recurrencia es una cota superior para $T_{\text{OM}}()$:

$$T(1) = c \tag{9.5}$$

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + dn, \quad n \geq 2, \tag{9.6}$$

Tenemos una ecuación de recurrencia, pero no del tipo que asocia n con $(n+1)$ ...

Queremos hallar cotas para la solución de esta ecuación.

Conjetura: existe alguna constante $a \geq 0$ tal que, para todo $n \geq 1$:

$$T(n) \leq a \cdot n \cdot \lceil \log_2(n) \rceil .$$

Inducción constructiva

Si tratamos de demostrar nuestra conjetura por inducción, encontraremos que la hipótesis de inducción no nos permite demostrar el paso inductivo como debiera, por lo que intentamos replantearla como sigue:

Conjetura': existen algunas constantes $a \geq 0$ y $b \in \mathbb{R}$ tales que, para todo $n \geq 1$:

$$T(n) \leq a \cdot n \cdot \lceil \log_2(n) \rceil + bn.$$

Demostración. El proceso de demostración de la conjetura es relativamente largo. Partiremos demostrando que la inducción es cierta (y determinando las constantes a y b) si n es una potencia de 2.

O sea, partimos demostrando que, si $n = 2^k$ entonces

$$T(n) \leq a \cdot n \cdot k + bn.$$

Esto puede hacerse por PSI o por PICV.

Concluimos que debe tenerse $b \geq t_o$, $a \geq c$ (por lo que podemos escoger $a = c$, $b = t_o$). O sea, podemos probar por PSI o por PICV que, si n es una potencia de 2, entonces

$$T(n) \leq cn \lceil \log_2(n) \rceil + t_o n.$$

El siguiente paso es demostrar que $T(n)$ es una función que no decrece, o sea, que $T(n) \leq T(n+1)$ para todo $n \geq 1$ (esto también se hace por PICV).

Finalmente, si n no es una potencia de 2, usamos el hecho de que

$$n \leq 2^{\lceil \log_2(n) \rceil} \leq 2^{\lfloor \log_2(n)+1 \rfloor} \leq 2 \cdot 2^{\lfloor \log_2(n) \rfloor}$$

para establecer la cota: como $T(\cdot)$ es no-decreciente,

$$\begin{aligned} T(n) &\leq T\left(2 \cdot 2^{\lfloor \log_2(n) \rfloor}\right) \\ &\leq 2 \cdot c \cdot 2^{\lfloor \log_2(n) \rfloor} \lceil \lfloor \log_2(n) \rfloor + 1 \rceil + 2t_o \cdot 2^{\lfloor \log_2(n) \rfloor} \\ &= 2 \cdot 2^{\lfloor \log_2(n) \rfloor} (c \cdot \lfloor \log_2(n) \rfloor + 1 + t_o) \\ &\leq (2c) n \log_2(n) + 2(1 + t_o) n. \end{aligned}$$

□

Hemos probado que $T_{\text{OM}}(n) \in O(n \log_2(n))$.

Ejercicio. Adapte la misma demostración para demostrar que $T(n) \in \Theta(n \log_2(n))$. Deduzca que $T_{\text{OM}}(n) \in \Theta(n \log_2(n))$.

Complejidad de algoritmos "*divide et regna*"

Algoritmos como *mergesort*, búsqueda binaria, y otros similares, están basados en el principio de estrategia llamado "*divide et regna*" (divide y reina, divide y vencerás, dividir para conquistar, etc.).

Estos algoritmos trabajan sobre una instancia de tamaño $n \geq n_0$ (digamos, formada por n valores), dividiéndola en varias subinstancias de tamaño menor, y resolviendo recursivamente algunas de (o todas) estas subinstancias. El valor n_0 marca el umbral a partir del cual se realizan estas llamadas recursivas; para valores de $n < n_0$ no se realizan llamadas recursivas.

En particular, en la mayoría de los casos los tamaños menores son obtenidos dividiendo n por una constante b y aproximando a un entero (mediante $\lfloor\ \rfloor$ o $\lceil\ \rceil$).

Supongamos que, ante una instancia de tamaño n, el algoritmo realiza a_1 llamadas recursivas con sub-instancias de tamaño $\lfloor \frac{n}{b} \rfloor$ y a_2 llamadas recursivas con sub-instancias de tamaño $\lceil \frac{n}{b} \rceil$.

Además de estas $a_1 + a_2$ llamadas recursivas, el algoritmo debe realizar un cierto "procesamiento adicional" de los valores que forman la instancia. Este procesamiento puede ser hecho antes de las llamadas recursivas (por ejemplo, en *quicksort*), después de dichas llamadas (por ejemplo, en *mergesort*), o incluso parte antes y parte después (veremos ejemplos más adelante).

Así, si llamamos $f(n)$ al tiempo adicional que se necesita para este procesamiento adicional, la ecuación de recurrencia que expresa la complejidad de este algoritmo será

$$T(n) = \begin{cases} c_0 & \text{si } 0 \leq n < n_0, \\ a_1 T\left(\lfloor \frac{n}{b} \rfloor\right) + a_2 T(\lceil \frac{n}{b} \rceil) + f(n) & \text{si } n \geq n_0. \end{cases}$$

De la relación entre las constantes a_1, a_2, b y la función $f(n)$ dependerá el orden de magnitud de la función de complejidad $T(n)$. En particular, en el caso en que $f(n)$ es un polinomio, podemos enunciar el siguiente teorema:

Teorema (el "teorema maestro"de los algoritmos *divide et regna*). *Si a_1, a_2, b, c y d son constantes positivas, y $T(n)$ satisface la ecuación de recurrencia*

$$T(n) = \begin{cases} c_0 & \textit{si } 0 \leq n < n_0, \\ a_1 T\left(\lfloor \frac{n}{b} \rfloor\right) + a_2 T(\lceil \frac{n}{b} \rceil) + cn^d & \textit{si } n \geq n_0, \end{cases}$$

entonces (llamando $a = a_1 + a_2$), se tiene:

$$T(n) \in \begin{cases} \Theta(n^d) & \textit{si } a < b^d, \\ \Theta(n^d \log n) & \textit{si } a = b^d, \\ \Theta(n^{\log_b a}) & \textit{si } a > b^d. \end{cases}$$

¿Qué condiciones debe satisfacer n_0 en la ecuación de recurrencia anterior?

Demostraremos que debe tenerse $n_0 \geq \frac{b}{b-1}$.

En efecto:

Para que la recurrencia corresponda a una función correctamente definida, las invocaciones recursivas de la función deben tener por argumentos a números *estrictamente menores que* n.

Visto desde el punto de vista algorítmico, si la recurrencia corresponde a un algoritmo recursivo correcto, es necesario que ninguna llamada recursiva (cuando se está resolviendo una instancia de tamaño n) se haga sobre una instancia de tamaño $\geq n$.

O sea, se requiere que tanto $\lfloor \frac{n}{b} \rfloor < n$ como $\lceil \frac{n}{b} \rceil < n$. Como $\lfloor \frac{n}{b} \rfloor \leq \lceil \frac{n}{b} \rceil$, basta exigir que $\lceil \frac{n}{b} \rceil < n$. Pero se sabe que $\lceil \frac{n}{b} \rceil < \frac{n}{b} + 1$, por lo que basta que $\frac{n}{b} + 1 \leq n$ para que $\lceil \frac{n}{b} \rceil < n$.

Así, exigimos que $\frac{n}{b} + 1 \leq n$. Pero esto es equivalente a exigir que $n + b \leq nb$, o lo que es lo mismo, que $nb - n \geq b$, de donde se deduce la condición $n \geq \frac{b}{b-1}$.

Capítulo 10

P y NP

10.1. Algoritmos eficientes: polinomial *vs* exponencial

Nos interesa estudiar qué problemas pueden se resueltos eficientemente. En términos teóricos, decimos que un problema puede ser resuelto eficientemente si existe un algoritmo $\mathcal{A}$ que lo resuelve *en tiempo polinomial*, o sea, si existe un polinomio $p(n)$ tal que $T_{\mathcal{A}}(n) \in O(p(n))$.

La razón para esto es la siguiente: un algoritmo con complejidad $\in O(p(n))$ (donde $p(n)$ es un polinomio en n) crece más lentamente que una función de la forma c^n con $c > 1$, o que otras funciones que aparecen al estudiar complejidad.

Ejemplo. Suponga que tiene dos algoritmos para el mismo problema, uno con complejidad $\approx 2^n$ (exponencial) y el otro con complejidad $\approx 1000n^3$ (cúbico).

1. ¿Para qué valores de n conviene usar uno u otro algoritmo?

R. Para $n \leq 23$, el algoritmo de complejidad exponencial es más rápido. Pero a partir de allí, la diferencia es dramática ...

2. Suponga ahora que tiene un computador que hace 1 millón de operaciones por segundo. ¿Cuál es el máximo tamaño de instancia que se puede resolver en 1 hora, con cada uno de los algoritmos?

R. El algoritmo exponencial puede resolver en 1 hora instancias de tamaño ≤ 31. El cúbico, en el mismo tiempo, puede resolver instancias d e tamaño ≤ 153.

3. Finalmente, suponga que hace un *upgrade* a su computador, y en lugar de ser capaz de hacer 1 millón de operaciones por segundo ahora es capaz de hacer 10 millones de operaciones por segundo. ¿Cómo aumentan los máximos tamaños de instancia que se pueden resolver en 1 hora, con cada uno de los algoritmos?

R. Para el exponencial, los 31 suben a 35. Para el cúbico, los 153 suben a 330.

El proverbial grano de sal

Note que, en la práctica, el preferir algoritmos polinomiales es relativo: un algoritmo exponencial con complejidad $T(n) = 3 \cdot 1{,}001^n$ será, en la práctica, mucho mejor que un algoritmo de complejidad $T(n) = 100000000000n^3$ (constante demasiado grande) o $T(n) = 2n^{40}$ (que aunque tiene una constante pequeña, tiene un exponente muy grande).

Pero como regla general, consideraremos que un problema es "soluble eficientemente" si puede ser resuelto en tiempo polinomial.

Ejercicio. ¿Exactamente, para qué valores resulta mejor el algoritmo exponencial que los dos algoritmos polinomiales mencionados?

10.2. Tipos de problemas

Los problemas computacionales pueden dividirse en varios tipos. Cuatro tipos importantes de problemas son:

- problemas de decisión;
- problemas de búsqueda;
- problemas de evaluación;
- problemas de optimización.

Ilustraremos cada uno de estos tipos de problemas mostrando una variante del llamado *problema de programación entera*, de gran importancia práctica.

Problemas de decisión Formalmente, un problema de decisión está formado por un dominio (conjunto de posibles instancias) D, y un subconjunto L de D, llamado *lenguaje* o *problema*.

Se busca un algoritmo $\mathcal{A}$ que, *aplicado a cualquier* $w \in D$, responde:

- **SI** si $w \in L$,
- **NO** si $w \in D - L$.

Ejemplo: Dada una matriz A de $m \times n$ con valores enteros, y un vector $\overrightarrow{\mathbf{b}} \in \mathbb{Z}^m$, ¿existe una solución $\overrightarrow{\mathbf{x}} \in \mathbb{Z}^n$ del sistema de inecuaciones $A\overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}$, $\overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{0}}$?

O sea, ¿existe una solución factible de la instancia de Programación Entera para estas restricciones?

Problemas de búsqueda Formalmente, un problema de búsqueda está formado por un dominio (conjunto de posibles instancias) D, y, para cada instancia I un conjunto $F(I)$ de *soluciones factibles*.

Se busca un algoritmo $\mathcal{A}$ que, *aplicado a cualquier* $I \in D$, indique si $F(I) = \emptyset$ o, en caso contrario, entregue un elemento de $F(I)$.

Ejemplo: Dada una matriz A de $m \times n$ con valores enteros, y un vector $\overrightarrow{\mathbf{b}} \in \mathbb{Z}^m$, indique si existe una solución $\overrightarrow{\mathbf{x}} \in \mathbb{Z}^n$ del sistema de inecuaciones $A\overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}$, $\overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{0}}$, o en caso contrario advierta que $F(I) = \emptyset$.

Problemas de evaluación Formalmente, un problema de evaluación está formado por un dominio (conjunto de posibles instancias) D, donde cada instancia I está formada por un conjunto $F(I)$ de *soluciones factibles* y una *función de costo* $c : F(I) \to \mathbb{Z}$.

Se busca un algoritmo $\mathcal{A}$ que, *aplicado a cualquier* $I \in D$, indique si $F(I) = \emptyset$ o, en caso contrario, entregue el valor de

$$\text{mín}\left\{c(x) : x \in F(I)\right\}.$$

Ejemplo: Dada una matriz A de $m \times n$ con valores enteros, un vector $\overrightarrow{\mathbf{b}} \in \mathbb{Z}^m$, y un vector $\overrightarrow{\mathbf{c}} \in \mathbb{Z}^n$, indique si existe una solución $\overrightarrow{\mathbf{x}} \in \mathbb{Z}^n$ del sistema de inecuaciones $A\overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}$, $\overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{0}}$, y de ser así, indique el valor de

$$\text{mín}\left\{\overrightarrow{\mathbf{c}}^T\overrightarrow{\mathbf{x}} : A\overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}, \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{0}}\right\}.$$

Problemas de optimización Formalmente, un problema de optimización está formado por un dominio (conjunto de posibles instancias) D, donde cada instancia I está formada por un conjunto $F(I)$ de *soluciones factibles* y una *función de costo* $c : F(I) \to \mathbb{Z}$.

Se busca un algoritmo $\mathcal{A}$ que, *aplicado a cualquier* $I \in D$, entregue algún $x \in F(I)$ que minimice el valor de $c(x)$.

Ejemplo: Dada una matriz A de $m \times n$ con valores enteros, un vector $\vec{\mathbf{b}} \in \mathbb{Z}^m$, y un vector $\vec{\mathbf{c}} \in \mathbb{Z}^n$, indique si existe una solución $\vec{\mathbf{x}} \in \mathbb{Z}^n$ del sistema de inecuaciones $A\vec{\mathbf{x}} \leq \vec{\mathbf{b}}$, $\vec{\mathbf{x}} \geq \vec{\mathbf{0}}$, y de ser así, encontrar una solución factible $\vec{\mathbf{x}}_{\text{o}}$ tal que

$$\vec{\mathbf{c}}^T \vec{\mathbf{x}}_{\text{o}} = \text{mín} \left\{ \vec{\mathbf{c}}^T \vec{\mathbf{x}} : A\vec{\mathbf{x}} \leq \vec{\mathbf{b}}, \vec{\mathbf{x}} \geq \vec{\mathbf{0}} \right\}.$$

Ejemplo. Consideremos el problema del vendedor viajero. Podemos encontrar varias versiones de este problema, pertenecientes a cada uno de los tipos mencionados arriba.

Versión de decisión Dada una matriz de distancias entre ciudades, y un número k, ¿existe algún recorrido del vendedor viajero con distancia total $\leq k$?

Versión de búsqueda Dada una matriz de distancias entre ciudades para cual existe un recorrido del vendedor viajero de largo total $\leq k$, hallar un tal recorrido.

Versión de evaluación Dada una matriz de distancias entre ciudades, ¿cuál es la distancia total mínima posible para un recorrido del vendedor viajero?

Versión de optimización Dada una matriz de distancias entre ciudades, encontrar un recorrido del vendedor viajero de costo óptimo.

10.3. Medidas del tamaño de una instancia

Dada una instancia I de un problema, nos interesa tener una medida de su *tamaño*.

En principio, quisiéramos medir la cantidad de *bytes* (o bits, o palabras) que ocupa la instancia, pero nos conformaremos con cualquier medida "razonable" de tamaño (qué exactamente es razonable lo discutiremos más adelante).

Las mediciones del tamaño de una instancia en términos de la cantidad de memoria que ocupa el representarla en el computador las llamaremos las medidas *naturales* del tamaño.

Supongamos ahora que hemos fijado una forma de medir tamaños de instancias. Nos referiremos al tamaño de una instancia I, medido de esa forma, por $|I|$.

Así, dado un algoritmo $\mathcal{A}$, su *complejidad* es la función

$$\mathcal{C}_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$$

definida por

$$\mathcal{C}_{\mathcal{A}}(n) = \text{máx} \{ T_{\mathcal{A}}(I) : |I| = n \}.$$

Medidas "razonables" de tamaño de una instancia

¿Qué formas de medir tamaños de instancias son considerados "razonables"?

Es posible medir tamaños de las instancias de un problema de varias maneras distintas.

Sean s_1 y s_2 dos medidas de tamaño de las instancias de un problema dado π. Diremos que dichas medidas están *polinomialmente relacionadas* si existen dos polinomios $p(x)$ y $q(x)$ tales que, dada cualquier instancia I de π, se cumple

$$s_1(I) \leq p(s_2(I)) \qquad \text{y} \qquad s_2(I) \leq q(s_1(I)).$$

O sea, si cada medida está "polinomialmente acotada" por la otra.

Note que si dos medidas s_1 y s_2 del tamaño de las instancias de π están polinomialmente relacionadas, y se tiene un algoritmo $\mathcal{A}$ para resolver π, la complejidad de $\mathcal{A}$ será polinomial en $s_1(I)$ si y sólo si es polinomial en $s_2(I)$.

O sea: la clase de problemas solubles eficientemente *no cambia si medimos los tamaños de las instancias en términos de s_1 o en términos de s_2*.

Estamos finalmente en condiciones de responder nuestra pregunta. Una medida de tamaño de instancias de π será considerada "razonable" si y sólo si está polinomialmente relacionada con las medidas naturales de tamaño de instancias de π.

Ejercicio. ¿Es posible que una medida de tamaño de instancias esté polinomialmente relacionada con la medida en *bits* y no con la medida en *bytes*, o viceversa?

10.4. Complejidad de un problema. Algoritmos eficientes.

Dado un problema π y una función $f : \mathbb{N} \to \mathbb{N}$, diremos que la *complejidad* de π es $O(f)$ si existe un algoritmo $\mathcal{A}$ para π tal que $\mathcal{C}_{\mathcal{A}} \in O(f)$.

Queremos identificar aquellos problemas que pueden ser resueltos eficientemente. Diremos que un problema puede ser resuelto eficientemente si existe un polinomio $p(n)$ tal que su complejidad es $O(p(n))$.

En particular, estamos interesados en la clase P definida como:

$$P = \{\textit{problemas de decisión para los que existe un algoritmo polinomial}\}.$$

Informalmente, diremos que los problemas de la clase P son *tratables*, mientras lo que no están en P son *intratables*.

¿Y qué pasa con los problemas que no son de decisión?

Problemas de decisión *vs* otros problemas

Desde ahora en adelante, estudiaremos principalmente problemas de decisión. Dado un problema de decisión π, nos interesa saber si $\pi \in P$ o no.

En principio, esto puede parecer restrictivo. Sin embargo, en general es posible, dado un problema de optimización, búsqueda o evaluación, *reducirlo* a un problema de decisión.

10.5. Reducciones entre problemas

Formalmente, una *reducción* de un problema π a otro π' consiste en la exhibición de un algoritmo que resuelve π, tomando como hipótesis la existencia de un algoritmo para resolver π'.

No estamos interesados en cualquier reducción, sino en reducciones que, a partir de una instancia I de π:

1. utilicen $\mathcal{A}'$ una cantidad polinomial de veces; y
2. cada vez que utilicen $\mathcal{A}'$, la instancia I' de π' pueda ser obtenida a partir de I en tiempo polinomial en $|I|$.

Es fácil ver que, si $\mathcal{A}'$ es un algoritmo polinomial para resolver π', entonces una reducción de este tipo tomará tiempo polinomial para resolver I.

Ejemplos de reducciones entre problemas

Quizás los ejemplos más importantes de reducción entre problemas sean los que relacionan problemas de optimización (o de evaluación) con los correspondientes problemas de decisión.

A continuación mostramos algunos de estos ejemplos.

1. **Problema:** CLIQUE

   Versión de optimización: Dado un grafo $G = (V, E)$, hallar un *clique* (subconjunto C de V, tal que todo vértice de C es adyacente a todos los otros) del tamaño máximo posible.

   Versión de decisión: Dados un grafo $G = (V, E)$ y un entero positivo k, ¿tiene G un clique de tamaño $\geq k$?

2. **Problema:** KNAPSACK (la mochila)

   Versión de optimización: Dados dos números enteros positivos n y C, y para cada $i \in \{1, \ldots, n\}$ dos números v_i y w_i, hallar un subconjunto S de $\{1, \ldots, n\}$ que satisfaga $\sum_{i \in S} w_i \leq C$ y que maximice el valor de $\sum_{i \in S} v_i$.

Si interpretamos los números v_i y w_i como los *valores* y los *pesos* de n ítems, este problema puede ser interpretado como hallar un conjunto de dichos ítems que, teniendo peso total $\leq C$ (la *capacidad* de una mochila) maximice el valor de un conjunto de ítems que puede ser llevado en la mochila.

Versión de decisión: Dados tres números enteros positivos n, C y K, y para cada $i \in \{1, \ldots, n\}$ dos números v_i y w_i, ¿existe un subconjunto S de $\{1, \ldots, n\}$ que satisfaga $\sum_{i \in S} w_i \leq C$ y $\sum_{i \in S} v_i \geq K$? O sea, ¿existe un conjunto de ítems de valor al menos K que quepa en la mochila de capacidad C?

3. **Problema:** SUMA DE UN SUBCONJUNTO

   Versión de optimización: Dados dos enteros positivos n y C, y n enteros positivos $\{x_1, \ldots, x_n\}$, hallar un subconjunto S de $\{1, \ldots, n\}$ que satisfaga $\sum_{i \in S} x_i \leq C$ y que alcance el mayor valor de esta suma, entre los que satisfacen esta restricción.

   Versión de decisión: Dados dos números enteros positivos n y C, y n enteros positivos $\{x_1, \ldots, x_n\}$ ¿existe un subconjunto S de $\{1, \ldots, n\}$ que satisfaga $\sum_{i \in S} x_i = C$?

4. **Problema:** RUTA MÁS LARGA

   Versión de optimización: Dado un entero positivo n, una matriz M de $n \times n$ de enteros positivos, y dos números $i, j \in \{1, 2, \ldots, n\}$ (donde M_{uv} representa la distancia que se recorre al ir directamente desde u hasta v en un grafo que tiene por conjunto de vértices a $\{1, \ldots, n\}$), hallar una ruta entre u y v (sin repetir vértices) que recorra una distancia total máxima posible.

   Versión de decisión: Dados dos enteros positivos n y K, una matriz M de $n \times n$ de enteros positivos, y dos números $i, j \in \{1, 2, \ldots, n\}$ (donde M_{uv} representa la distancia que se recorre al ir directamente desde u hasta v en un grafo que tiene por conjunto de vértices a $\{1, \ldots, n\}$), ¿existe una ruta entre u y v (sin repetir vértices) que recorra una distancia total $\geq K$?

5. **Problema:** VENDEDOR VIAJERO

   Versión de optimización: Dados un entero positivo n y una matriz M de $n \times n$ formada por enteros positivos, hallar una permutación $(\sigma(1), \sigma(2), \ldots, \sigma(n))$ que minimice el valor de

   $$C(\sigma) = \sum_{i=1}^{n-1} M_{\sigma(i),\sigma(i+1)} + M_{\sigma(n),\sigma(1)}.$$

   El valor $C(\sigma)$ representa el costo de un recorrido del vendedor viajero por n ciudades, en que el costo de ir de la ciudad i a la j está dado por $M_{i,j}$ y el viajereo recorre las ciudades en orden $(\sigma(1), \sigma(2), \ldots, \sigma(n), \sigma(1))$.

   Versión de decisión: Dados dos enteros positivos n y P, una matriz M de $n \times n$ formada por enteros positivos, ¿existe una permutación $(\sigma(1), \sigma(2), \ldots, \sigma(n))$ tal que

   $$C(\sigma) = \sum_{i=1}^{n-1} M_{\sigma(i),\sigma(i+1)} + M_{\sigma(n),\sigma(1)} \leq P?$$

   Podemos interpretar esta pregunta como sigue: dado, además de los datos de la versión de optimización, un *presupuesto* P, ¿existe un recorrido del vendedor viajero que tenga un costo $\leq$ que el presupuesto P?

   Solución:

   Asumamos que el grafo tiene n nodos, y que los valores de los pesos de las aristas son números de m bits.
   Para poder resolver el problema tenemos que resolver 2 cosas:

a) Calcular el costo del ciclo óptimo.
La manera más simple de lograr este objetivo es simplemente preguntar al problema de decisión si se puede encontrar una solución del problema del vendedor viajero para cada valor entero partiendo desde 0, luego esto nos dará el valor óptimo con el primer entero que el algoritmo retorne verdadero, con lo que tendríamos esta parte solucionada.
Lo anterior tiene un problema, en el peor de los casos, tendremos que llamar al algoritmo de decisión una cantidad de veces igual al valor máximo de la suma de todos los costos posibles, lo que es $n^2 2^m$, lo que claramente es exponencial en m, lo que no cumple con la parte de encontrar una solución polinomial para el problema de optimización.
Luego de esto tenemos que tener un enfoque un poco más inteligente al problema, y esto puede solucionarse fácilmente utilizando el algoritmo de BINSEARCH que ya estudiamos, y que ya demostramos que para un problema de largo n la complejidad es $O(\log_2(n))$, con lo que podemos ver que en el peor de los casos estaremos llamando al algoritmo de decisión una cantidad de veces $O(\log_2(n^2 2^m)) = O(2\log_2(n) + m)$ lo que claramente es polinomial en n y en m.

b) Encontrar un ciclo del costo antes calculado.
Llamando l al costo del ciclo óptimo (calculado anteiormente), utilizaremos el siguiente algoritmo:

- Para cada arista en la matriz M le cambiaremos su costo por $l + 1$.
- Le preguntaremos al algoritmo de decisión si es posible encontrar un ciclo que pase por todos los nodos y que tenga costo menor o igual a l (con lo que no dejamos que el ciclo pase por la arista que estamos analizando).
- Si la respuesta anterior es no, cambiamos el valor de la arista por el que tenía originalmente

Finalmente, en la matriz M habrán muchas aristas con costo $l + 1$, y otras con sus valores originales (menores o iguales a l), las aristas con sus valores originales son las que hacen el ciclo óptimo.
Esta parte se ejecuta n^2 veces, por lo que esta parte es polinomial tanto en n como en m.

Como ya resolvimos de manera polinomial las 2 partes del problema, el problema completo es solucionable en tiempo polinomial.

10.6. La clase NP (Non-deterministic Polynomial)

Hay (al menos) dos maneras equivalentes de definir la clase NP:

1. Son los problemas de decisión que pueden ser resueltos en tiempo polinomial por un *algoritmo no determinístico*.
2. Son los problemas de decisión para los que, asociado con cada instancia I para la que la respuesta es **SI** existe un *certificado* $c(I)$ de largo polinomial en $|I|$ y que puede ser chequeado en tiempo polinomial en $|I|$.

Muchos problemas de decisión pertenecen a esta clase.

Notación. Si $\pi = (D, L)$ es un problema de decisión, dada una instancia I de π (o sea, un elemento de D) anotaremos $I \in \pi$ para indicar $I \in L$.

10.6.1. Algoritmos no determinísticos

Un algoritmo no determinístico de tiempo polinomial para un problema π consta de dos módulos: un módulo *adivinador* y un módulo *verificador*, y se comporta así: dada una entrada w,

- el módulo adivinador genera, en forma no determinista, una palabra c en tiempo polinomial (c.r. al largo de la entrada w).
- Entrega w y c al módulo verificador, que responde **SI** o **No** (en forma determinista) en tiempo polinomial (c.r. al largo de w yc), según si c es un certificado de pertenencia de w a π.

Un problema de decisión $\pi = (D, L)$ es soluble por un algoritmo no determinístico de tiempo polinomial $\mathcal{A}$ si, para cada instancia positiva $w \in L$, existe *alguna* computación con entrada w que termina en respuesta **SI**.

10.6.2. Ejemplos de problemas en NP

- $CG(k)$: grafos k-coloreables. Aquí el certificado es una k-coloración propia.
- PVV_D: el problema del vendedor viajero, versión de decisión. Aquí el certificado es un circuito de costo $\leq k$.
- $CLIQUE_D$: dado un grafo G y un entero positivo k, ¿tiene G un *clique* (subgrafo completo) de tamaño k? Aquí el certificado es la lista de los k vértices.
- *Compuesto*: dado $n \in \mathbb{N}$, un certificado de que n es compuesto es uno de sus factores.

Ejercicio. ¿Es posible dar un certificado eficiente de que un número es primo?

10.7. Problemas NP-completos

10.7.1. Transformaciones entre problemas de decisión

Dados dos problemas de decisión $\pi = (D, L)$ y $\pi' = (D', L')$, diremos que π es *transformable en* π' si existe un algoritmo *polinomial en* $|I|$ que, dada una instancia I de π genera una instancia I' de π' de modo que

$$I \in L \leftrightarrow I' \in L'.$$

Si π es transformable en π', anotaremos $\pi \propto \pi'$.
Para ejemplos de transformaciones, ver más adelante.
Estamos en condiciones de definir la clase de problemas NP-completos.
Un problema π se dice NP-*completo* si:

1. $\pi \in NP$; y
2. dado cualquier problema $\pi' \in NP$, existe una transformación $\pi' \propto \pi$.

Una consecuencia inmediata de que un problema de decisión π sea NP-completo es que, si hubiera un algoritmo polinomial para resolver π, entonces dado *cualquier problema* $\pi' \in NP$, habrí a un algoritmo polinomial para resolver π'.

A su vez, esto implicaría que muchos problemas de búsqueda, optimización y evaluación podrían ser resueltos en tiempo polinomial.

Ejemplos de transformaciones entre problemas de decisión

1. Considere los siguientes problemas:

   - CAMINO HAMILTONIANO
     Dado un grafo G ¿Existe un camino que pase por todos los nodos sin repetir ninguno?.
   - CICLO HAMILTONIANO
     Dado un grafo G ¿Existe un ciclo que pase por todos los nodos sin repetir ninguno?.

   a) Encuentre una función f, que sea calculable en tiempo polinomial, tal que

$$\text{CAMINO HAMILTONIANO } (G) \leftrightarrow \text{CICLO HAMILTONIANO } (f(G))$$

b) Encuentre una función h, que sea calculable en tiempo polinomial, tal que

$$\text{CICLO HAMILTONIANO } (G) \leftrightarrow \text{CAMINO HAMILTONIANO } (h(G))$$

Solución:

a) Insertele al grafo un nodo llamado u y conéctelo a todos los nodos de G.
Luego de esto debemos demostrar que

$$\text{CAMINO HAMILTONIANO } (G) \leftrightarrow \text{CICLO HAMILTONIANO } (f(G))$$

- CAMINO HAMILTONIANO $(G) \rightarrow$ CICLO HAMILTONIANO $(f(G))$
  Sabemos que existe un camino hamiltoniano en G, por lo que existen 2 nodos v y w tales que son los extremos de ese camino, por lo que podemos detectar un ciclo hamiltoniano en $f(G)$ que pasa por el camino hamiltoniano en G, luego desde v pasa a u y de u a w, con lo que hemos cerrado el ciclo.
- CICLO HAMILTONIANO $(f(G)) \rightarrow$ CAMINO HAMILTONIANO (G)
  Sabemos que existe un ciclo hamiltoniano en $f(G)$, por lo que podemos encontrar 2 nodos v y w que son vecinos a u en el ciclo hamiltoniano, luego de esto vemos que necesariamente tiene que existir un camino hamiltoniano entre v y w en el grafo G, por lo que necesariamente existe un camino hamiltoniano en G.

Finalmente, como hemos demostrado la implicancia para ambos lados, hemos demostrado que

$$\text{CAMINO HAMILTONIANO } (G) \leftrightarrow \text{CICLO HAMILTONIANO } (f(G)).$$

b) Llevense a cabo los siguientes pasos:

- Tómese un nodo y llámesele v.
- Insertele al grafo un nodo llamado v' y conéctelo a v.
- Insertele al grafo un nodo llamado w y conéctelo a todos los vecinos de v en el grafo original.
- Insertele al grafo un nodo llamado w' y conéctelo a w.

Luego de esto debemos demostrar que

$$\text{CICLO HAMILTONIANO } (G) \leftrightarrow \text{CAMINO HAMILTONIANO } (h(G))$$

- CICLO HAMILTONIANO $(G) \rightarrow$ CAMINO HAMILTONIANO $(h(G))$
  Tómese el camino hamiltoniano en G que comienza en v y termina en uno de sus vecinos (camino que existe, ya que G tiene un ciclo hamiltoniano), luego tenemos el camino hamiltoniano en $h(G)$ que parte en v', luego pasa por v, tomando el camino hamiltoniano en G que se encontró anteriormente, que termina en un vecino de v, por lo que también es vecino de w, el camino pasa luego por w y termina en w'. Hemos encontrado un camino hamiltoniano en $h(G)$.
- CAMINO HAMILTONIANO $(h(G)) \rightarrow$ CICLO HAMILTONIANO (G)
  Si se tiene un camino hamiltoniano en $h(G)$, entonces ese camino debe comenzar en v' y terminar en w' (debido a que ambos nodos son de grado 1), por lo que:
  - Existe un camino entre v y w' que pasa por todos los nodos de $h(G)$ excepto por v'.
  - Existe un camino entre v y w que pasa por todos los nodos de $h(G)$ excepto por v' y w'.
  - Existe un camino entre v y uno de sus vecinos que pasa por todos los nodos de $h(G)$ excepto por v', w y w'.
  - Existe un ciclo que pasa por todos los nodos de $h(G)$ excepto por v', w y w'.

- Existe un ciclo que pasa por todos los nodos de G.
- Existe un ciclo hamiltoniano en G.

Finalmente, como hemos demostrado la implicancia para ambos lados, hemos demostrado que

$$\text{CICLO HAMILTONIANO } (G) \leftrightarrow \text{CAMINO HAMILTONIANO } (h(G)).$$

- SUMA DE UN SUBCONJUNTO
  Dados 2 números enteros positivos n y C, y un conjunto de n enteros positivos $A = \{x_1, ..., x_n\}$ ¿Existe un conjunto $S \subseteq A$ que satisfaga $\sum_{i \in S} x_i = C$?.
- PARTICION
  Dados un número entero positivo n y un conjunto de n enteros positivos $A = \{x_1, ..., x_n\}$ ¿Existe un conjunto $S \subseteq A$ que satisfaga $\sum_{i \in S} x_i = \sum_{i \in A-S} x_i$?.

Se desea mostrar que es posible transformar el problema de SUMA DE UN SUBCONJUNTO al problema de PARTICION, o sea, dada una instancia I del primero, encontrar una instancia I' del segundo, en forma tal que la respuesta (SI o NO) a ambas sea la misma. Muestre como operaría esta transformación si se tiene la instancia I de SUMA DE UN SUBCONJUNTO dada por

$n = 20, C = 1000, A = \{14, 25, 29, 38, 40, 47, 59, 73, 99, 100, 125, 126, 132, 133, 134, 144, 158, 164, 173, 177\}$.

Nota: Puede usar el hecho de que

14+25+29+38+40+47+59+73+99+100+125+126+132+133+134+144+158+164+173+177 = 1990.

Solución:

Primero que todo definiremos la variable $s = \sum_{a \in A} a$, con esta variable definida podemos ver claramente 2 casos:

a) $2 \times C = s$. En este caso el problema de SUMA DE UN SUBCONJUNTO se reduce al problema de PARTICION.

b) $2 \times C \neq s$. Tómese $\triangle = |s - 2 \times C|$. Con esto tenemos 2 posibilidades:

1) $\triangle \notin A$. En este caso definimos $A' = A \cup \{\triangle\}$, y podemos ver que ahi

$$\text{SUMA DE UN SUBCONJUNTO } (n, C, A) \leftrightarrow \text{PARTICION } (n+1, A')$$

Ya que si $2 \times C > s$, la partición que no contiene a $\triangle$ sumará C, y en caso contrario, la partición que si contiene a $\triangle$ sumará $C + \triangle$.

2) $\triangle \in A$, en este caso definimos $\triangle_1 = s + 1$ y $\triangle_2 = s + \triangle + 1$. Con estos valores nos aseguramos que si PARTICION nos dice que si, entonces $\triangle_1$ y $\triangle_2$ estarán en particiones distintas- Luego de esto definimos $A'' = A \cup \{\triangle_1, \triangle_2\}$, y podemos ver que ahi

$$\text{SUMA DE UN SUBCONJUNTO } (n, C, A) \leftrightarrow \text{PARTICION } (n+2, A'')$$

Ya que si $2 \times C > s$, la partición que contiene a $\triangle_1$ sumará $C + s + 1$, y en caso contrario, la partición que contiene a $\triangle_2$ sumará $C + \triangle + s + 1$.

Finalmente encontramos una solución para cada una de las posibilidades, por lo que siempre podemos encontrar una instancia de PARTICION a partir de una instancia de SUMA DE UN SUBCONJUNTO que cumpla con tener igual respuesta.
Para ver como se comportaría esto con los datos anteriormente ingresados tenemos que ver lo siguiente:

- $s = 1990$.
- Vemos que $2 \times C \neq s$, por lo que definimos $\triangle = |s - 2 \times C| = |1990 - 2000| = |-10| = 10$.
- Vemos que $10 \notin A$, por lo que definimos

  $A' = \{10, 14, 25, 29, 38, 40, 47, 59, 73, 99, 100, 125, 126, 132, 133, 134, 144, 158, 164, 173, 177\}$.

- Resolvemos PARTICION $(21, A')$.

En las siguientes preguntas, muestre una reducción en tiempo polinomial de la versión de optimización a la versión de decisión del problema presentado. En otras palabras, diseñe un algoritmo que, dada una instancia de la versión de optimización, la resuelva creando una cantidad polinomial de instancias de la versión de decisión y resolviendo esas instancias.

En las siguientes preguntas, muestre una transformación en tiempo polinomial desde el primer problema de decisión al segundo. O sea, muestre cómo, dada una instancia del primer problema, es posible construir —en tiempo polinomial— una instancia del segundo problema, de modo que las respuestas a ambas instancias sean iguales (ambas SI o ambas NO).

2. **Problema 1:** CIRCUITO HAMILTONIANO

   Dado un grafo $G = (V, E)$, ¿existe en G un *circuito Hamiltoniano*, es decir, un camino cerrado que pase por cada vértice exactamente una vez?

   Problema 2: CAMINO HAMILTONIANO

   Dado un grafo $G = (V, E)$, ¿existe en G un *camino Hamiltoniano*, es decir, un camino —no cerrado— que pase por cada vértice exactamente una vez?

3. **Problema 1:** CAMINO HAMILTONIANO

   Problema 2: CIRCUITO HAMILTONIANO

4. **Problema 1:** CUBRIMIENTO POR TRIPLES

   Dados un entero positivo n, tres conjuntos $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$ y $Z = \{z_1, \ldots, z_n\}$, y un conjunto T de *triples* de la forma (x, y, z) con $x \in X, y \in Y, z \in Z$, ¿es posible seleccionar n triples de T de modo que cada elemento de $X \cup Y \cup Z$ aparezca en exactamente un triple del conjunto seleccionado?

   Problema 2: SUMA DE UN SUBCONJUNTO

10.7.2. Ejemplo de problema de decisión: SAT

SAT: satisfactibilidad de fórmulas proposicionales.

Problema: dada una fórmula proposicional φ, decidir si existe una asignación de valores de verdad (0 o 1) que la hace verdadera.

Un posible algoritmo de decisión: chequear la tabla de verdad de φ.

Respóndase **SI** o **NO** según haya una fila que termine en 1.

En el peor caso, este algoritmo es exponencial en el largo de la fórmula.

No se conoce una solución de tiempo polinomial, i.e., no se sabe si $SAT \in P$.

10.7.3. SAT en forma normal conjuntiva

Otro problema de decisión (que denotaremos $SAT\text{-}FNC$) es similar a SAT, pero exigiendo que todas las fórmulas están expresadas en *Forma Normal Conjuntiva*.

Dado F_0 (el conjunto de *proposiciones atómicas*), un *literal* es toda fórmula atómica (variable proposicional) o negación de una fórmula atómica. Por ejemplo, si $F_0 = \{p, q, r\}$, entonces los literales son $p, q, r, \neg p, \neg q$ y $\neg r$.

Recordemos que una fórmula proposicional está en *Forma Normal Conjuntiva* (FNC) si es una conjunción de disyunciones de literales:

$$\bigwedge_{i=0}^{n} (\bigvee_{j=0}^{m_i} l_{ij}).$$

Por ejemplo, la fórmula $(p \vee q) \wedge (\neg p \vee \neg r \vee s) \wedge (q \vee t)$ está en FNC.

Cada disyunción se llama *cláusula*.

10.7.4. Transformaciones entre $SAT\text{-}FNC$ y SAT

Se sabe que $SAT\text{-}FNC \propto SAT$ y viceversa.

Que $SAT\text{-}FNC \propto SAT$ es obvio: dada una instancia I de $SAT\text{-}FNC$, basta tomar $I' = I$ como instancia de SAT y vemos que $I \in SAT\text{-}FNC \leftrightarrow I' \in SAT$.

Para el recíproco, uno estaría tentado de usar el siguiente teorema:

Teorema. Toda fórmula de un lenguaje $L(P)$ es lógicamente equivalente a una fórmula que está en FNC.

¿Cuál sería el problema?

Mencionaremos una demostración un poco más alambicada de que $SAT \propto SAT\text{-}FNC \ldots$

10.7.5. Relación entre P y NP

Se tiene trivialmente $P \subseteq NP$.

Tenemos muchos problemas en NP que no sabemos si están en P o no (SAT, $CLIQUE_D$, PVV_D, etc.).

Conjetura de Steve Cook (1971): $P \subsetneq NP$, es decir, habría problemas en NP que no están en P.

¿Candidatos? Todos los problemas NP-completos. ?Sabemos si existen problemas NP-completos? Sí: el primero fue $SAT - FNC$; después vinieron otros como $CLIQUE_D$, $CG(3)$, PVV_D, ...

10.8. El teorema de Cook

Teorema (Cook, 1971). *SAT-FNC es NP-completo.*

Recordemos que esto quiere decir que:

- $SAT\text{-}FNC \in NP$.
- Dado un problema cualquiera $\pi \in NP$, se tiene $\pi \propto SAT\text{-}FNC$ (o sea, π puede ser transformado en tiempo polinomial a $SAT\text{-}FNC$).

Es decir, $SAT\text{-}FNC$ es el problema más difícil (o uno de los problemas más difíciles) en la clase NP.

Si se pudiera resolver $SAT\text{-}FNC$ en tiempo polinomial determinista, **todo** problema de la clase NP también, automáticamente ...

En particular, $CG(3)$, PVV_D, $CLIQUE_D$, ..., pueden ser transformados en tiempo polinomial a $SAT\text{-}FNC$.

Más aún, se demostró después (Richard Karp, 1972) que c/u de estos problemas también es NP-completo.

10.8.1. ¿Cómo se demostraría que $P = NP$ (o que $P \neq NP$)?

Ésta es la pregunta del millón de dólares[1] ...

Para demostrar que $P = NP$, bastaría probar que existe *algún* problema NP-completo que está en P.

Para demostrar que $P \neq NP$, bastaría probar que existe *algún* problema NP-completo que no está en P.

Demostrar que un problema es NP-completo es dar un fuerte indicio de que es muy difícil, de hecho, un indicio (pero no demostración) de que no es soluble en tiempo polinomial por algoritmos deterministas.

Si se demuestra que uno de ellos está (o no está) en P, todos los otros problemas NP-completos le siguen ...

Hay unos 10.000 problemas NP-completos esperando, entre ellos muchos de alta importancia práctica.

10.8.2. ¿Cómo se demuestra que un problema es NP-completo?

1. Demostrando que está en NP, y
2. Transformando a él, en tiempo polinomial, otro problema ya establecido como NP-completo.

En lugar de demostrar que todo problema $\in NP$ puede ser transformado a nuestro candidato a problema NP-completo, basta probar (2), gracias a la transitividad de $\propto$.

Ejemplo: CLIQUE es NP-completo

Solución: Por la transitividad de las reducciones polinomiales, y porque $CLIQUE \in NP$ (¿por qué?), basta con demostrar que: $SAT \propto CLIQUE$.

Es decir, hay que encontrar una función f computable en tiempo polinomial tal que, si x codifica una fórmula φ en FNC, entonces $f(x)$ codifica un grafo $G = (V, E)$ y un entero k tales que: φ es satisfactible si y sólo si G tiene un subgrafo completo de tamaño k (la misma construcción que sigue funciona si pedimos "de tamaño $\geq k$").

Veamos cómo construir G y k a partir de φ: primero, k es el número de cláusulas en φ, es decir, φ es de la forma $C_1 \wedge \cdots \wedge C_k$.

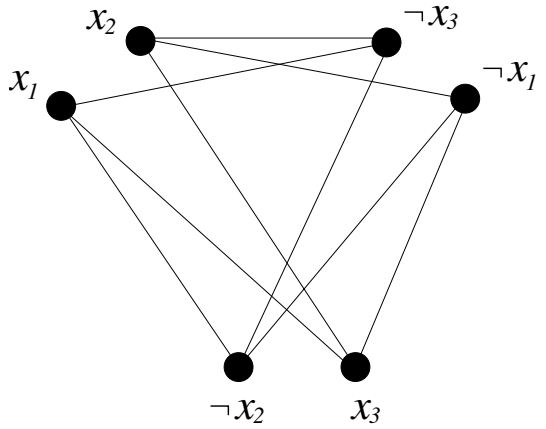
Ahora, cada literal en φ (esté repetido o no) aporta un vértice al grafo G, es decir, si C_i tiene m literales, esta clásula aporta m vértices a G.

Con respecto a las aristas, colocamos una arista uniendo a los vértices i y j si y sólo si:

[1] Literalmente.

- i y j provienen de diferentes clásulas, y
- i y j (mejor dicho, sus correspondientes literales) no son complementarios, es decir, no es uno la negación del otro.

Por ejemplo, la fórmula $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee \neg x_1)$ da lugar a $k = 3$ y a un grafo con 6 vértices:



Afirmación: G tiene un subgrafo completo de tamaño k si y sólo si φ es satisfactible.

En efecto: Primero demostramos "$\Rightarrow$".

Nótese que si i y j están conectados por una arista en G, entonces se les puede dar el valor de verdad 1 en φ, sin conflicto.

Si hay un subgrafo completo de tamaño k, entonces φ tiene al menos k literales, todos de cláusulas diferentes, y a cada uno se le puede asignar el valor de verdad 1 en φ, sin entrar en conflictos con los demás.

Como φ tiene exactamente k cláusulas, esto da un literal en cada clásula, y así, se puede hacer verdadera la fórmula completa.

Esta es una asignación de verdad que satisface la fórmula.

Ahora demostramos la implicación inversa "$\Leftarrow$".

Supongamos que φ es satisfecha por una asignación σ.

Fija σ, cada cláusula C_j debe tener un literal l^j tal, que $\sigma(l^j) = 1$.

El conjunto de literales $\{l^1, \ldots, l^k\}$ corresponde a un subgrafo completo en G de tamaño k.

Para probar esto, necesitamos sólo establecer que, para i y j arbitrarios, l^i y l^j están conectados por una arista en G.

Nótese que:

- l^i y l^j provienen de cláusulas diferentes (por construcción), y
- l^i y l^j no entran en conflicto uno con otro (pues a ambos σ les da el valor de verdad 1).

Esto demuestra la afirmación.

Finalmente, hay que mencionar que esta construcción puede ser hecha en tiempo polinomial en $|\varphi|$.

Esto es fácil de ver, pues G tiene tantos vértices como literales tiene φ, digamos n.

Determinar cuáles son las aristas a colocar en G toma tiempo n^2.

10.8.3. La demostración del teorema de Cook

Todo eso está muy bien, pero cuando Cook demostró que $SAT\text{-}FNC$ es NP-completo lo hizo sin tener un problema NP-completo en el cual apoyarse.

¿Cómo lo hizo?

Sea π un problema en NP. Eso quiere decir que existe un polinomio $p(n)$ y un algoritmo no determinístico $\mathcal{A}$ (modelado matemáticamente por una máquina de Turing) que, dada una

instancia I de π de largo $|I| = n$, intenta adivinar un certificado $c(I)$ e intenta chequearlo en tiempo $\leq p(n)$.

Cook mostró que existe un polinomio $q(n)$ tal que es posible, dada la instancia I, construir en tiempo $\leq q(|I|)$ una fórmula proposicional $\varphi(I)$ que es satisfactible si y sólo si existe una computación de $\mathcal{A}$ que le permite demostrar que $I \in \pi$ (de hecho, la asignación de verdad que satisface $\varphi(I)$ corresponde a las distintas etapas y condiciones de dicha computación).

10.9. Otros problemas NP–completos

Recubrimiento de un Grafo por Vértices: Dado un grafo G y un entero positivo k, ¿existe un recubrimiento de G de tamaño a lo más k?

Partición: Dado un conjunto A de enteros positivos, ¿es posible encontrar un subconjunto S de A tal que

$$\sum_{x \in S} x = \sum_{x \in A-S} x?$$

Suma de un subconjunto: Dado un conjunto A de números enteros positivos, y un entero positivo K, ¿es posible encontrar un subconjunto S de A tal que

$$\sum_{x \in S} x = K?$$

3-colorabilidad: Dado un grafo G, ¿es posible pintar los vértices de G con 3 colores de modo que vértices adyacentes tengan colores distintos?

10.9.1. Ejemplo: recubrimiento de un grafo por vértices

Considere el siguiente problema:

Problema de Recubrimiento de un Grafo por Vértices (RGV).

Dado un grafo $G = (V, E)$, tenemos las siguientes versiones:

Problema de optimización (RGV_O): encontrar un recubrimiento de tamaño mínimo, es decir, un conjunto de vértices de tamaño mínimo tal, que cada arista tiene al menos un extremo en el subconjunto.

Problema de decisión (RGV_D): dado un entero positivo k, ¿existe un recubrimiento de G de tamaño a lo más k? (este problema es NP-completo).

Demuestre que, si RGV_D se puede resolver en tiempo polinomial, enton ces también RGV_O se puede resolver en tiempo polinomial (el recíproco es inmediato).

Ayuda: primero determine el tamaño N de un recubrimiento óptimo de G.

¡Cuidado con la tentación ...!

Uno podría sentirse tentado, una vez calculado el tamaño N de un recubrimiento óptimo de G, a chequear todos los posibles subconjuntos de vértices de tamaño N, para ver si forman un recubrimiento. Este algoritmo no es de tiempo polinomial:

Primero que todo, chequear si un subconjunto de tamaño N es recubrimiento toma (incluso usando fuerza bruta) tiempo $O(N^3) \subseteq O(n^3)$, es decir, polinomial en n. Ahí no está el problema.

Por otro lado, hay $\binom{n}{N}$ subconjuntos a verificar. Esto da tiempo $O(n^3 \times \binom{n}{N}$. El coeficiente binomial da $\frac{(N+1)\cdots n}{1\cdots(n-N)}$ que es menor o igual a n^N. Esto podría sugerir que, como N está fijo, que el tiempo es polinomial en n. Sin embargo, el problema original es el de encontrar el recubrimiento óptimo, y en ese problema no aparece el N como parámetro. El N salió de manera intermedia y depende de n (y de la instancia), de hecho, N puede ser arbitrariamente cercano a n (en el peor caso). Así, tenemos un tiempo $O(n^n)$.

Capítulo 11

Aritmética modular y criptografía

11.1. Divisibilidad. Máximo común divisor

Definición 50. Sean $a, b \in \mathbb{Z}$. Decimos que b *divide a* a (o también que a *es divisible por* b, o que a *es un múltiplo de* b, o que b *es un factor de* a, o que b *es un divisor de* a) si y sólo si existe $k \in \mathbb{Z}$ tal que $a = bk$.

Notación. Denotamos el que b divide a a por $b \mid a$.

11.1.1. El algoritmo de la división

El siguiente teorema es llamado el *algoritmo de la división*; aunque no es precisamente un algoritmo, *sugiere* (o más bien dicho, su demostración sugiere) un algoritmo para calcular el cuociente y el resto de una "división entera":

Teorema (Algoritmo de la división). *Sean* $a \in \mathbb{Z}$, $b \in \mathbb{Z}$, $b \neq 0$.

Entonces existen $q, r \in \mathbb{Z}$ *tales que* $a = bq + r$ *y* $0 \leq r < |b|$.

Estos enteros q *y* r *son únicos, y son llamados, respectivamente, el* cuociente *y el* resto *de la división de* a *por* b.

Demostración. Considérese el conjunto

$$S = \{a - bx : x \in \mathbb{Z} \text{ y } a - bx \geq 0\}.$$

Si $a \geq 0$ entonces $a \in S$, y si $a < 0$ entonces $a - |b|\, a \in S$, por lo que —en cualquier caso— $S \neq \emptyset$.

Como además todos los elementos de S son ≥ 0, S es acotado inferiormente. Por el principio del menor entero, S debe tener un elemento mínimo. Llamemos a éste r.

Sea $q \in \mathbb{Z}$ algún entero tal que $a - bq = r$ (dicho entero debe existir, por la definición de S).

Claramente, $r \geq 0$. Pasamos a demostrar que $r < |b|$. Si así no fuera, entonces $r' = r - |b| = a - bq - |b| \geq 0$ y por lo tanto $r' \in S$, $r' < r$ contradiciendo la minimalidad de r.

Así, existen $q, r \in \mathbb{Z}$ tales que $a = bq + r$, $0 \leq r < |b|$.

Demostramos a continuación que q y r son los únicos enteros con estas propiedades.

Si $a = bq' + r'$ con $0 \leq r' < |b|$, entonces $0 = bq + r - (bq' + r') = b(q - q') + (r - r')$, de donde $r' - r = b(q' - q)$, por lo que $r' - r$ es divisible por b.

Pero $0 \leq r, r' < |b|$ implica que $-|b| < r' - r < |b|$, por lo que el único posible valor de $r' - r$ divisible por b es $r' - r = 0$, o sea, $r = r'$.

Finalmente, como $a = bq + r = bq' + r$ y $b \neq 0$, debe tenerse $q = q'$.

□

11.1.2. Divisores comunes. Máximo común divisor

Sean $a, b \in \mathbb{Z}$, no ambos cero.

Consideremos el conjunto

$$S = \{n \in \mathbb{N} : n \text{ divide a } a \text{ y a } b\}.$$

Sabemos que a y b no son ambos cero. SPG, supongamos $a \neq 0$. Entonces $n \in S \rightarrow n \leq |a|$, por lo que S es acotado superiormente. Como además $1 \in S$, $S \neq \emptyset$, por lo que S debe tener un *elemento máximo*.

Sea d el máximo elemento de S. Entonces tenemos:

1. d es un divisor común de a y b; y
2. si d' es un divisor común de a y b, entonces $d' \leq d$.

La propiedad anterior nos permite dar la siguiente definición:

Definición 51. El mayor elemento del conjunto S es llamado el *máximo común divisor de a y b*, y es denotado $\text{mcd}(a, b)$; en algunos libros también (a, b).

Note que esta definición no nos permite hallar en forma eficiente el máximo común divisor entre dos enteros.

Antes de dar un algoritmo para calcular el máximo común divisor entre dos enteros, veamos que

$$\text{mcd}(a, b) = \text{mcd}(|a|, |b|).$$

O sea, podemos siempre reducir el cálculo del máximo común divisor al caso en que ni a ni b son negativos.

11.1.3. El algoritmo de Euclides.

PENDIENTE

11.1.4. El algoritmo extendido de Euclides

Vimos que, dados $a, b \in \mathbb{N}$, $a, b > 0$, se tiene $\text{mcd}(a, b) = \text{mín}\{ax + by : x, y \in \mathbb{Z}\}$.

Una modificación al algoritmo de Euclides nos permite hallar dos enteros x, y tales que $ax + by = \text{mcd}(a, b)$.

PENDIENTE

11.2. Aritmética modular

Consideramos $\mathbb{Z}_n$ con las operaciones $+$ mód n y $\cdot$ mód n (suma y producto módulo n).

Se puede demostrar (hacerlo!) que $[\mathbb{Z}_n, +$ mód $n, \cdot$ mód $n, 0, 1]$ es un *anillo*, es decir, como un cuerpo, pero puede fallar la existencia de inversos multiplicativos, dependiendo de n.

La estructura de $[\mathbb{Z}_n, +$ mód $n, \cdot$ mód $n, 0, 1]$ es como sigue:

- $[\mathbb{Z}_n, +$ mód $n, 0]$ es grupo conmutativo.
- $[\mathbb{Z}_n, \cdot$ mód $n, 1]$ es conmutativo, asociativo, 1 es neutro.
- $\cdot$ mód n distribuye c.r.a. $+$ mód n.

11.2.1. Relación entre $\mathbb{Z}$ y $\mathbb{Z}_n$

Tanto $\mathbb{Z}$ como $\mathbb{Z}_n$ son anillos. ¿Qué propiedades algebraicas se mantienen entre uno y otro? Consideremos $f : \mathbb{Z} \to \mathbb{Z}_n$, definida por:

$$a \mapsto a \mod n. \qquad \text{(el resto de la división de } a \text{ por } n).$$

Se tiene que f es *homomorfismo de anillos* (también $\mathbb{Z}$ es anillo), o sea:

$$f(a_1 \cdot a_2) = (f(a_1) \cdot f(a_2)) \mod n$$
$$f(a_1 + a_2) = (f(a_1) + f(a_2)) \mod n$$

Si escribimos $x +_n y$ en lugar de $(x + y) \mod n$, podemos escribir:

$$f(a_1 + a_2) = f(a_1) +_n f(a_2)$$
$$f(a_1 \cdot a_2) = f(a_1) \cdot_n f(a_2)$$

Otra forma equivalente de escribir esto mismo es:

$$(a_1 + a_2) \mod n = ((a_1 \mod n) + (a_2 \mod n)) \mod n$$
$$(a_1 \cdot a_2) \mod n = ((a_1 \mod n) \cdot (a_2 \mod n)) \mod n$$

Obviamente f no es *isomorfismo*, ya que entonces $\mathbb{Z}$ y $\mathbb{Z}_n$ tendrían la misma cardinalidad.

11.2.2. Inversos en $\mathbb{Z}_n$

Decíamos que $\mathbb{Z}_n$ es un anillo, o sea, tiene estructura similar a un cuerpo, pero no todo elemento $\neq 0$ tiene (necesariamente) un inverso multiplicativo.

¿Qué elementos en $\mathbb{Z}_n - \{0\} = \{1, \ldots, n-1\}$ tienen inverso multiplicativo?

Veamos:

Teorema. $a \in \mathbb{Z}_n - \{0\}$ *tiene inverso multiplicativo en* $\mathbb{Z}_n$ *(o sea, existe* $x \in \mathbb{Z}_n$ *tal que* $a \cdot x \mod n = 1$*) si y sólo si* $\text{mcd}(a, n) = 1$.

Demostración. Sea $d = \text{mcd}(a, n)$.

Supondremos primero que $x \in \mathbb{Z}_n$ es el inverso multiplicativo de a módulo n. Entonces existe $k \in \mathbb{Z}$ tal que $ax = kn + 1$. Pero entonces $1 = ax - kn$, de donde $d \mid 1$ (ya que, como $d \mid a$ y $d \mid n$, se tiene $d \mid ax$ y $d \mid kn$). Así, $d = 1$.

Supongamos ahora que $d = 1$. Entonces podemos asegurar la existencia de dos nńumeros enteros x, y tales que $ax + ny = 1$ (ver problema 2, I3)[1] Pero entonces $ax = -ny + 1$, o sea, $ax \cong 1 \pmod n$, de donde $x = a^{-1}$ en $\mathbb{Z}_n$.

□

11.3. Cuerpos

La propiedad demostrada en la última sección tiene el siguiente

Corolario. $\mathbb{Z}_n$ *es un cuerpo si y sólo si* n *es primo.*

Demostración. Ejercicio.

Ejemplo. $n = 5$.

$+_5$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\cdot_5$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

[1] Más aún: el *Algoritmo Extendido de Euclides* nos da una forma eficiente de calcular estos enteros.

Ejemplo. $n = 4$, $\mathbb{Z}_4$ **no es cuerpo**

$+_4$	0	1	2	3
0	0	1	2	3
1	1	1	3	0
2	2	3	0	1
3	3	0	1	2

$\cdot_4$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

El 2 no tiene inverso multiplicativo: $\text{mcd}(4, 2) = 2 \neq 1$.

A continuación enunciamos (sin demostración todavía) dos teoremas importantes. Daremos las demostraciones más adelante.

11.3.1. El pequeño teorema de Fermat

Teorema (Pequeño teorema de Fermat). *Sea p primo. Para todo $a \in \{1, 2, \ldots, p-1\}$ se tiene*

$$a^{p-1} \text{ mód } p = 1.$$

Ejemplo. Consideremos $p = 11$. Se tiene:

$$1^{10} \text{ mód } 11 = 2^{10} \text{ mód } 11 = \cdots = 10^{10} \text{ mód } 11 = 1.$$

11.3.2. El teorema chino de los restos

Sean $a, b \in \mathbb{Z}$, $a, b \geq 1$, con $\text{mcd}(a, b) = 1$.
Para todo $u, v \in \mathbb{N}$, con $0 \leq u < a, 0 \leq v < b$, existe un único $x \in \mathbb{N}$, tal que:

1. $0 \leq x < ab$
2. $x \text{ mód } a = u$
3. $x \text{ mód } b = v$

Para demostrar el teorema hay que demostrar existencia y unicidad.

11.4. Introducción a la Criptografía

Propósito: poder codificar (encriptar) mensajes (información) para ocultar su contenido.
El proceso de codificación debe ser *efectivo* y *eficiente*.
La decodificación (desencriptación) debe ser *computacionalmente intratable* (cuando no se cuenta con cierta información específica, propia del método de encriptación).
Se denomina "criptografía moderna" a aquella basada en teoría computacional de números.
Hay varios protocolos criptográficos, que tienen que ver con la forma en que se distribuye las funciones de encriptación (o sus claves), y con las funciones mismas.

11.4.1. Un protocolo de encriptación de clave pública

1977: Rivest, Shamir & Adleman (RSA).
Se supone que los mensajes (antes de la encriptación) ya son números enteros.
Agente A quiere recibir mensajes encriptados de otros agentes.
Publica una clave que permite a los otros codificar los mensajes que le envían, pero sólo él puede leerlos.

11.4.2. Codificación en RSA

El agente A:

- Elige dos números primos grandes p y q, y calcula $N = p \cdot q$.
- Elige e tal que: $\text{mcd}(e, (p-1)(q-1)) = 1$, primo relativo con $(p-1)(q-1)$ (en particular, puede ser otro primo).
- Calcula s, t tales que: $se + t(p-1)(q-1) = 1$.

  Esto se puede hacer en tiempo polinomial con el algoritmo extendido de Euclides.
- Publica N y e
- Dice a los otros: al enviarme mensajes M, encríptenlos usando la siguiente función de encriptación:

  $$E(M) = M^e \text{ mód } n.$$

  El resto de la división de M^e por n se puede calcular en tiempo polinomial (pendiente)..

11.4.3. Decodificación en RSA

- ¡¡¡¡A se guarda p, q, s, t!!!!
- Para leer el mensaje, A aplica la función de desencriptación:

  $$D(E(M)) = (E(M))^s \text{ mód } n \stackrel{?}{=} M.$$

  Esto lo puede hacer en tiempo polinomial *si conoce* s y n.

11.4.4. Supuestos para que RSA funcione

Se supone que sólo A conoce la función D.

Se *cree* que, a no ser que sea fácil factorizar números naturales en factores primos, una persona que conoce sólo n y e, no va a poder calcular eficientemente s (pues *parece* que requiere de p y q).

Tampoco se conoce otra forma eficiente de calcular D a partir de n y e.

La idea es que D no pueda ser obtenida fácilmente a partir de E (es lo que se llama "función en un solo sentido", o "funciones con puerta trampa").

Uno de los supuestos básicos es que factorizar números enteros no puede ser hecho en tiempo polinomial (una suposición base de la criptografía, un problema abierto).

11.4.5. Firma de mensajes

Lo anterior también se puede usar para firmar mensajes.

Veíamos que $D(E(M)) = M$ para todo M. Supongamos que, además, $E(D(M)) = M$, para todo M.

A puede firmar un mensaje M enviado a B:

$$A \stackrel{(M, D(M))}{\longrightarrow} B.$$

El número $D(M)$ es la firma del mensaje, (pero no envía la función D).

Sólo el que conoce D, es decir, A, puede aplicar D.

A puede firmar eficientemente mensajes con D.

¿Cómo reconoce B que M fue enviado por A?

11.4.6. Verificación de la firma

B ha recibido un par (M, M').

Para convencerse de que, efectivamente, éste ha sido enviado por A, debe convencerse de que $M' = D(M)$.

¿Cómo se convence de esto?

Como B tiene E (que le fue enviado por A), B aplica E a la "firma" M'.

Si $E(M') = M = E(D(M))$, entonces $M' = D(M)$, y así B sabe que el mensaje fue enviado por A.

B puede verificar eficientemente todo esto.

Hay que probar que el protocolo es correcto, es decir, que

1. $D(E(M)) = M$; 2. $E(D(M)) = M$.

Veremos la demostración de 1. (la otra es similar).

$$\begin{aligned} D(E(M)) &= (E(M))^s \text{ mód } n \\ &= (M^e)^s \text{ mód } n \\ &= M^{e\cdot s} \text{ mód } n \\ &= M^{1-t\cdot(p-1)(q-1)} \text{ mód } n \\ &= M^{(-t)(p-1)(q-1)+1} \text{ mód } n \\ &\stackrel{?}{=} M \text{ mód } n \\ &= M(\text{si } n \text{ es grande}) \end{aligned}$$

Hay que probar que:

$$M^{(-t)(p-1)(q-1)+1} \text{ mód } n = M \text{ mód } n.$$

Lema. Sean p primo, $a, K \in \mathbb{Z}, K \geq 0$. Entonces:

$$a^{K(p-1)+1} \text{ mód } p = a \text{ mód } p.$$

(generalización del pequeño teorema de Fermat)

Demostración.

$$\begin{aligned} a^{K(p-1)+1} \text{ mód } p &= (((a^K \text{ mód } p)^{p-1} \text{ mód } p) \cdot (a \text{ mód } p)) \text{ mód } p \\ &= (1 \cdot (a \text{ mód } p)) \text{ mód } p \\ &= a \text{ mód } p. \end{aligned}$$

Teorema. Para enteros $M, k \geq 0$ y primos p_1, p_2:

$$M^{k(p_1-1)(p_2-1)+1} \text{ mód } (p_1p_2) = M \text{ mód } (p_1p_2).$$

Demostración. Usar el lema anterior con $a = M, K = k(p_1 - 1)$ y $p = p_2$:

$$M^{k(p_1-1)(p_2-1)+1} \text{ mód } p_2 = M \text{ mód } p_2.$$

También podemos aplicar el lema con $a = M, k = K(p_2 - 1)$ y $p = p_1$:

$$M^{k(p_1-1)(p_2-1)+1} \text{ mód } p_1 = M \text{ mód } p_1.$$

Tenemos:

$$M^{k(p_1-1)(p_2-1)+1} \text{ mód } p_2 = M \text{ mód } p_2 \quad (11.1)$$

$$M^{k(p_1-1)(p_2-1)+1} \text{ mód } p_1 = M \text{ mód } p_1 \quad (11.2)$$

¿Cómo las combinamos para obtener lo deseado?.

Por el Teorema Chino del Resto, como $\text{mcd}(p_1, p_2) = 1$, la siguiente función es 1-1 y sobre:

$$\begin{aligned} \sigma : \mathbb{Z}_{p_1 p_2} &\longrightarrow \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \\ x &\longmapsto (x \text{ mód } p_1, x \text{ mód } p_2) \end{aligned}$$

Es fácil verificar, usando (11.1) y (11.2), que

$$\sigma(M^{k(p_1-1)(p_2-1)+1} \text{ mód } (p_1 p_2)) = \sigma(M \text{ mód } (p_1 p_2))$$

Como σ es 1-1, los argumentos son iguales. □

Volviendo a nuestro escenario criptográfico, tenemos:
$M^{k(p-1)(q-1)+1} \text{ mód } n = M \text{ mód } n$, para $\underline{k \geq 0}$.

Problema:

- ¿Cómo podemos asegurar que $M^{-t\cdot(p-1)(q-1)+1} \text{ mód } n = M \text{ mód } n$?
- ¿Podemos asegurar que $-t \geq= 0$?

El $-t \in \mathbb{Z}$ sale del AEE.
¿Podemos elegir $t \leq 0$, tal que

$$s \cdot e + t \cdot (p-1)(q-1) = 1?$$

Si es así, tendremos finalmente:

$$M^{(-t)(p-1)(q-1)+1} \text{ mód } n = M \text{ mód } n.$$

¿Se puede controlar el signo de t?
No hay necesariamente unicidad en la CLE.

Lema. Sean $a, b \in \mathbb{Z}, b > 1, \text{mcd}(a, b) = 1$.
Existen $s, t \in \mathbb{Z}$, *tales que* $a \cdot s + b \cdot t = 1$ *y* $t \leq 0$.

Demostración. Ejercicio.

11.5. Exponenciación modular

Un ingrediente fundamental para poder implementar eficientemente RSA es el poder calcular $b^e \text{ mód } n$ en forma eficiente. Veremos dos implementaciones de esta función.

11.5.1. Exponenciación modular (*naïve*)

Dados $b, n \in \mathbb{Z}^+, e \in \mathbb{N}$, el siguiente algoritmo calcula $b^e \text{ mód } n$:

Precondición: $b, n \in \mathbb{Z}^+, e \in \mathbb{N}$.
1: $x \leftarrow n$; $y \leftarrow b$; $z \leftarrow 1$;
2: **while** $x \neq 0$ **do**
3: **if** x es impar **then**
4: $z \leftarrow z \cdot y$
5: **end if**
6: $x \leftarrow \lfloor \frac{x}{2} \rfloor$
7: $y \leftarrow y^2$
8: **end while**
9: **return** $z \text{ mód } n$

11.5.2. Complejidad del algoritmo anterior

¿Cuál es la *complejidad* del procedimiento anterior para calcular b^e mód n?

Supondremos que sumar dos números de k bits toma $O(k)$ operaciones y multiplicar dos números de k bits toma $O(k^2)$ operaciones (esta última suposición puede ser mejorada).

Sean b, e y n tres números de k bits c/u. Si llamamos x_i, y_i, z_i a los valores de x, y, z después de i iteraciones, y $|a|$ a la cantidad de bits de a, vemos que:

- $i \leq \lceil \log_2 n + 1 \rceil \leq k$.
- $|y_i| \leq k \cdot 2^i$.
- $|z_i| \leq k(1 + 2 + \cdots + 2^{i-1}) \leq k \cdot 2^i$.

El tiempo que toma, en cada iteración, calcular $z \cdot y$ e y^2 es aproximadamente, $O((k \cdot 2^i)^2) = O(k^2 2^{2i})$.

Al final del loop, cada una de estas operaciones toma $O(k^2 2^{2k})$.

¡Esto es exponencial en k!

¿Es posible mejorar esto? Si en lugar de dejar crecer y y z indiscriminadamente, los mantenemos "chicos" (no más de k bits en cada iteración), las multiplicaciones no serán tan costosas.

11.5.3. Exponenciación modular (más astuta)

El siguiente algoritmo usa la idea anterior:

Precondición: $b, n \in \mathbb{Z}^+, e \in \mathbb{N}$.

1: $x \leftarrow e$; $y \leftarrow b$; $z \leftarrow 1$;
2: **while** $x \neq 0$ **do**
3: **if** x es impar **then**
4: $z \leftarrow z \cdot y$ mód n
5: **end if**
6: $x \leftarrow \lfloor \frac{x}{2} \rfloor$
7: $y \leftarrow y^2$ mód n
8: **end while**
9: **return** z

11.5.4. Complejidad del algoritmo anterior

La complejidad del algoritmo anterior es $O(k \cdot k^2) + O(k \cdot T_m) = O(k^3) + O(k \cdot T_m)$, donde T_m es el tiempo que toma calcular a mód n, siendo a un número de $2k$ bits y n un número de k bits.

Ejercicio. Demuestre que T_m es $O(k^2)$.

Con el resultado del ejercicio anterior, es posible demostrar que la complejidad del algoritmo completo es $O(k^3)$.

11.6. Otros teoremas importantes, y demostraciones pendientes

11.6.1. Teorema fundamental de la aritmética

Teorema. Todo número entero positivo N tiene una única representación como producto de potencias de números primos (excepto por reordenamiento de los factores). En otras palabras, dado $N \in \mathbb{Z}^+$, es posible escribir en forma única

$$N = p_1^{e_1} \cdots p_s^{e_s}$$

con $s \geq 1, p_1, < p_2 < \ldots < p_s$ primos.

Ejemplo. $2200 = 2^3 \cdot 5^2 \cdot 11^1$.

El teorema se demuestra usando los siguientes tres siguientes resultados, más inducción por curso de valores:

Lema. Sean $a, b, c \in \mathbb{Z}$, tales que $a \neq 0$, $a \mid bc$. Si $\text{mcd}(a, b) = 1$, entonces a divide a c.

Corolario. Sean $p, a, b \in \mathbb{Z}, p$ primo. Si $p \mid ab$, entonces $p \mid a$ o $p \mid b$.

Ejemplo. $3 \mid 12 \rightarrow 3 \mid 2 \cdot 6 \rightarrow 3 \mid 2$ o $3 \mid 6$.

El último ingrediente que necesitamos para demostrar el TFA es el siguiente:

Lema. Sean $n \geq 1$ y $p_1, p_2, \cdots, p_n$ primos; p primo.
Si p divide al producto $p_1 \cdots p_n$, entonces, para algún i, $1 \leq i \leq n$, $p = p_i$.

Ejercicio. Usando los dos lemas y el corolario anterior, demuestre el TFA.

11.6.2. Demostración del pequeño teorema de Fermat

Algunos hechos ya conocidos:

1. Para p primo, y todo $a \in \{1, \ldots, p-1\}$, existe $b \in \{1, \ldots, p-1\}$ tal que $ab \equiv 1 \pmod{p}$.
2. Para p primo, y todo $a, c \in \{1, \ldots, p-1\}$, existe $b \in \{1, \ldots, p-1\}$ tal que $ab \equiv c \pmod{p}$.

Además, se puede demostrar lo siguiente:

3. Para p primo, y cada $a \in \{1, \ldots, p-1\}$, la función f_a definida por: $f_a(b) = ab \bmod p$, con $b \in \mathbb{Z}_p^*$, tiene las siguientes propiedades:
   - Está definida en todo $\mathbb{Z}_p^*$.
   - Toma sólo valores en $\mathbb{Z}_p^*$, es decir, nunca $f_a(b) = 0$.
   - Toma todos los valores en $\mathbb{Z}_p$, es decir, es una función sobreyectiva de $\mathbb{Z}_p^*$ en $\mathbb{Z}_p^*$.
   - Luego, es también 1-1 ahí.
   - Es una *permutación* de $\mathbb{Z}_p^*$.

Ejemplo. Consideremos $p = 3$.
$\mathbb{Z}_3^* = \{1, 2\}$
$f_2 = (2 \cdot x) \bmod 3$, $x = 1, 2$.
$f_2(1) = 2, f_2(2) = 1$, una permutación de $\{1, 2\}$.
Fijemos $a \in \mathbb{Z}_p^*$. Como $f_a(1), \ldots, f_a(p-1)$ es una permutación de $\mathbb{Z}_p^*$, se tiene

$$\begin{aligned}
\prod_{b=1}^{p-1} f_a(b) &= \prod_{b=1}^{p-1} b \\
\prod_{b=1}^{p-1} ((ab) \mod p) &= \prod_{b=1}^{p-1} b \quad / \mod p \\
(\prod_{b=1}^{p-1} [(ab) \mod p]) \mod p &= (\prod_{b=1}^{p-1} b) \mod p \\
(\prod_{b=1}^{p-1} (ab)) \mod p &= (\prod_{b=1}^{p-1} b) \mod p
\end{aligned}$$

El último paso a la izquierda lo justifica la propiedad de homomorfismo.
Luego:

$$((\prod_{b=1}^{p-1} b) \cdot a^{p-1}) \mod p = (\prod_{b=1}^{p-1} b) \mod p$$

$$(((\prod_{b=1}^{p-1} b) \mod p) \cdot ((a^{p-1}) \mod p)) \mod p = (\prod_{b=1}^{p-1} b) \mod p.$$

Ahora, $(\prod_{b=1}^{p-1} b) \mod p \in \mathbb{Z}_p$ y es distinto de 0, es decir, pertenece a $\mathbb{Z}_p^*$ (si lo fuera, el producto sería divisible por p, luego, uno de los factores tendría que ser divisible por p, lo que no es posible pues todos son menores que p).

Otra manera de justificarlo, con un resultado general de la teoría de cuerpos ($\mathbb{Z}_p$ es un cuerpo): en un cuerpo, "no hay divisores del cero", es decir, si $a \cdot b = 0$, entonces $a = 0$ o $b = 0$.

Entonces, existe el inverso multiplicativo del producto en $\mathbb{Z}_p^*$. Podemos multiplicar a ambos lados por él, como al lado izquierdo tenemos la multiplicaci'on en $\mathbb{Z}_p^*$, queda:

$$a^{p-1} \mod p = 1.$$

El pequeño teorema de Fermat sale como consecuencia inmediata de un resultado general de la teoría de grupos:.

Teorema (Lagrange). *Para todo grupo finito* $[\mathrm{G}, *, e]$ *y para todo* $g \in \mathrm{G}$*:*

$$g^{|\mathrm{G}|} = e.$$

Caso Particular: $[\{1, \ldots, p-1\}, \cdot \mod p, 1]$ es un grupo, su cardinalidad es $p-1$, luego, para todo $a \in \{1, \ldots, p-1\}$, se tiene $a^{p-1} = 1$.

¡Notar el poder de resultados generales del álgebra abstracta, en este caso, de la teoría de grupos!

11.6.3. El (gran) teorema de Fermat

Dados $x, y, z \in \mathbb{Z} - \{0\}$, $n \in \mathbb{N}$, $n > 2$, se tiene

$$x^n + y^n \neq z^n.$$

11.6.4. Demostración del teorema chino de los restos

Unicidad

Supongamos $0 \leq x, x' < ab$, y que:

1. $x \mod a = u$
2. $x \mod b = v$
3. $x' \mod a = u$
4. $x' \mod b = v$

Hay que demostrar que $x = x'$.

De 1 y 3, $x \mod a = x' \mod a$, de donde $a \mid (x - x')$. Así, $(x - x') = c \cdot a$ para algún c entero.

Análogamente, $b \mid (x - x')$, de donde b divide a $c \cdot a$. Pero $\mathrm{mcd}(a, b) = 1$, de donde $b \mid c$, es decir, $c = k \cdot b$.

Luego: $(x - x') = k \cdot a \cdot b$. Entonces, ab divide a $(x - x')$. Pero x, x' son no negativos y $x, x' < ab$, por lo que necesariamente $x - x' = 0$.

Existencia

Daremos una demostración constructiva, es decir, un procedimiento para construir x.

Este procedimiento puede entenderse como inspirado en el proceso de interpolación de Lagrange: si se quiere hallar un polinomio $p(x)$ con valores $v_1, v_2, \ldots, v_n$ en los puntos $x_1, x_2, \ldots, x_n$, podemos tomar

$$p(x) = \sum_{i=1}^{n} v_i p_i(x),$$

donde p_i es un polinomio que vale 1 en x_i y 0 en todos los x_j con $j \neq i$.

Ejercicio. Encuentre una fórmula para $p_i(x)$.

Sabemos que existe una CLE: $s \cdot a + t \cdot b = 1$, con $s, t \in \mathbb{Z}$.

Esta CLE es *computable eficientemente*.

Los valores sa y tb juegan el rol de los p_i en la interpolación de Lagrange: sa y tb satisfacen

$$\begin{aligned} tb \mod a &= 1, \\ tb \mod b &= 0, \\ sa \mod a &= 0, \\ sa \mod b &= 1. \end{aligned}$$

De aquí, obtenemos que

$$x = (sav + tbu) \mod ab$$

(que es computable eficientemente) satisface las condiciones pedidas.