

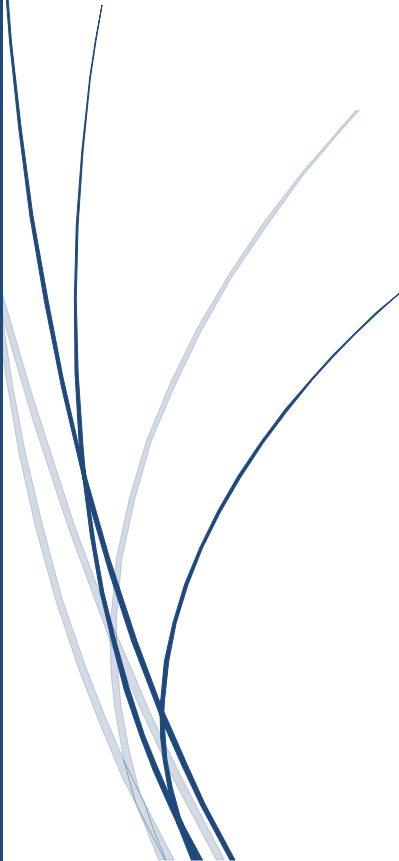


MATEMÁTICAS DISCRETAS

ALGEBRAS BOOLEANAS Y
CIRCUITOS LÓGICOS

**UNIVERSIDAD DE SAN
BUENAVENTURA CALI**

**FACULTAD DE INGENIERÍAS - PROGRAMA DE ING.
SISTEMAS**



Juan Pablo Silvestre (99127), Tomas Mancera (98649),
Sebastián López (97500), Alejandro Salazar (97502), Juan
Manuel Padilla (97196)

UNIVERSIDAD SAN BUENAVENTURA DE CALI

UNIDAD 1

ALGEBRAS BOOLEANAS Y CIRCUITOS LÓGICOS

Axiomas

B1) Leyes Conmutativas: Para todo elemento $x \in B$ se cumple:

$$1) x + y = y + x \quad 2) x * y = y * x$$

B2) Leyes Distributivas. Cada operación es distributiva con respecto a la otra:
Para todo elemento $x, y, \in B$, se cumple que:

$$1) x + (y * z) = (x + y) * (x + z) \quad 2) x * (y + z) = (x * y) + (x * z)$$

B3) Leyes Modulativas: Para todo $x \in B$, existen dos elementos diferentes 0 y 1 en B tales que:

$$1) x + 0 = 0 + x = x \quad 2) x * 1 = 1 * x = x$$

B4) Leyes del Complemento: Para todo $x \in B$ existe un elemento $x' \in B$ tal que:

$$1) x + x' = 1 \quad 2) x * x' = 0$$

Álgebra Binaria de Boole.

Sea el conjunto $B = \{0, 1\}$ en el cual se definen las operaciones +(disyunción) y *(conjunción) de acuerdo a las siguientes tablas:

DISYUNCIÓN (+)

+	0	1
0	0	1
1	1	1

CONJUNCIÓN (*)

*	0	1
0	0	0
1	0	1

Demostración de ley distributiva $x + (y * z) = (x + y) * (x + z)$

				A	B			
x	y	z	y * z	x + (y * z)	(x + y)	(x + z)	(x + y) * (x + z)	
1	1	1	1	1	1	1	1	
1	1	0	0	1	1	1	1	
1	0	1	0	1	1	1	1	
1	0	0	0	1	1	1	1	
0	1	1	1	1	1	1	1	
0	1	0	0	0	1	0	0	
0	0	1	0	0	0	1	0	
0	0	0	0	0	0	0	0	

Observación: Columna A y B son iguales, por consiguiente, se demuestra la proposición.

RESULTADOS QUE SE DERIVAN DE LOS AXIOMAS DE LAS ÁLGEBRAS DE BOOLE

EXPRESIONES BOOLEANAS (TEOREMAS):

TEOREMA 1. Leyes de Idempotencia. Todo elemento de una Álgebra Booleana es idempotente. Para todo elemento $x \in B$; se cumple que:

$$1) x + x = x \quad 2) x * x = x.$$

TEOREMA 2. Leyes de acotamiento. Para todo $x \in B$, se verifica que:

$$1) x + 1 = 1 \quad 2) x * 0 = 0.$$

TEOREMA 3. Leyes de absorción. Para todo $x, y \in B$, se verifica que:

$$1) x + (x * y) = x \quad 2) x * (x + y) = x.$$

TEOREMA 4. Unicidad del complemento. Para cada $x \in B$, siendo B un Álgebra de Boole, el complemento de x , denotado por x' , es único.

TEOREMA 5. Leyes de De Morgan. Para todo $x, y \in B$, se verifica que:

$$1) (x + y)' = x' * y' \quad 2) (x * y)' = x' + y'.$$

TEOREMA 6. Leyes asociativas. Para todo $x, y, z \in B$, se verifica que:

$$9 a) (x + y) + z = x + (y + z) \quad y \quad 9 B) (x * y) * z = x * (y * z).$$

TEOREMA 7. Ley de involución. Para cada $x \in B$, $(x')' = x$.

TEOREMA 8. Leyes para el 0 y el 1: $0' = 1$ y $1' = 0$.

DEFINICIÓN DE DUAL. El dual de una expresión E de un Álgebra Booleana, es la expresión que resulta a partir de E intercambiando $+$ por $*$ y 0 por 1 y recíprocamente, en cada operación de estos símbolos.

Ejemplo: Dada la ecuación $E1 = x + xz' = x$, su dual es la ecuación $E1^d = x * (x + z') = x$.

ALGUNOS EJERCICIOS RESUELTOS (5.1)

3. Escriba el dual de cada una de las siguientes expresiones.

3.1. $(x + y) (x + 1) = x + xy + y.$

Solución: $xy + x0 = x (x + y) y.$

3.2. $(x' + y')' = xy.$

Solución: $(x'y')' = x + y.$

3.3. Si $x + y = x + z$ y $x' + y = x' + z$, entonces $y = z$.

Solución: Si $xy = xz$ Y $x'y = x'z$, entonces $y = z$.

3.5. $x + x (y + 1) = x.$

Solución: $x*x + (y * 0) = x.$

3.7. $a (a' + b) = ab.$

Solución: $a + (a'b) = a + b.$

3.8. $(x + 1) (x + 0) = x.$

Solución: $x*0 + x*1 = x.$

3.9. $(x + y) (y + z) = xz + y.$

Solución: $xy + yz = (x + z) y.$

3.10. $b \{a + [a' (b + b')]\} = b.$

Solución: $b + (a*a') + (b*b') = b.$

4. Verifique cada uno de los enunciados del ejercicio 3.

3.1. $(x + y)(x + 1) = x + xy + y.$

Solución:
$$\begin{aligned} &= xx + x.1 + yx + y.1. \\ &= x + x + yx + y. \\ &= x + yx + y. \end{aligned}$$

3.2. $(x' + y')' = xy.$

Solución:
$$= xy.$$

3.5. $x + x(y + 1) = x.$

Solución:
$$\begin{aligned} &= x + xy + x. \\ &= x + xy. \\ &= x. \end{aligned}$$

3.7. $a(a' + b) = ab.$

Solución:
$$\begin{aligned} &= a*a' + ab. \\ &= 0 + ab. \\ &= ab. \end{aligned}$$

3.8. $(x + 1)(x + 0) = x.$

Solución:
$$\begin{aligned} &= x*x + x*0 + 1*x + 1*0. \\ &= x + 0 + x 0. \\ &= x. \end{aligned}$$

3.9. $(x + y)(y + z) = xz + y.$

Solución:
$$\begin{aligned} &= xy + xz + y*y + yz. \\ &= xy + xz + y + yz. \\ &= xz + y. \end{aligned}$$

$$3.10. b \{a + [a' (b + b')]\} = b.$$

Solución:

$$\begin{aligned}
 &= b \{a + [a' (1)]\}. \\
 &= b (a + a'). \\
 &= b (1). \\
 &= b.
 \end{aligned}$$

5. Verifique las siguientes ecuaciones:

$$\begin{aligned}
 5.1. xy' &= (x' + y)'. \\
 &= xy'.
 \end{aligned}$$

Leyes de D 'Morgan.

$$\begin{aligned}
 5.2. x (yz)' &= xy' + xz'. \\
 &= x (y' + z'). \\
 &= xy' + xz'.
 \end{aligned}$$

Leyes de D 'Morgan.
Ley distributiva.

$$\begin{aligned}
 5.3. (x + y) (z + w) &= zx + zy + wx + wy. \\
 &= z (x + y) + w (x + y). \\
 &= (z + w) (x + y). \\
 &= (x + y) (z + w).
 \end{aligned}$$

Factor común.
Factor común.
Leyes conmutativas.

6. Pruebe o dé un contraejemplo, si las ecuaciones siguientes son o no verdaderas:

$$\begin{aligned}
 6.1. x'y' &= x + y. \\
 (x + y)' &\neq x + y.
 \end{aligned}$$

Aplicando leyes de D 'Morgan, se verifica que la ecuación no es verdadera.

6.2. $x' (yz + xyz) = yz.$

$x'yz + x'xyz = yz.$

$x'yz + 0yz = yz.$

$x'yz + 0 = yz.$

$x'yz \neq yz.$

Aplicando ley distributiva, ley del Complemento, y ley modulativa, se verifica que la ecuación no es verdadera.

6.3. $(x'y + xz')' = (x + y') (x + z').$

$(x'y)' (xz')' = (x + y') (x + z').$

$(x + y') (x' + z') \neq (x + y') (x + z').$

Aplicando leyes de D 'Morgan, se verifica que la ecuación no es verdadera.

6.4. $(x + y) (z' + w)' (zy') = 0.$

$(x + y) (zw') (zy') = 0.$

$(x + y) (zw'y') = 0.$

$(xzw'y') + (yzw'y') = 0.$

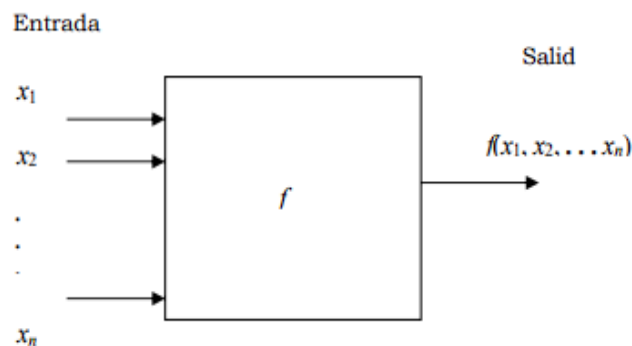
$(xzw'y') + 0 = 0.$

$(xzw'y') \neq 0.$

Aplicando leyes de D 'Morgan, leyes de Idempotencia, leyes Distributivas, leyes del Complemento, y, por último, leyes Modulativas, se comprueba que la ecuación no es verdadera.

CIRCUITOS LÓGICOS.

Representación de las logic gates, donde su mayoría toma 2 inputs y genera 1 output, los inputs son representados por 0s y 1s, donde el 0 representa la "ausencia" de corriente, y por deducción, 1 representa el flujo de la misma. Teniendo en cuenta esto, la cantidad de combinaciones como input que puede tener una logic gate puede verse reflejada de la forma 2^n , Donde n representa el número de casos a tener en cuenta.



CIRCUITOS DE CONMUTACIÓN (O DE CONMUTADORES)

Un circuito eléctrico de interruptores normalmente contiene alguna fuente de energía (una pila o batería), un dispositivo de salida (por ejemplo, una bombilla), y uno o más interruptores o “switches”, todos ellos conectados por alambres. El funcionamiento se define por On(encendido) y off(apagado).

Cuando un interruptor x está en la posición cerrado (On) permite el paso de la corriente, se escribe $x = 1$, y cuando está en la posición abierta (off), no hay paso de corriente, se escribe $x = 0$.

INTERRUPTOR CERRADO (ON)



$$x = 1$$

INTERRUPTOR ABIERTO (OFF)



$$x = 0.$$

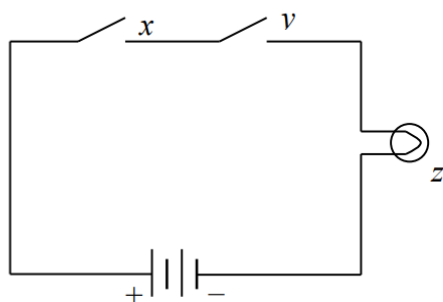
Existen dos formas básicas para interconectar interruptores: conexión en serie y conexión en paralelo.

CIRCUITO EN SERIE (CIRCUITO AND).

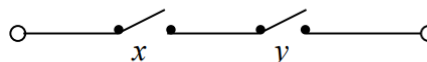
Se caracteriza por tener un paso de corriente z si, y solo si, x y y se encuentran cerrados, es decir $x=1$ y $y=1$. Esta expresión se denota como $z=xy$.

x	y	$z=xy$
1	1	1
1	0	0
0	1	0
0	0	0

REPRESENTACIÓN SIMPLIFICADA DEL CIRCUITO



Circuito en serie



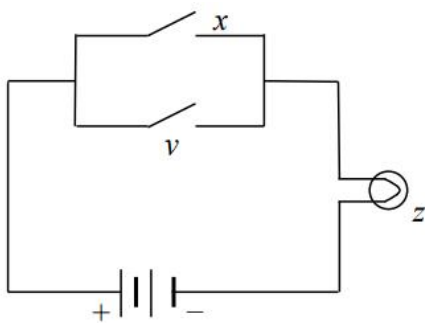
Circuito simplificado

CIRCUITO EN PARALELO (CIRCUITO OR).

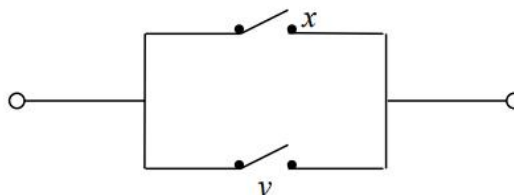
A diferencia del AND, z solo necesita al menos una compuerta en 1, ya sea $x=1$ o $y=1$. Esta expresión se denota como $z = x + y$.

x	y	$z = x + y$
1	1	1
1	0	1
0	1	1
0	0	0

Representación simplificada del circuito.

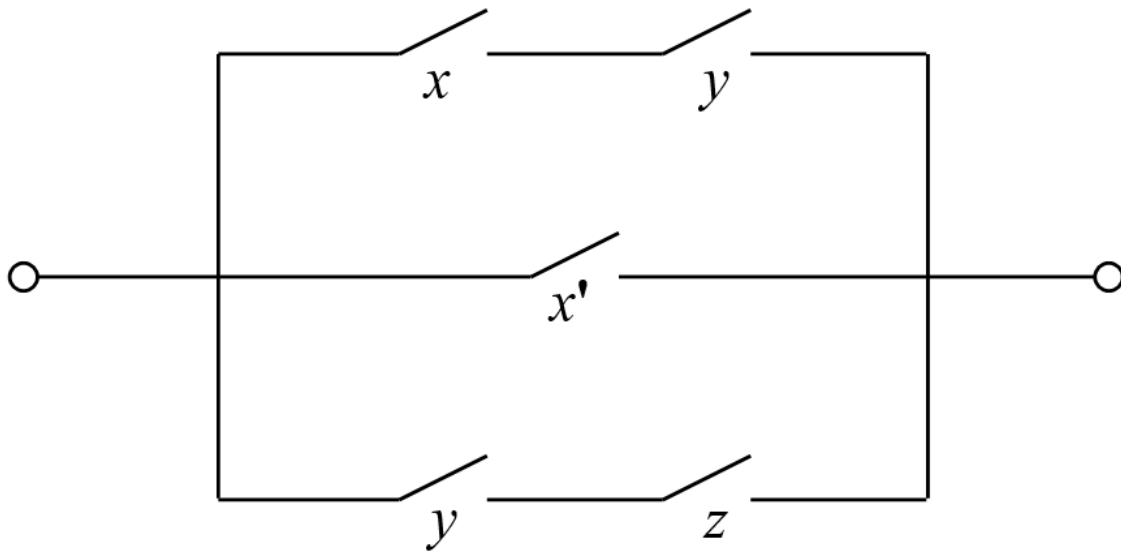


Circuito en paralelo



Circuito simplificado

Ejemplo:

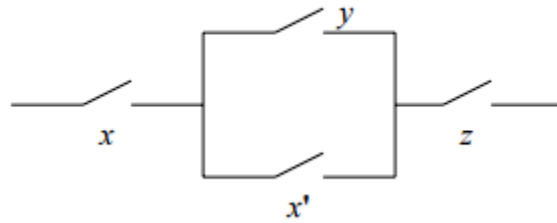


x	y	z	xy	x'	yz	xy+x'+yz
1	1	1	1	0	1	1
1	1	0	1	0	0	1
1	0	1	0	0	0	0
1	0	0	0	0	0	0
0	1	1	0	1	1	1
0	1	0	0	1	0	1
0	0	1	0	1	0	1
0	0	0	0	1	0	1

EJERCICIOS 5.2 RESUELTOS

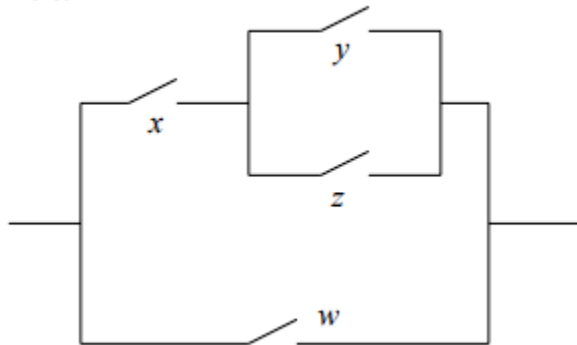
1. Expresé cada circuito en forma simbólica y construya su correspondiente tabla conmutadora:

1.1.



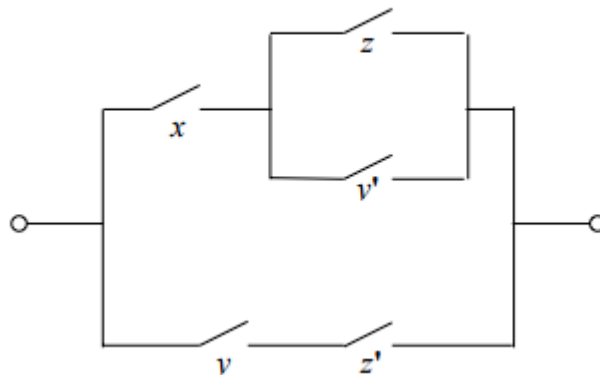
solución: $x^*(y + x') z$.

1.2.



solución: $(x^* (y + z) + w)$.

1.3.

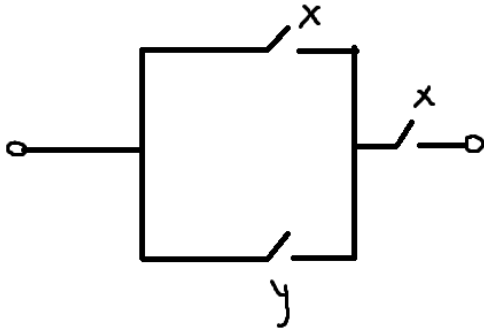


solución: $(x (z + v') + (v z'))$.

2. Represente las expresiones de los siguientes ejercicios como circuitos conmutadores y escriba las correspondientes tablas conmutadoras:

2.1. $(x + y) x$

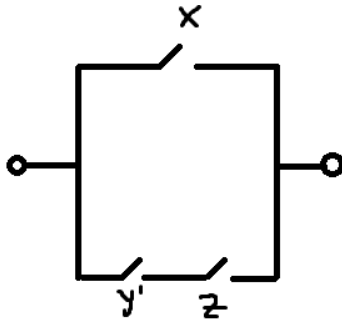
Solución:



X	Y	X + Y	(X + Y) x
1	1	1	1
1	0	1	1
0	1	1	0
0	0	0	0

2.2. $x + (y' z)$

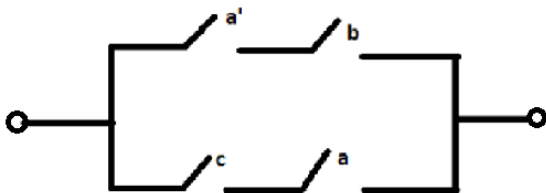
Solución:



X	Y	Z	Y'	Y'Z	X + (Y'Z)
1	1	1	0	0	1
1	1	0	0	0	1
1	0	1	1	1	1
1	0	0	1	0	1
0	1	1	0	0	0
0	1	0	0	0	0
0	0	1	1	1	1
0	0	0	1	0	0

2.3. $(a' b) + (c a)$

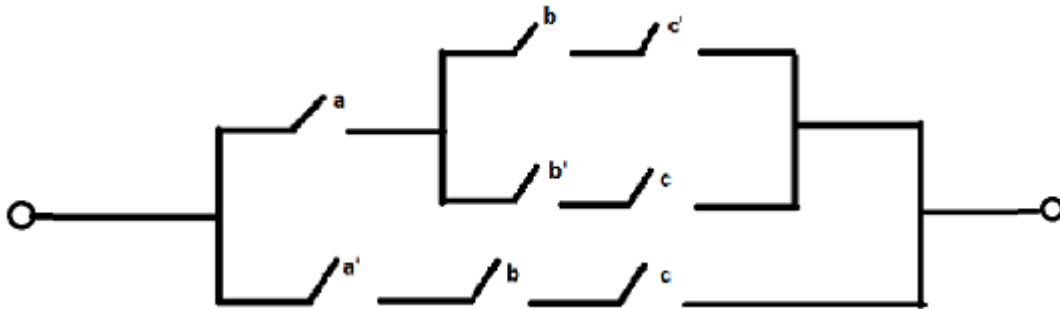
Solución:



a	b	c	a'	a'b	c a	a'b + c a
1	1	1	0	0	1	1
1	1	0	0	0	0	0
1	0	1	0	0	1	1
1	0	0	0	0	0	0
0	1	1	1	1	0	1
0	1	0	1	1	0	1
0	0	1	1	0	0	0
0	0	0	1	0	0	0

2.4. $\{a [(b c') + (b' c)]\} + (a' b c)$

Solución:



X						Y		
a	b	c	b' c	b c'	b c' + b' c	a ((b c') + (b' c))	a' b c	X + Y
1	1	1	0	0	0	0	0	0
1	1	0	0	1	1	1	0	1
1	0	1	1	0	1	1	0	1
1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	1	1
0	1	0	0	1	1	0	0	0
0	0	1	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0

Compuertas Lógicas.

AND GATE: Toma de valor 2 inputs ($x, y \in B$), y produce un output ($z \in B$), el cual es 1 si, y sólo si $x = 1$ y $y = 1$.

Observación: esta compuerta también puede tomar más de 2 inputs.

Input X	Input Y	Output Z = X * Y
1	1	1
1	0	0
0	1	0
0	0	0



OR GATE: Toma de valor 2 inputs ($x, y \in B$), y produce un output ($z \in B$), el cual es 1 si;

$x = 1$ o $y = 1$.

Input X	Input Y	Output $Z = X + Y$
1	1	1
1	0	1
0	1	1
0	0	0

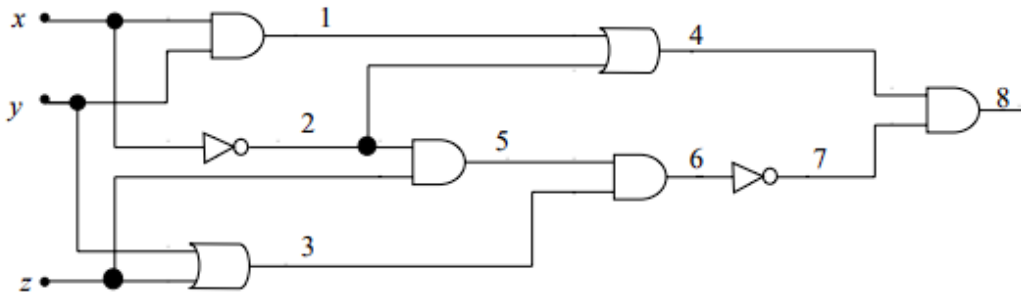


NOT GATE: toma de input un solo valor de $x \in B$, y tiene como output el complemento de x (x').

Input	Output
x	x'
1	0
0	1



EJERCICIO: Determinar expresión booleana del siguiente circuito:



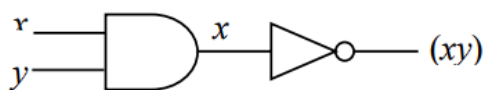
Solución: $[(xy') + x'] * [(x'z) (z + y)]'$.

OTRAS COMPUERTAS LÓGICAS

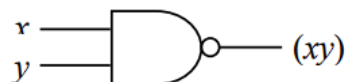
NAND GATE: Se obtiene sacando el complemento del output de una AND gate, también se le conoce como operación de Pierce.

Input X	Input Y	Output AND = XY	Output NAND = $(XY)'$
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	1

Esta compuerta se representaría con la siguiente figura:



Compuerta NAND

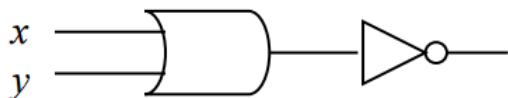


Compuerta NAND simplificada

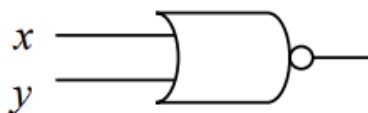
NOR GATE: Dominada también como **operación de Pierce**, se obtiene negando el output de una OR gate:

Input X	Input Y	Output OR = $X + Y$	Output NOR = $(X + Y)'$
1	1	1	0
1	0	1	0
0	1	1	0
0	0	0	1

Esta compuerta se representaría con la siguiente figura:



Compuerta NOR

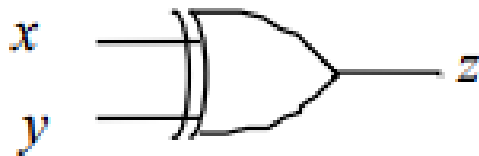


Compuerta NOR simplificada

XOR GATE (Or exclusiva): Corresponde a la disyunción excluyente, lo cual significa que el output es 1 si, y sólo si, UNO de los inputs es 1. Su expresión booleana quedaría de la siguiente forma: $f(x, y) = x \oplus y = xy' + y'x$.

X	Y	X'Y	XY'	Output XOR = $X \oplus Y$
1	1	0	0	0
1	0	0	1	1
0	1	1	0	1
0	0	0	0	0

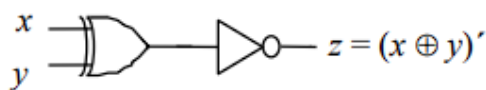
Esta compuerta se representaría con la siguiente figura:



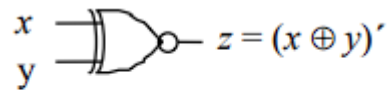
XNOR GATE: Se obtiene sacando el complemento del output de una XOR gate, lo cual quedaría de la siguiente forma.

X	Y	X'Y	XY'	Output XOR = $X \oplus Y$	Output XNOR = $(X \oplus Y)'$
1	1	0	0	0	1
1	0	0	1	1	0
0	1	1	0	1	0
0	0	0	0	0	1

Esta compuerta se representaría con la siguiente figura:



Compuerta XNOR

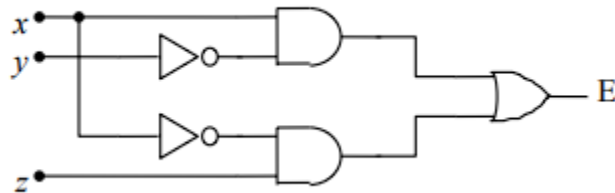


Compuerta XNOR simplificada

ALGUNOS EJERCICIOS RESUELTOS (5.3)

1. En los siguientes ejercicios escriba las expresiones Booleanas que representan los circuitos combinatorios y elabore la tabla lógica correspondiente:

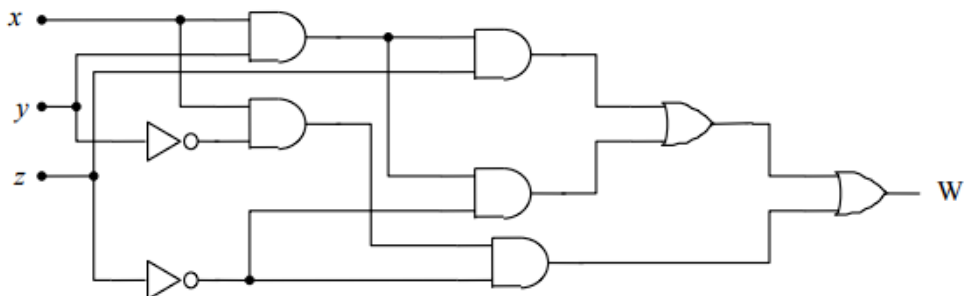
1.1.



x	y	z	x'	y'	xy'	x'z	xy' + x'z
1	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	0	1	0	1	1	0	1
1	0	0	0	1	1	0	1
0	1	1	1	0	0	1	1
0	1	0	1	0	0	0	0
0	0	1	1	1	0	1	1
0	0	0	1	1	0	0	0

Solución: $(xy') + (x'z) = E$

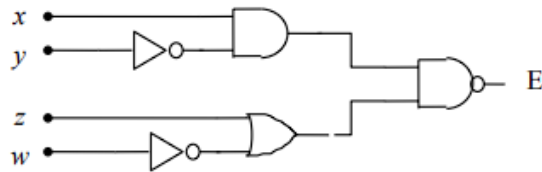
1.3.



Solución: $[(xyz) + (xy'z')]' + (xyz') = w$

						A	B	C	D
x	y	z	y'	z'	xyz	xyz'	A+B	xy'z'	C+D
1	1	1	0	0	1	0	1	0	1
1	1	0	0	1	0	1	1	0	1
1	0	1	1	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1	1
0	1	1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0

1.6.



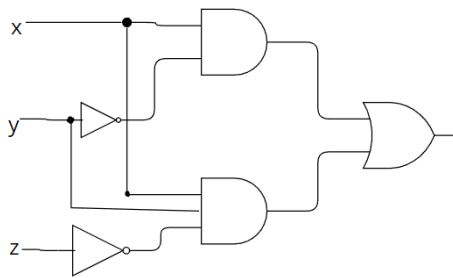
Solución: $(xy' * (z + w'))' = E$

							A	B	C
x	y	z	w	y'	w'	xy'	z + w'	A+B	C'
1	1	1	1	0	0	0	1	1	0
1	1	1	0	0	1	0	1	1	0
1	1	0	1	0	0	0	0	0	1
1	1	0	0	0	1	0	1	1	0
1	0	1	1	1	0	1	1	1	0
1	0	1	0	1	1	1	1	1	0
1	0	0	1	1	0	1	0	1	0
1	0	0	0	1	1	1	1	1	0
0	1	1	1	0	0	0	1	1	0
0	1	1	0	0	1	0	1	1	0
0	1	0	1	0	0	0	0	0	1
0	1	0	0	0	1	0	1	1	0
0	0	1	1	1	0	0	1	1	0
0	0	1	0	1	1	0	1	1	0
0	0	0	1	1	0	0	0	0	1
0	0	0	0	1	1	0	1	1	0

2. Encuentre el circuito combinatorio correspondiente para cada una de las expresiones Booleanas de los siguientes ejercicios y elabore la tabla lógica correspondiente:

2.1. $xy' + xyz'$

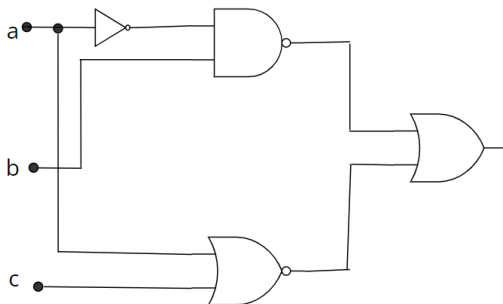
Solución:



		A		B				
x	y	z	y'	z'	xy'	xyz'	A+B	
1	1	1	0	0	0	0	0	
1	1	0	0	1	1	0	1	
1	0	1	1	0	1	0	1	
1	0	0	1	1	1	0	1	
0	1	1	0	0	0	0	0	
0	1	0	0	1	0	0	0	
0	0	1	1	0	0	0	0	
0	0	0	1	1	0	0	0	

2.3. $(a'b)' + (a + c)'$

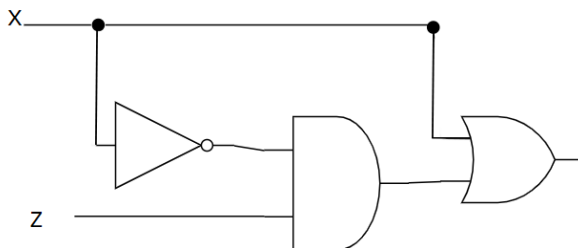
Solución:



		A		B				
a	b	c	a'	a'b	(a'b)'	a+c	(a+c)'	A+B
1	1	1	0	0	1	1	0	1
1	1	0	0	0	1	1	0	1
1	0	1	0	0	1	1	0	1
1	0	0	0	0	1	1	0	1
0	1	1	1	1	0	1	0	0
0	1	0	1	1	0	0	1	1
0	0	1	1	0	1	1	0	1
0	0	0	1	0	1	0	1	1

2.4. $x + x'z$

Solución:



X	Z	X'	X'Z	X + X'Z
1	1	0	0	1
1	0	0	0	1
0	1	1	1	1
0	0	1	0	0

3. Determine la expresión de Boole en forma de suma de productos de cada uno de los siguientes ejercicios y trace el circuito combinatorio correspondiente:

3.4

Solución:

x	y	z	$f(x, y, z)$	
1	1	1	1	xyz
1	1	0	0	
1	0	1	0	
1	0	0	1	xy'z'
0	1	1	0	
0	1	0	0	
0	0	1	0	x'y'z'
0	0	0	1	

$$(xyz) + (xy'z') + (x'y'z')$$

$$(xyz) + y'z'(x+x')$$

$$xyz + y'z'$$

4. Escriba cada expresión de Boole $f(x, y, z)$ en forma suma de productos (o forma normal disyuntiva), y luego en forma completa de suma de productos:

4.2. $(x + y) (x' + y')$.

Solución: $(xy') + (x'y)$.

x	y	$x + y$	x'	y'	$x' + y'$	$(x + y) (x' + y')$
1	1	1	0	0	0	0
1	0	1	0	1	1	1
0	1	1	1	0	1	1
0	0	0	1	1	1	0

4.3. $(x' + y)' + y'z$.

Solución:

x	y	z	x'	$(x' + y)$	$(x' + y)'$	$y'z$	$(x' + y)' + y'z$	
1	1	1	0	1	0	0	0	
1	1	0	0	1	0	0	0	
1	0	1	0	0	1	1	1	$x'y'z$
1	0	0	0	0	1	0	1	$xy'z'$
0	1	1	1	1	0	0	0	
0	1	0	1	1	0	0	0	
0	0	1	1	1	0	1	1	$x'y'z$
0	0	0	1	1	0	0	0	

$$\begin{aligned}
 & (xy'z) + (xy'z') + (x'y'z) \\
 &= y' (xz + xz' + x'z') \\
 &= y' (x(z + z') + x'z') \\
 &= y' (x + x'z') \\
 &= y' ((x + x') (x + z')) \\
 &= y' (x + z') \\
 &= y'x + y'z'.
 \end{aligned}$$

4.6. $(x'y)' (x' + xyz')$

Solución:

x	y	z	x'	$(x'y)$	$(x'y)'$	z'	xyz'	$(x' + xyz')$	$(x'y)' (x' + xyz')$	
1	1	1	0	0	1	0	0	0	0	
1	1	0	0	0	1	1	1	0	0	
1	0	1	0	0	1	0	0	0	0	
1	0	0	0	0	1	1	0	0	0	
0	1	1	1	1	0	0	0	1	0	
0	1	0	1	1	0	1	0	1	0	
0	0	1	1	0	1	0	0	1	1	$x'y'z$
0	0	0	1	0	1	1	0	1	1	$x'y'z'$

Formal normal disyuntiva

$$\begin{aligned}
 & (x'y'z) + (x'y'z') \\
 &= x'y' (z + z') \\
 &= x'y'.
 \end{aligned}$$

Ecuación original

$$(x' y)' (x' + xyz').$$

$$= (x + y') (x' + xyz').$$

$$= xx' + xyz' + y'x' + xyz'y'.$$

$$= 0 + xyz' + y'x' + 0.$$

$$= xyz' + y'x'.$$

UNIDAD 2

PRODUCTO CARTESIANO

Definición. Sean A y B dos conjuntos no nulos, se denomina producto cartesiano de A y B, que se denota $A \times B$, al conjunto formado por las parejas ordenadas (a, b) tal que: $a \in A$ y $b \in B$. Esto es:

$$A \times B = \{(a, b) / a \in A \text{ y } b \in B\}$$

$$\text{sean } A = \{a, b, c, d\} \text{ y } B = \{1, 2, 3\}$$

Hallar: $A \times B$.

Solución:

$$A \times B = \{(a, 1), (b, 1), (c, 1), (d, 1), (a, 2), (b, 2), (c, 2), (d, 2), (a, 3), (b, 3), (c, 3), (d, 3)\}$$

$$B \times A = \{(1, a), (1, b), (1, c), (1, d), (2, a), (2, b), (2, c), (2, d), (3, a), (3, b), (3, c), (3, d)\}$$

RELACIONES BINARIAS Y SUS PROPIEDADES

Sean A y B dos conjuntos, una relación binaria entre el conjunto A en B es un subconjunto

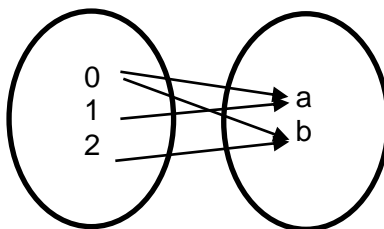
$A \times B$.

En palabras binarias, existe un conjunto R que proviene de la relación $A \times B$, que está compuesto de pares ordenados en los que: El primer elemento de cada par ordenado pertenece al conjunto A, y el segundo elemento de B.

Se usa la notación $a R b$ para $(a, b) \in R$; y $a \not R b$ para $(a, b) \notin R$.

Ejemplo: Sea $A = \{0, 1, 2\}$ y $B = \{a, b\}$, Por consiguiente: $\{(0, a), (0, b), (1, a), (2, b)\}$.

Se sabe que: $0 R a$, pero $1 \not R b$.



RELACIONES

Sean A y B dos conjuntos, una relación binaria entre el conjunto A en B es un subconjunto

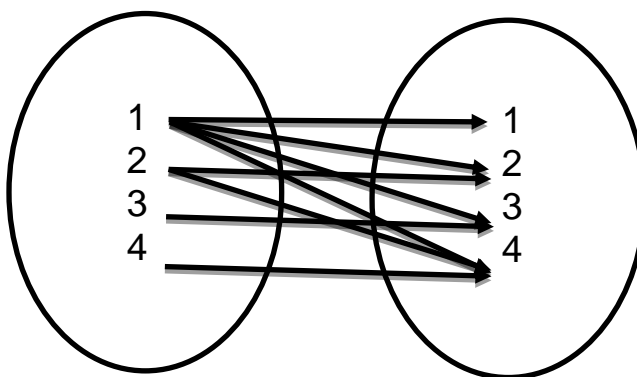
$A \times B$.

Relaciones en un conjunto: Una relación en un conjunto A es una relación de A en A

Ejemplo: Sea $A = \{1, 2, 3, 4\}$ ¿Qué pares ordenados están en la relación $R = \{(a, b) \mid a \text{ divide a } b\}$?

Solución: Como (a, b) está en R si, si y sólo si, a y b son enteros positivos menores o iguales que 4 tales que a divide a b.

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



Ejemplo: Considerar las siguientes relaciones en el conjunto de los enteros:

$$R_1 = \{(a, b) \mid a \leq b\}.$$

$$R_2 = \{(a, b) \mid a > b\}.$$

$$R_3 = \{(a, b) \mid a = b \text{ ó } a = -b\}.$$

$$R_4 = \{(a, b) \mid a = b\}.$$

$$R_5 = \{(a, b) \mid a = b + 1\}.$$

$$R_6 = \{(a, b) \mid a + b \leq 3\}.$$

¿Cuáles relaciones contienen a cada uno de los pares $(1,1)$, $(1, 2)$, $(2,1)$, $(1,-1)$ y $(2,2)$?

Solución:

$(1,1)$ Este par está en: R_1, R_3, R_4, R_6 .

$(1, 2)$ Este par está en: R_1, R_6 .

$(2, 1)$ Este par está en: R_2, R_5, R_6 .

$(1, -1)$ Este par está en: R_2, R_3, R_6 .

$(2, 2)$ Este par está en: R_1, R_3, R_4 .

Ejemplo: ¿Cuántas relaciones hay en un conjunto de elementos?

Solución: Una relación en un conjunto A es un subconjunto de $A \times A$. Como $A \times A$ tiene n^2 elementos si A tiene n elementos y un conjunto m elementos tiene 2^m subconjuntos, hay 2^{n^2} subconjuntos de $A \times A$. Por tanto, hay 2^{n^2} relaciones en un conjunto con n elementos. Por ejemplo, hay $2^{3^2} = 2^9 = 512$ relaciones en el conjunto $\{a, b, c\}$.

Propiedades de las relaciones: En algunas relaciones, un elemento está siempre relacionado consigo mismo. Por ejemplo, sea R la relación en el conjunto de todas las personas formada por aquellos pares (x, y) tales que X y Y tienen la misma madre, y el mismo padre. Entonces, $x R x$ para cada persona x .

Definición: Se dice que una relación R en un conjunto A es reflexiva si $(a, a) \in R$ para cada elemento $a \in A$.

Considérense las siguientes relaciones en $\{1,2,3,4\}$:

$$R_1 = \{(1,1), (1,2), (2,1), (2,2), (3,4), (4,1), (4,4)\}$$

$$R_2 = \{(1,1), (1,2), (2,1)\}$$

$$R_3 = \{(1,1), (1,2), (1,4), (2,1), (2,2), (3,3), (4,1), (4,4)\}$$

$$R_4 = \{(2,1), (3,1), (3,2), (4,1), (4,1), (4,3)\}$$

$$R_5 = \{(1,1), (1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)\}$$

$$R_6 = \{(3,4)\}$$

¿Cuáles relaciones son reflexivas?

Solución:

Tanto R_3 y R_5 contienen todos los pares (a, a) lo cual sería $(1,1)$, $(2,2)$, $(3,3)$ y $(4,4)$

RELACIONES SIMÉTRICAS Y ANTISIMÉTRICAS.

Definición Simétrica: Se dice que una relación R en un conjunto A es simétrica si para cualquiera $a, b \in A$ se tiene que $(b, a) \in R$ siempre que $(a, b) \in R$.

Definición Antisimétrica: Se dice que una relación R en un conjunto A es antisimétrica si para cualquiera $a, b \in A$ se tiene que se tiene que $(a, b) \in R$ y $(b, a) \in R$ sólo si $a = b$. Para esta definición sería lo contrario a simétrica puesto que la simétrica deben existir en R : (a, b) y (b, a) , en cambio, en la antisimétrica necesita que en R : exista (a, b) y no exista (b, a) , ó que exista (b, a) y no exista (a, b) .

Ejemplo: Del siguiente ejercicio, ¿qué relaciones son simétricas y cuáles son antisimétricas?

$$R_1 = \{(1,1), (1,2), (2,1), (2,2), (3,4), (4,1), (4,4)\}$$

$$R_2 = \{(1,1), (1,2), (2,1)\}$$

$$R_3 = \{(1,1), (1,2), (1,4), (2,1), (2,2), (3,3), (4,1), (4,4)\}$$

$$R_4 = \{(2,1), (3,1), (3,2), (4,1), (4,1), (4,3)\}$$

$$R_5 = \{(1,1), (1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)\}$$

$$R_6 = \{(3,4)\}$$

Solución:

Simétricas: R_2 y R_3

Antisimétricas: R_4 , R_5 y R_6

Relaciones transitivas:

Definición: Se dice que una relación R en un conjunto A es transitiva si para cualquiera $a, b, c \in A$ tales que $(a, b) \in R$ y $(b, c) \in R$ se tiene que $(a, c) \in R$.

$$R_1 = \{(1,1), (1,2), (2,1), (2,2), (3,4), (4,1), (4,4)\}$$

$$R_2 = \{(1,1), (1,2), (2,1)\}$$

$$R_3 = \{(1,1), (1,2), (1,4), (2,1), (2,2), (3,3), (4,1), (4,4)\}$$

$$R_4 = \{(2,1), (3,1), (3,2), (4,1), (4,1), (4,3)\}$$

$$R_5 = \{(1,1), (1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)\}$$

$$R_6 = \{(3,4)\}$$

Relación transitiva del ejemplo anterior:

$$R_4: (3, 2) \text{ y } (2, 1) \Rightarrow (3, 1); (4, 2) \text{ y } (2, 1) \Rightarrow (4, 1); (4, 3) \text{ y } (3, 1) \Rightarrow (4, 1)$$

Ejercicios:

1. Enumera los pares ordenados de la relación R de $A = \{0,1,2,3,4\}$ en $B = \{0,1,2,3\}$, donde $(a, b) \in R$ si, y solo si.

a) $a = b$ b) $a + b = 4$ c) $a > b$

d) a / b e) $\text{mcd}(a,b) = 1$ f) $\text{mcm}(a,b) = 2$

2. Para cada una de las siguientes relaciones en el conjunto $[1,2,3,4]$, decide si es o no: reflexiva, simétrica, antisimétrica o transitiva.

- a) $\{(2,2),(2,3),(2,4),(3,2),(3,3),(3,4)\}$
- b) $\{(1,1),(1,2),(2,1),(2,2),(3,3),(4,4)\}$
- c) $\{(2,4),(4,2)\}$
- d) $\{(1,2),(2,3),(3,4)\}$
- e) $\{(1,1),(2,2),(3,3),(4,4)\}$
- f) $\{(1,3),(1,4),(2,3),(2,4),(3,1),(3,4)\}$

Solución

1.

- a) $R = \{(0,0),(1,1),(2,2),(3,3)\}$
- b) $R = \{(1,3),(2,2),(3,1),(4,0)\}$
- c) $R = \{(1,0),(2,0),(2,1),(3,0),(3,1),(3,2),(4,0),(4,1),(4,2),(4,3)\}$
- d) $R = \{(0,1),(0,2),(0,3),(1,1),(2,1),(2,2),(3,1),(3,3),(4,1),(4,2)\}$
- e) $R = \{(1,1),(1,2),(1,3),(2,1),(3,1),(4,1)\}$
- f) $R = \{(1,2),(2,1),(2,2)\}$

2.

- a) $\{(2,2), (2,3), (2,4), (3,2), (3,3), (3,4)\}$ **Transitiva**
- b) $\{(1,1), (1,2), (2,1), (2,2), (3,3), (4,4)\}$ **Reflexiva, Simétrica y Transitiva**
- c) $\{(2,4), (4,2)\}$ **Simétrica**
- d) $\{(1,2), (2,3), (3,4)\}$ **Antisimétrica**
- e) $\{(1,1), (2,2), (3,3), (4,4)\}$ **Reflexiva y simétrica**
- f) $\{(1,3), (1,4), (2,3), (2,4), (3,1), (3,4)\}$ **No tiene ninguna de las propiedades**

COMBINACIÓN DE RELACIONES

Debido a que las relaciones A en B , son subconjuntos $A \times B$, dos relaciones A en B , se pueden combinar de cualquiera de las maneras en que se pueden combinar dos conjuntos, considerar los siguientes ejemplos:

Ejemplo:

Sean $A = \{1, 2, 3\}$ y $B = \{1, 2, 3, 4\}$. Las relaciones $R_1 = \{(1, 1), (2, 2), (3, 3)\}$ y $R_2 = \{(1, 1), (1, 2), (1, 3), (1, 4)\}$ se pueden combinar para obtener:

Solución:

$$R_1 \cup R_2 = \{(1, 1), (2, 2), (3, 3), (1, 2), (1, 3), (1, 4)\}.$$

$$R_1 \cap R_2 = \{(1, 1)\}$$

$$R_1 - R_2 = \{(2, 2), (3, 3)\}$$

$$R_2 - R_1 = \{(1, 2), (1, 3), (1, 4)\}$$

Ejemplo:

Sea R_1 la relación <menor que> en el conjunto de los números reales y sea R_2 la relación <mayor que> en el conjunto de los números reales, lo cual quedaría:

$$R_1 = \{(x, y) \mid x < y\}, \text{ y } R_2 = \{(x, y) \mid x > y\}.$$

Hallar: $R_1 \cup R_2$, $R_1 \cap R_2$, $R_1 - R_2$, $R_2 - R_1$ y $R_1 \oplus R_2$.

Solución:

Se observa que $(x, y) \in R_1 \cup R_2$, si, y solo si, que $(x, y) \in R_1$ o $(x, y) \in R_2$. Por lo tanto: $(x, y) \in R_1 \cup R_2$, $x < y$ o $x > y$. Como la condición anterior coincide con la condición $x \neq y$, lo cual sigue que $R_1 \cup R_2 = \{(x, y) \mid x \neq y\}$.

Para completar, es imposible un par (x, y) que pertenezca a R_1 y R_2 , lo ya que no se puede dar a la vez $x < y$ y $x > y$ ($R_1 \cap R_2$), misma lógica se aplica para las demás combinaciones.

Definición 6

Sean R una relación de un conjunto A en un conjunto B y S , una relación de B en un conjunto C . La composición de R y S es la relación que consiste en los pares ordenados (a, c) con $a \in A$ y $c \in C$, para los cuales existe un elemento $b \in B$ tal que $(a, b) \in R$ y $(b, c) \in S$. La composición de R y S , se denota $S \circ R$.

Ejemplo:

¿Cuál es la composición de las relaciones R y S , donde R es la relación de $\{1, 2, 3\}$ en $\{1, 2, 3, 4\}$ con $R = \{(1, 1), (1, 4), (2, 3), (3, 1), (3, 4)\}$ y S es la relación de $\{1, 2, 3, 4\}$ en $\{0, 1, 2\}$ con $S = \{(1, 0), (2, 0), (3, 1), (3, 2), (4, 1)\}$?

Solución: se va a construir usando los pares ordenados de R y los pares ordenados de S tales que el segundo elemento del par ordenado R coincida con el primer elemento del par ordenado de S , es decir, considerando a los pares ordenados como (a, b) se toma b del par ordenado de R , este anterior tiene que ser igual al par a de S para producir los pares ordenados de $S \circ R$. Calculando todos los pares ordenados en la composición $S \circ R$, tenemos:

$$S \circ R = \{(1, 0), (1, 1), (2, 1), (2, 2), (3, 0), (3, 1)\}.$$

Definición 7:

Sea R una relación en un conjunto A . Las potencias R^n , donde $n \in \mathbb{N} \mid n = 1, 2, 3, \dots$, se definen recursivamente como:

$$R^1 = R \text{ y } R^{n+1} = R^n \circ R$$

La definición muestra que $R^2 = R \circ R$; $R^3 = R^2 \circ R = (R \circ R) \circ R$.

Ejemplo:

Sea $R = \{(1, 1), (2, 1), (3, 2), (4, 3)\}$. Hállense las potencias R^n , $n = 2, 3, 4, \dots$

Solución: Si $R^2 = R \circ R$, entonces obtenemos:

$(1, 1), (2, 1), (3, 2), (4, 3)$

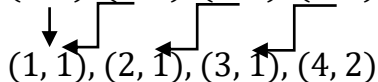


$(1, 1), (2, 1), (3, 2), (4, 3)$

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

Si $R^3 = R \circ R^2$, entonces obtenemos:

$(1, 1), (2, 1), (3, 2), (4, 3)$



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

TEOREMA 1

Definición: La relación R en un conjunto A es transitiva si, y sólo si, $R^n \subseteq R$ para $n = 1, 2, 3, \dots$

Demostración: Se demuestra la parte “sí” del teorema. Suponemos que $R^n \subseteq R$ para $n = 1, 2, 3, \dots$. En particular, $R^2 \subseteq R$. Para ver que R es transitiva, se ve que $(a, b) \in R$ y $(b, c) \in R$, entonces por la definición de composición se tiene $(a, c) \in R^2$. Como $R^2 \subseteq R$, esto significa que $(a, c) \in R$. Por tanto, R es transitiva.

Ahora se demuestra la parte “sólo si” por el teorema de inducción. Y se nota que cuando $n = 1$ esto se cumple.

Supongamos que $R^n \subseteq R$ para un entero positivo n . Ésta es hipótesis de inducción. Para completar el paso de inducción debemos demostrar que se implica R^{n+1} es también un subconjunto de R . Para hacerlo se supone que $(a, b) \in R^{n+1}$. Entonces, como $R^{n+1} = R^n \circ R$, existe un elemento x con $x \in A$ | $(a, x) \in R$ y $(x, b) \in R^n$. La hipótesis de inducción, esto es, el hecho de que $R^n \subseteq R$, implica que $(x, b) \in R$. Además, como R es transitiva con $(a, x) \in R$ y $(x, b) \in R$, se sigue que $(a, b) \in R$. Esto demuestra que $R^{n+1} \subseteq R$, demostrándolo.

Relaciones n-arias y sus aplicaciones:

Una relación n -aria es una relación que involucra a n conjuntos de elementos, donde n es un número natural mayor o igual a 2. En otras palabras, es una relación que conecta múltiples elementos de diferentes conjuntos al mismo tiempo. Por ejemplo, una relación ternaria conecta tres conjuntos de elementos, mientras que una relación cuaternaria conecta cuatro conjuntos de elementos, y así sucesivamente. En el ámbito de las bases de datos, las relaciones n -arias se utilizan para modelar las relaciones complejas entre múltiples entidades.

RELACIÓN N-ARIAS

Definición: Sean A_1, A_2, \dots, A_n conjuntos. Una relación n -aria en estos conjuntos es subconjunto de $A_1 \times A_2 \times \dots \times A_n$. Los conjuntos A_1, A_2, \dots, A_n se llaman dominios de la relación y n es el grado de la relación.

Ejemplo: Sea R la relación en $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ que consta de ternas (a, b, c) en las que a, b y c son enteros positivos con $a < b < c$. Entonces $(1, 2, 3) \in R$, pero $(2, 3, 4) \notin R$. El grado de esta relación es 3.

Sus dominios son todos iguales al conjunto de los números naturales.

BASES DE DATOS

Las bases de datos constan de registros, cuyos están formados por n -tuplas formados a partir de **campos**. Estos campos son las entradas de la n -tupla. Por ejemplo, una base de datos de registros de estudiantes puede estar formada por campos que contienen el nombre, el ID, la titulación en la que este matriculado y la nota promedio. El modelo relacional de datos representa una base de datos de registros como una relación n -aria. Así los registros se representan como 4-tuplas de la siguiente forma: (Nombre Estudiante, ID, Titulación, Nota Media). Un ejemplo sería.

(Juan Silv, 12934, Ing. Sistemas, 4.1)
(Alejandro Laz, 23441, Ing. Sistemas, 4.1)
(Juan Pad, 18239, Ing. Sistemas, 4.2)
(Sebastián Pez, 92341, Ing. Sistemas, 4.2)
(Tomas Mance, 85472, Ing. Sistemas, 4.1)

Tabla 1. Estudiantes			
Nombre Estudiantes	Numero ID	Titulación	Nota Promedio
Ackermann	231455	Informática	3.88
Adams	888323	Física	3.45
Chou	102147	Informática	3.49
Goodfriend	453876	Matemáticas	3.45
Rao	678543	Matemáticas	3.90
Stevens	786576	Psicología	2.99

Una manera diferente de representar las bases de datos seria con tablas, pues son la manera más eficaz e intuitiva de representarlas. Cada columna representa un atributo de la base de datos. En este caso los atributos de esta base de datos serian: Nombre estudiante, numero de ID (identificación), titulación, nota promedio.

Se dice que un dominio de una relación n -aria es una **clave primaria** si el valor de la n -tupla en dicho dominio determina la n -tupla. Esto es, un dominio en una clave primaria cuando no hay dos n -tuplas en la relación que tengan el mismo valor en ese dominio.

A menudo se añaden y se eliminan registros de la base de datos así que un dominio como clave primaria puede variar seguido. Como buena práctica, se debería elegir una clave primaria que continúe siendo la misma sin importar los cambios en la base de datos, algo relacionado con la intención de la base de datos

que pueda llegar a contener todas las n -tuplas y las relaciones n -arias que se puedan llegar a añadir en un futuro a la base

Ejemplo: ¿Que dominios son claves primarias para la relación n -aria que se muestra en la Tabla 1, suponiendo que no se va a añadir en el futuro ninguna n -tupla?

Solución: Como hay una sola 4-tupla en la tabla para cada nombre de estudiante, el dominio de nombres de estudiantes es una clave primaria. A su vez los IDs en la tabla son únicos, lo cual los hace también clave primaria. Ahora, tanto la titulación como la nota media no son claves únicas, debido a que más de una 4-tupla que contiene la misma información.

OPERACIONES CON RELACIONES n -ARIAS

La operación más sencilla sobre una relación n -aria es determinar todas las n -tuplas de la relación n -aria que satisfacen ciertas condiciones. Por ejemplo, puede que queramos encontrar en una base de datos de estudiantes los registros de todos los estudiantes matriculados en la titulación de informática. O podemos buscar en esa misma base de datos todos los registros de todos los estudiantes con una nota media por encima de 3,5 sobre 5. O podemos buscar los registros de todos los estudiantes matriculados en informática con una nota media mayor que 3,5. Para llevar a cabo estas tareas utilizamos el operador de selección.

Definición 2: Sea R una relación n -aria y C una condición que puede ser satisfecha por los elementos de R . Entonces, el operador de selección S_C transforma la relación n -aria R en la relación n -aria formada por todas las n -tuplas de R que satisfacen la condición C .

Ejemplo:

Tabla 1 (Estudiantes)			
Nombre_estudiante	Numero_ID	Titulación	Nota_media
Ackermann	231455	Informática	3,88
Adams	888323	Física	3,45
Chou	102147	Informática	3,49
Goodfriend	453876	Matemáticas	3,45
Rao	678543	Matemáticas	3,90
Stevens	786576	Psicología	2,99

Para hallar los registros de estudiantes de informática en la relación n-aria que se muestra en la tabla usamos el operador S_{C_1} , donde C_1 es la condición Titulación = <<Informática>>. El resultado son las dos 4-tuplas (Ackermann,231455, Informatica,3,88) y (Chou,102147, Informatica,3,49). De forma parecida, para determinar los registros de los estudiantes con una nota media superior a 3,5 en esta base de datos utilizamos el operador S_{C_2} , donde C_2 es la condición Nota Media > 3,5. El resultado son las dos 4-tuplas (Ackermann,231455, Informática, 3,88) y (Rao,678543, Matemáticas, 3,90). Finalmente, para hallar los registros de estudiantes de informática con una nota media superior a 3,5 usamos el operador S_{C_3} , donde C_3 es la condición (Titulación = <<Informática>> ^ Nota Media > 3,5). El resultado consta de una única 4-tupla (Ackermann,231455, Informática, 3,88).

Las proyecciones se utilizan para formar nuevas relaciones n-arias eliminando los mismos campos en cada registro de la relación.

Definición 3: La proyección p_{i_1, i_2, \dots, i_m} transforma la n-tupla (a_1, a_2, \dots, a_n) en la m-tupla $(a_{i_1}, a_{i_2}, \dots, a_{i_m})$, donde $m \leq n$.

En otras palabras, la proyección p_{i_1, i_2, \dots, i_m} , elimina n – m componentes de una n-tupla, manteniendo los componentes i_1 -ésima, i_2 -ésima, ... e i_m -ésima.

SQL

SQL siglas de Structured Query Language, en español Lenguaje de Consulta Estructurado, es lo que se usa para demostrar los descripto en estos ejemplos.

Ejemplo 10.

Vamos a hacer uso del Select la cual es una consulta de una o varias columnas de la tabla deseada.

Vuelos				
Id_Vuelo	Hora_Salida	Hora_Llegada	Destino_Salida	Destino_Llegada
154	10:00:00 a. m.	11:00:00 p. m.	Cali	Bogotá
541	1:00:00 p. m.	2:00:00 p. m.	Cali	Bogotá
651	7:00:00 a. m.	8:00:00 p. m.	Bogotá	Madrid
841	6:00:00 a. m.	7:00:00 a. m.	Cali	Medellín

Teniendo la tabla Vuelos seleccionaremos los vuelos que tengan como destino Bogotá, como solo queremos saber la hora de salida, seleccionamos el campo Hora_Salida.

Vuelos	Select
<pre>SELECT Hora_Salida AS Resultado FROM Vuelos WHERE (((Vuelos.[Destino_Llegada])='Bogota'));</pre>	

Ahora se muestra como se hizo la selección.

Vuelos	Select
	Resultado
	10:00:00 a. m.
	1:00:00 p. m.

REPRESENTACIÓN DE RELACIONES

Introducción

Existen muchas formas de representar una relación entre conjuntos finitos (una de ellas es enumerar sus pares ordenados). En la siguiente sección, se mostrarán 2 formas diferentes de represar relaciones; Con matrices booleanas, y el otro con grafos dirigidos.

Representación de relaciones usando matrices

Una relación entre conjuntos finitos se puede representar utilizando una matriz booleana. Supongamos que R es una relación $A = \{a_1, a_2, \dots, a_m\}$ en $B = \{b_1, b_2, \dots, b_m\}$ (escribimos los elementos de los conjuntos A y B en un orden particular, aunque arbitrario. Además, cuando $A = B$ usamos la misma ordenación para A y para B). La relación R puede representarse por medio de la Matriz $\mathbf{M}_R = [m_{ij}]$ donde:

$$m_{ij} = \begin{cases} 1 & \text{si } (a_i, b_j) \in R \\ 0 & \text{si } (a_i, b_j) \notin R \end{cases}$$

En otras palabras, la matriz booleana que representa a R tiene un 1 como elemento (i, j) si a_i está relacionado con b_j (esta representación depende de las ordenaciones utilizadas para A y B).

EJEMPLOS:

- 1) Supongamos que $A = \{1, 2, 3\}$ y $B = \{1, 2\}$. Sea R la relación de A en B que contiene a (a, b) si $a \in A$, $b \in B$ y $a > b$. ¿Cuál es la matriz que representa a R si $a_1 = 1$, $a_2 = 2$, $a_3 = 3$, $b_1 = 1$, $b_2 = 2$?

2)

Solución: Como $R = \{(2, 1), (3, 1), (3, 2)\}$, la matriz de R es:

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

Los M_R muestran que los pares $(2, 1)$, $(3, 1)$ y $(3, 2)$ pertenecen a R . Los ceros muestran que ningún otro par pertenece a R .

- 3) Sean $A = \{a_1, a_2, a_3\}$ y $B = \{b_1, b_2, b_3, b_4, b_5\}$. ¿Qué pares ordenados están en la relación R representada por la matriz?

$$M_R = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} ?$$

Solución: Como R está formada por los pares ordenados (a_i, b_j) con $m_{ij} = 1$, se sigue que:

$$R = \{(a_1, b_2), (a_2, b_1), (a_2, b_3), (a_2, b_4), (a_3, b_1), (a_3, b_3), (a_3, b_5)\}$$

La matriz de una relación en un conjunto, que es cuadrada, puede usarse para determinar para ver si cumple o no ciertas propiedades.

Matriz simétrica: La matriz simétrica es una matriz cuadrada en que los elementos en posición simétrica respecto a la diagonal principal son iguales.

También decimos que una matriz cuadrada es simétrica si es igual a su transpuesta; es decir: $A \Leftrightarrow A^t$

$$A_n + A_n^t = S_N$$

$$\begin{pmatrix} 2 & 8 & 3 & -1 \\ 1 & 2 & 3 & 7 \\ -2 & 3 & 1 & 2 \\ 3 & -4 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 2 & 1 & -2 & 3 \\ 8 & 2 & 3 & -4 \\ 3 & 3 & 1 & 0 \\ -1 & 7 & 2 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 2+2 & 8+1 & 3-2 & -1+3 \\ 1+8 & 2+2 & 3+3 & 7-4 \\ -2+3 & 3+3 & 1+1 & 2+0 \\ 3-1 & -4+7 & 0+2 & 1+1 \end{pmatrix} = \begin{pmatrix} 4 & 9 & 1 & 2 \\ 9 & 4 & 6 & 3 \\ 1 & 6 & 2 & 2 \\ 2 & 3 & 2 & 2 \end{pmatrix}$$

Imagen tomada de:

<https://www.universoformulas.com/matematicas/algebra/matriz-simetrica/>

Matriz reflexiva:

La relación descrita en una matriz es reflexiva si la diagonal principal de la matriz contiene sólo 1's (unos). Por lo tanto, lo que se debe de recorrer es la diagonal principal.

Ejemplo:

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

Matriz antisimétrica: Decimos que una matriz cuadrada es antisimétrica si es igual a la opuesta de su transpuesta; es decir:

$$A \text{ es antisimétrica} \Leftrightarrow A^t = -A \Leftrightarrow A = -A^t.$$

Ejemplo:

$$A = \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -4 \\ 1 & 4 & 0 \end{bmatrix} \Rightarrow A^t = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 4 \\ -1 & -4 & 0 \end{bmatrix} \Rightarrow -A^t = \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -4 \\ 1 & 4 & 0 \end{bmatrix}$$

Observación: Se observa que $A = -A^t$, por lo tanto, es antisimétrica.

Matriz transitiva: Una matriz transitiva es una matriz cuadrada que representa una relación transitiva entre los elementos de un conjunto. En una matriz transitiva, los elementos de la diagonal principal son siempre ceros, ya que un elemento no puede estar relacionado consigo mismo.

Ejemplo: Supongamos que tenemos un conjunto de cinco elementos: $A = \{a, b, c, d, e\}$. Y que existe una relación "mayor o igual que" entre estos elementos. Es decir, si decimos que "a" es mayor o igual que "b", y "b" es mayor o igual que "c", entonces podemos concluir que "a" es mayor o igual que "c". En este caso, la relación es transitiva.

La matriz transitiva que representa esta relación es la siguiente:

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

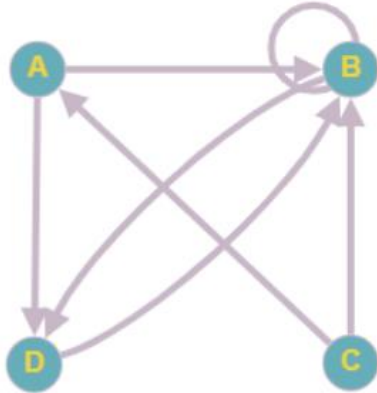
REPRESENTACIÓN DE RELACIONES USANDO GRAFOS

Grafos dirigidos: Otra manera de representar relaciones son los grafos dirigidos o dígrafos. Los elementos del conjunto se representan mediante un punto y los pares ordenados mediante un segmento orientado cuyo sentido viene indicado por una flecha.

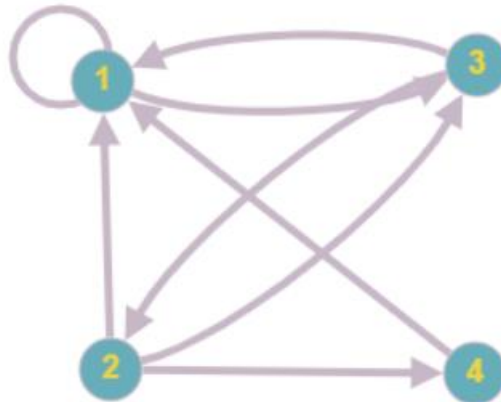
Definición:

Un grafo dirigido, o dígrafo, consta de un conjunto de V vértices (o nodos) junto con un conjunto E de pares ordenados de elementos de V llamados aristas (o arcos). Al vértice a se le llama *vértice inicial* de la arista (a, b) , y al vértice b se le llama *vértice final* de la arista.

Ejemplo: El grafo dirigido con vértices a, b, c y d y aristas:
 (a, b) , (a, d) , (b, b) , (b, d) , (c, a) , (c, b) y (d, b) .



Otro ejemplo: El grafo dirigido de la relación:
 $R = \{(1, 1), (1, 2), (2, 1), (2, 3), (2, 4), (3, 1), (3, 2), (4, 1)\}$



Observación: nótese que una relación simétrica se puede representar por medio de un grafo no dirigido, que es un grafo en el que las aristas no se les asigna un sentido.

CIERRE DE RELACIONES

Introducción

En general, sea R una relación en un conjunto A . R puede o no cumplir una cierta propiedad P , como la reflexividad, la simetría o la transitividad. Sea S una relación que cumple la propiedad P , que contiene a R y tal que S es subconjunto de cualquier otra relación con la propiedad P que contenga a R . Entonces se dice que S es el cierre o la clausura de R con respecto a P (nótese que el cierre de una relación con respecto a una propiedad puede no existir).

Cierres

La relación $R = \{(1,1), (1, 2), (2, 1), (3, 2)\}$ en el conjunto $A = \{1, 2, 3\}$ no es reflexiva. ¿Cómo podemos producir una relación que contenga a R y sea lo más pequeña posible? Esto se puede hacer añadiendo $(2, 2)$ y $(3,3)$ a R , ya que éstos son los únicos pares de la forma (a, a) que no están en R . Claramente, esta nueva relación contiene a R . Además, cualquier relación reflexiva que contenga a R tiene que contener también a $(2, 2)$ y a $(3, 3)$. Como esta relación contiene a R , es reflexiva y está contenida en cualquier relación reflexiva que contenga a R . Se le llama cierre reflexivo (o clausura reflexiva) de R .

Ejemplo 1: ¿Cuál es el cierre reflexivo de la relación $R = \{(a, b) \mid a < b\}$ en el conjunto de los números enteros?

Solución: El cierre reflexivo de R es:

$$R \cup \mu = \{(a, b) \mid a < b\} \cup \{(a, a) \mid a \in \mathbb{Z}\} = \{(a, b) \mid a \leq b\}$$

Cierre simétrico: La relación $((1, 1), (1,2), (2,2), (2,3), (3, 1), (3, 2))$ en $\{1, 2, 3\}$ no es simétrica. ¿Cómo podemos producir una relación simétrica tan pequeña como sea posible y que contenga a R ? Para hacerlo sólo necesitamos añadir $(2, 1)$ y $(1,3)$. ya que éstos son los únicos pares de la forma (b, a) .

Con $(a, b) \in R$ que no están en R . Esta nueva relación es simétrica y contiene a R . Además, cualquier relación simétrica que contenga a R tiene que contener también a esta nueva relación, ya que una relación simétrica que contenga a R debe contener a $(2,1)$ y a $(1,3)$. En consecuencia, a esta nueva relación se le llama cierre simétrico (o clausura simétrica) de R .

Tal como se ilustra en el ejemplo el cierre simétrico de R se puede construir añadiendo aquellos pares ordenados de la forma (b, a) que no estén en R y tales que $(a, b) \in R$.

Cierre transitivo: Supongamos que una relación no es transitiva. ¿Cómo podemos producir una relación transitiva que contenga a R y que esté contenida en cualquier relación transitiva que contenga a R ? ¿Puede construirse el cierre transitivo de una relación R añadiendo todos los pares de la forma (a, c) tales que (a, b) y (b, c) ya están en R ? Consideremos la relación $R = \{(1, 3), (1, 4), (2, 1), (3, 2)\}$ en el conjunto $\{1, 2, 3, 4\}$. Esta relación no es transitiva porque no contiene a todos los pares de la forma (a, c) tales que (a, b) y (b, c) están en R . Los pares de esta forma que no están en R son $(1, 2)$, $(2, 3)$, $(2, 4)$ y $(3, 1)$. Añadir estos pares no produce una relación transitiva, porque la relación resultante contiene a $(3, 1)$ y a $(1, 4)$, pero no contiene a $(3, 4)$. Esto demuestra que construir el cierre transitivo de una relación es más complicado que construir el cierre reflexivo o el cierre simétrico.

Por ende, para este caso se usan los caminos de grafos dirigidos.

Uso de Grafos dirigidos

Un camino en un grafo dirigido puede pasar más de una vez por un mismo vértice. Además, una arista del grafo dirigido puede aparecer más de una vez en el camino.

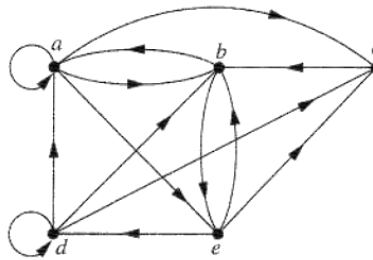


Figura 1. Un grafo dirigido.

¿Cuáles de entre los siguientes son caminos en el grafo dirigido que se muestra en la Figura 1: a, b, e, d ; a, e, c, d, b ; b, a, c, b, a, a, b ; d, c ; c, b, a ; e, b, a, b, a, b, e ? ¿Qué longitud tienen aquellos que son caminos? ¿Cuáles de entre los caminos de esta lista son circuitos?

Solución: Puesto que tanto (a, b) como (b, e) y (e, d) son aristas, a, b, e, d es un camino de longitud tres. Como (c, d) no es arista, a, e, c, d, b no es un camino. La sucesión b, a, c, b, a, a, b es un camino de longitud seis, ya que (b, a) , (a, c) , (c, b) , (b, a) , (a, a) y (a, b) son todas aristas. Vemos que d, c es un camino de longitud uno, ya que (d, c) es una arista. También c, b, a es un camino de longitud dos, ya que (c, b) y (b, a) son aristas. Todos los pares (e, b) , (b, a) , (a, b) , (b, a) , (a, b) y (b, e) son aristas, de modo que e, b, a, b, a, b, e es un camino de longitud seis.

Los dos caminos b, a, c, b, a, a, b y e, b, a, b, a, b, e son circuitos, ya que empiezan y terminan en el mismo vértice. Ninguno de los caminos a, b, e, d ; c, b, a , y d, c es un circuito.

EL ALGORITMO DE WARSHALL

Introducción

El algoritmo de Warshall, llamado así en honor de Stephen Warshall, quien describió el algoritmo

en 1960, es un método eficiente para calcular el cierre transitivo de una relación. El Algoritmo 1 puede hallar el cierre transitivo de una relación en un conjunto de n elementos usando $2n^3$ ($n - 1$) operaciones con bits. Veremos que el cierre transitivo se puede determinar mediante el algoritmo de Warshall haciendo sólo $2n^3$ operaciones con bits.

El algoritmo de Warshall se basa en la construcción de una sucesión de matrices booleanas. Estas matrices son w_0, w_1, \dots, w_n siendo $w_0 = M_R$.

Ejemplo: Sea R la relación cuyo grafo dirigido se muestra en la figura 3. Sea a, b, c, d un listado de los elementos del conjunto. Calcula las matrices w_0, w_1, w_2, w_3, w_4 . La matriz w_4 es el cierre transitivo de R .

Solución: Sean $v_1 = a, v_2 = b, v_3 = c$ y $v_4 = d$. w_0 es la matriz de la relación. Por tanto,

$$w_0 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$w_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$w_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$w_3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$w_4 = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

w_4 es el cierre transitivo.

RELACIONES DE EQUIVALENCIA

Una relación R en un conjunto A es una relación de equivalencia si es reflexiva, simétrica y transitiva.

Si dos elementos están relacionados por una relación de equivalencia, se dice que son **equivalentes** (lo cual tiene sentido ya que una relación de equivalencia es simétrica). Como una relación de equivalencia es reflexiva, en una relación de equivalencia cualquier elemento es equivalente a sí mismo. Además, como toda relación de equivalencia es transitiva si, a y b son equivalentes y b y c son equivalentes, se sigue que a y c son equivalentes.

Ejemplo: Sea R la relación en el conjunto de los números reales tal que $a R b$ sí, y solo sí, $a - b$ es un número entero. ¿Es R una relación de equivalencia?

Solución: Como $a - a = 0$ es entero para cualquier número real a , $a R a$ para todos los números reales a . Por tanto, R es reflexiva. Ahora supongamos que $a R b$. Entonces, $a - b$ es un número entero, por lo que $b - a$ es también un entero, Por tanto, $b R a$, Se sigue que R es simétrica. Si $a R b$ y $b R c$, entonces $a - b$ y $b - c$ son enteros, por lo que $a - c = (a - b) + (b - c)$ es también un entero. Por tanto, $a R c$ y R es transitiva. En consecuencia, R es una relación de equivalencia.

Una de las relaciones de equivalencia que se emplea con más frecuencia es la congruencia módulo m , donde m es un entero positivo mayor que 1.

CLASES DE EQUIVALENCIA

Una clase de equivalencia es un conjunto de elementos que están relacionados entre sí de acuerdo con una relación de equivalencia. Una relación de equivalencia es una relación binaria que satisface tres propiedades: reflexividad, simetría y transitividad.

Dado un conjunto A y una relación de equivalencia R en A , una clase de equivalencia $[a]$ es el conjunto de todos los elementos de A que están relacionados con el elemento a por la relación R . Es decir, $[a] = \{x \in A \mid x R a\}$.

Para ilustrar esto, consideremos el conjunto $A = \{0, 1, 2, 3, 4, 5\}$ y la relación de equivalencia R definida como sigue: para cualquier par de elementos (a, b) en A , $a R b$ si y solo si $a \text{MOD}(3)$ y $b \text{MOD}(3)$ tienen el mismo residuo cuando se dividen por 3. Entonces, por ejemplo, $1 R 4$:

$1 \text{MOD}(3)=1$ y $4 \text{MOD}(3)=1$, porque su residuo es igual.

Este sería R para todo $A: R =$

$\{(0,0), (0,3), (1,1), (1,4), (2,2), (2,5), (3,0), (3,3), (4,1), (4,4), (5,2), (5,5)\}$

Las clases de equivalencia para el anterior caso son: $[0] = \{0, 3\}$, $[1] = \{1, 4\}$ y $[2] = \{2, 5\}$. La clase de equivalencia $[0]$ contiene todos los elementos que tienen residuo 0 al dividirse por 3, la clase de equivalencia $[1]$ contiene todos los elementos que tienen residuo 1 al dividirse por 3, y la clase de equivalencia $[2]$ contiene todos los elementos que tienen resto 2 al dividirse por 3.

Es importante destacar que todas las clases de equivalencia son disjuntas, lo que significa que no tienen elementos en común. Además, la unión de todas las clases de equivalencia de un conjunto A es igual a A , y cada elemento de A pertenece a exactamente una clase de equivalencia.

Particiones:

Una partición de un conjunto es una forma de dividir ese conjunto en subconjuntos disjuntos no vacíos, de modo que cada elemento del conjunto pertenezca exactamente a uno de los subconjuntos.

Formalmente, si A es un conjunto, una partición P de A es una colección de subconjuntos no vacíos de A , denotados como P_1, P_2, \dots, P_k , tales que:

- Cada P_i es un subconjunto no vacío de A .
- Los P_i son disjuntos dos a dos, es decir, si $i \neq j$, entonces $P_i \cap P_j = \emptyset$.
- La unión de todos los P_i es igual a A , es decir, $A = P_1 \cup P_2 \cup \dots \cup P_k$.

Una partición de un conjunto A puede ser vista como una forma de agrupar los elementos de A en categorías o clases, de manera que cada elemento pertenezca exactamente a una de ellas y no haya elementos que pertenezcan a más de una. Por lo tanto, las particiones son una herramienta útil en muchas áreas de las matemáticas, incluyendo la teoría de conjuntos, la teoría de números y la combinatoria.

Tomando el ejemplo anterior, podemos crear las particiones a partir del resultado de R ya que son diferentes. Para las clases de equivalencias: $[0] = \{0, 3\}$, $[1] = \{1, 4\}$ y $[2] = \{2, 5\}$ se le enumera para cumplir la unión de todos los P_i :

Conjunto $A = \{0,1,2,3,4,5\}$ del cual salen las particiones $A_1 = \{0,3\}$, $A_2 = \{1,4\}$ y $A_3 = \{2,5\}$ y no existe ningún $P_i = \emptyset$, los P_i son disjuntos dos a dos $A_1 \cap A_2 = \emptyset$, $A_1 \cap A_3 = \emptyset$ y $A_2 \cap A_3 = \emptyset$, y la unión de todos los P_i es igual a A , $A_1 \cup A_2 \cup A_3 = A$.

ORDENES PARCIALES

Definición 1: Se dice que una relación R en un conjunto S es una ordenación parcial, o que es un orden parcial, si es reflexiva, antisimétrica y transitiva. Se llama conjunto parcialmente ordenado a cualquier conjunto S con un orden parcial R , y se denota por (S, R) .

Ejemplo: Demuestra que la relación “Mayor o igual que” (\geq) es un orden parcial en el conjunto de los enteros.

Solución: como $a \geq a$ para cada entero a es reflexiva. Si $a \geq b$ y $b \geq a$, entonces $a = b$. Por lo tanto, \geq es antisimétrica. Finalmente, \geq es transitiva, ya que $a \geq b$ y $b \geq c$, implica que $a \geq c$.

Se sigue que \geq es un orden parcial en el conjunto de los enteros (\mathbb{Z}^+, \geq) es un conjunto parcialmente ordenado.

Definición 2: Se dice que los elementos a y b de un conjunto parcialmente ordenado (S, \leq) son comparables si se tiene que $a \leq b$ o bien $b \leq a$. Cuando a y b son elementos de S tales que ni $a \leq b$ ni $b \leq a$ se dice que a y b son no compatibles.

Ejemplo: En el conjunto parcialmente ordenado $(\mathbb{Z}^+, |)$ ¿son comparables los números enteros 3 y 9? ¿Son comparables 5 y 7?

Solución: Los enteros 3 y 9 son compatibles, ya que $3 \mid 9$. Los enteros 5 y 7 son no comparables, ya que $5 \nmid 7$ y $7 \nmid 5$.

Definición 3: Si (S, \leq) es un conjunto parcialmente ordenado y cada dos elementos S son comparables, se dice que S es un *conjunto totalmente ordenado o linealmente ordenado*. En tal caso, se dice que \leq es un *orden total u orden lineal*. A un conjunto totalmente ordenado también se le llama *cadena*.

Ejemplo: El conjunto parcialmente ordenado $(\mathbb{Z}, >)$ está totalmente ordenado, ya que $a \leq b$ o $b \geq a$ para todo par de números enteros a y b .

Definición 4: (S, \leq) es un *conjunto bien ordenado* si \leq es un orden total, y cualquier subconjunto no vacío de S tiene un elemento mínimo.

Ejemplo: En el conjunto de pares ordenados $(\mathbb{Z}^+ \times \mathbb{Z}^+)$, con $(a_1, a_2) \leq (b_1, b_2)$ si $a_1 < b_1$ o si $a_1 = b_1$ y $a_2 \leq b_2$ (el orden lexicográfico), es un conjunto bien ordenado. El conjunto \mathbb{Z} , con el orden usual \leq , no está bien ordenado, ya que el conjunto de los elementos negativos, que es un subconjunto de \mathbb{Z} , no tiene elemento mínimo.

ORDEN LEXICOGRAFICO

Las palabras de un diccionario aparecen en orden alfabético, o lexicográfico. Este orden se basa en el orden de las letras en el alfabeto y no es más que un caso particular de una ordenación de las cadenas de un conjunto construida a partir de un orden parcial definido sobre el conjunto. Veremos cómo funciona esta construcción en cualquier conjunto parcialmente ordenado.

Primero veremos cómo construir un orden parcial sobre el producto cartesiano de dos conjuntos parcialmente ordenados, (A_1, \leq_1) y (A_2, \leq_2) . **El orden lexicográfico** \leq en $A_1 \times A_2$ se define especificando que un par es menor que un segundo par si la primera componente del primer par es menor, (en A_1) que la primera componente del segundo par o si las primeras componentes son iguales, pero la segunda componente del primer par (en A_2) que la segunda componente par. En otras palabras, (a_1, a_2) es menor que (b_1, b_2) , esto es:

$$(a_1, a_2) < (b_1, b_2)$$

Si bien $a_1 <_1 b_1$ o bien $a_1 = b_1$ y $a_2 <_2 b_2$

Obtenemos un orden parcial \leq añadiendo la igualdad al orden $<$ en $A_1 \times A_2$.

Ejemplo: Determinar si es cierto o no $(3, 4) < (4, 8)$, que $(3, 8)$, $(4, 5)$ y que $(4, 9) < (4, 11)$ en el conjunto parcialmente ordenado $(\mathbf{Z} \times \mathbf{Z}, \leq)$, donde \leq es el orden lexicográfico construido a partir de la relación usual \leq en \mathbf{Z} .

Solución: Como $3 < 4$, se sigue que $(3, 5) < (4, 8)$ y que $(3, 8) < (4, 5)$. Se tiene $(4, 9) < (4, 11)$, ya que las primeras componentes de $(4, 9)$ y $(4, 11)$ son iguales, pero $9 < 11$.

Pueden definirse un orden lexicográfico en el producto cartesiano de n conjuntos parcialmente ordenados $(A_1, \leq_1), (A_2, \leq_2), \dots, (A_n, \leq_n)$. Se define el orden parcial \leq en $A_1 \times A_2 \times \dots \times A_n$ como $(a_1, a_2, \dots, a_n) < (b_1, b_2, \dots, b_n)$.

Ejemplo: Nótese que $(1, 2, 3, 5) < (1, 2, 4, 3)$, ya que los elementos que están en las dos primeras posiciones de las dos 4-tuplas coinciden, pero el elemento que está en la tercera posición de la primera 4-tupla, 3, es menor que el de la segunda 4-tupla, 4 (en este caso, el orden de las 4-tuplas es el orden lexicográfico que proviene de la relación usual “menor o igual que” en el conjunto de los enteros).

Ejemplo: Se considera el conjunto de letras minúsculas del alfabeto. Podemos construir un orden lexicográfico en el conjunto de cadenas de letras usando el orden de letras en el alfabeto. Una cadena es menor que una segunda cadena si la letra de la primera cadena que está en la primera posición en que difieren ambas cadenas es anterior en el alfabeto a la letra de la segunda cadena que está en esa misma posición o si la primera y la segunda cadena coinciden en todas las posiciones, pero la segunda contiene más letras. Ejemplo:

Discreta < Discreto

Discreta < Discretamente

Discreta < Discretizar

Discreta < Discreti

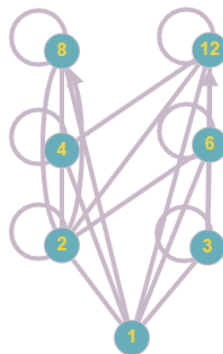
Diagramas de Hasse

Los diagramas de Hasse son representaciones visuales utilizadas en matemáticas discretas para ilustrar las relaciones de orden entre elementos de un conjunto finito. En un diagrama de Hasse, cada elemento se representa como un punto, y se dibuja una línea ascendente desde un elemento hacia otro si el primero es menor que el segundo en la relación de orden. De esta manera, se pueden visualizar fácilmente las propiedades del orden, como la transitividad y la existencia de elementos máximos y mínimos. Los diagramas de Hasse son útiles en áreas como la teoría de conjuntos, la teoría de grafos y la teoría de números.

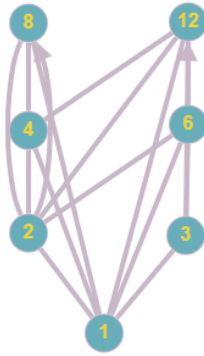
Ejemplo 11

Dibuja el diagrama de Hasse que representa al orden parcial $\{(a, b) \mid a \text{ divide } b\}$ en $\{1, 2, 3, 4, 6, 8, 12\}$

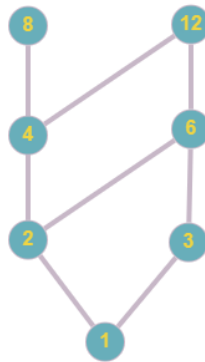
Se comienza con el grafo dirigido de este orden parcial.



Se eliminan todos los bucles.

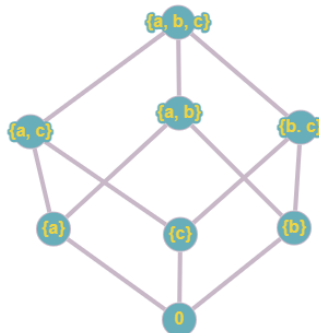


Después se eliminan todas las aristas que aparecen debido a la prioridad transitiva.



Ejemplo 12

Dibuja el diagrama de Hasse para el orden parcial $\{(A, B) \mid A \subseteq B\}$ en el conjunto de partes $P(S)$ siendo $S = \{a, b, c\}$.



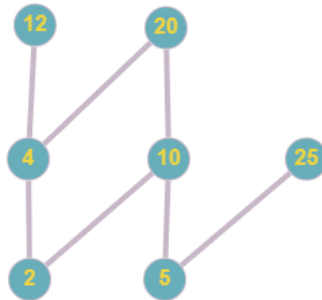
El diagrama de Hasse para este orden parcial se obtiene a partir del grafo dirigido asociado eliminando todos los bucles y todas las aristas que aparecen debido a la transitividad.

Elementos Maximales y Minimales

En matemáticas discretas, los términos "maximal" y "minimal" se utilizan para describir elementos de un conjunto que tienen ciertas propiedades de orden. Un elemento es "maximal" si no existe ningún otro elemento en el conjunto que sea mayor que él en la relación de orden. De manera similar, un elemento es "minimal" si no existe ningún otro elemento en el conjunto que sea menor que él en la relación de orden. Es importante tener en cuenta que los términos "máximo" y "mínimo" se refieren a elementos únicos que son los mayores o menores en todo el conjunto, mientras que "maximal" y "minimal" se refieren a elementos que son máximos o mínimos solo en su vecindad inmediata dentro del conjunto. En matemáticas discretas, los conceptos de maximal y minimal son importantes en áreas como la teoría de grafos, la teoría de conjuntos y la teoría de números.

Ejemplo 13

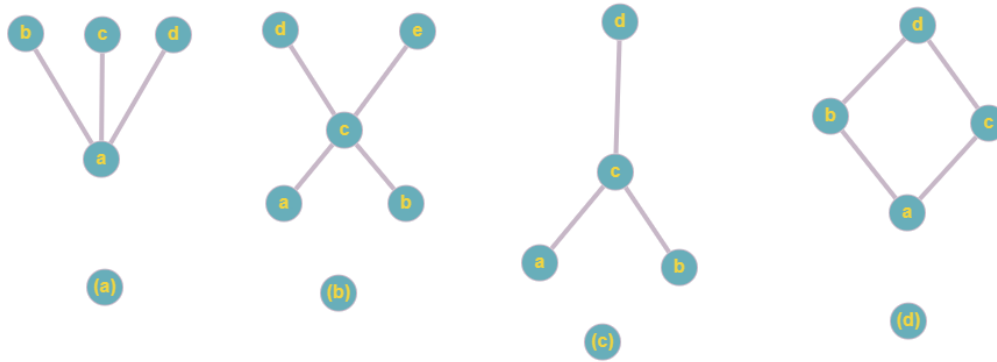
¿Qué elementos del conjunto parcialmente ordenado $\{2, 4, 5, 10, 12, 20, 25\}$ son maximales y cuales son minimales?



En el grafico se aprecia que los elementos maximales son 12, 20, 25 y los elementos minimales son 2, 5. Y como se puede apreciar, puede haber más de un elemento maximal y minimal.

Ejemplo 14

Determine cuáles de los conjuntos parcialmente ordenados representados en la siguiente figura por sus diagramas de Hasse tienen máximo y mínimo.



El mínimo del conjunto (a) es a, este conjunto no tiene máximo. El conjunto (b) no tiene máximo ni mínimo. El máximo del conjunto (c) es d, este conjunto no tiene mínimo. El máximo del conjunto (d) es d y el mínimo es a.

Ejemplo 15

Sea S un conjunto. Determine si hay o no un máximo y un mínimo en el conjunto parcialmente ordenado $(P(S), \subseteq)$.

El mínimo es el conjunto vacío, ya que $\emptyset \subseteq T$ para cualquier subconjunto T de S . El conjunto S es el máximo de este conjunto, ya que $T \subseteq S$ si T es subconjunto de S .

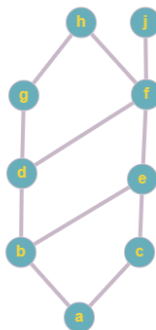
Ejemplo 16

¿Hay un máximo y un mínimo en el conjunto parcialmente ordenado $(\mathbb{Z}^+, |)$?

El número entero 1 es el mínimo, ya que $1 \mid n$ para todo entero positivo n . Como no hay ningún entero que sea divisible por todos los enteros positivos, no hay máximo.

Ejemplo 17

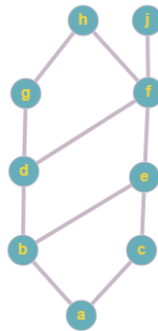
Determine las cotas inferiores y superiores de los subconjuntos $\{a, b, c\}$, $\{j, h\}$ y $\{a, c, d, f\}$ del conjunto parcialmente ordenado cuyo diagrama de Hasse es el siguiente.



Las cotas superiores de $\{a, b, c\}$ son e, f, j y h y su única cota inferior es a . No hay cotas superiores de $\{j, h\}$ y sus cotas inferiores son a, b, c, d, e y f . Las cotas superiores de $\{a, c, d, f\}$ son f, h y j y su cota inferior es a .

Ejemplo 18

Halla el ínfimo y el supremo, si es que existen, de $\{b, d, g\}$ en el conjunto parcialmente ordenado que se muestra en la siguiente figura.



Las cotas superiores de $\{b, d, g\}$ son g y h . Como $g < h$, g es el supremo. Las cotas inferiores de $\{b, d, g\}$ son a y b . Como $a < b$, b es el ínfimo.

Ejemplo 19

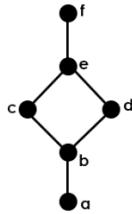
Halla el ínfimo y el supremo, si es que existen, de los conjuntos $\{3, 9, 12\}$ y $\{1, 2, 4, 5, 10\}$ en el conjunto parcialmente ordenado $(\mathbb{Z}^+, |)$.

Un número entero es cota inferior de $\{3, 9, 12\}$ si $3, 9$ y 12 son divisibles por ese entero. Los únicos enteros que cumplan esto son el 1 y el 3 . Como $1 \mid 3$, 3 es el ínfimo de $\{3, 9, 12\}$. La única cota inferior del conjunto $\{1, 2, 4, 5, 10\}$ con respecto a $|$ es el elemento 1 . Por tanto, 1 es el ínfimo de $\{1, 2, 4, 5, 10\}$.

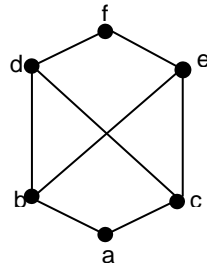
RETÍCULOS

A un conjunto parcialmente ordenado en el que cada par de elementos tiene tanto un supremo como un ínfimo se le llama **retículo**. Además, desempeñan un importante papel en las álgebras de Boole.

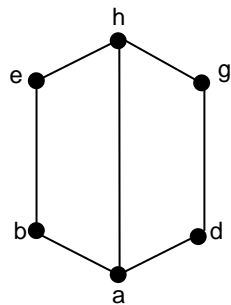
Ejemplo: Determinar si los siguientes conjuntos parcialmente ordenados representados por cada uno de los diagramas de Hasse son o no retículos.



Es un retículo debido a que tiene un supremo (e) y un ínfimo (b).



No es un retículo debido a que no tiene supremo.



Es un retículo debido a que tiene un supremo y un ínfimo.

Ejemplo: Determinar si los conjuntos parcialmente ordenados $(\{1, 2, 3, 4, 5\}, |)$, y $(\{1, 2, 4, 8, 16\}, |)$, so o no retículos.

Solución: Como 2 y 3 no tienen cotas superiores en $(\{1, 2, 3, 4, 5\}, |)$, ciertamente no tienen supremo, Por lo tanto, el primer conjunto no es un retículo.

ORDENACIÓN TOPOLÓGICA

Definición: Se dice que el orden total \leq es **compatible** con el orden parcial R si $a \leq b$ siempre que $a R b$. Al hecho que construir un orden total compatible a partir de un orden parcial se le llama **ordenación topológica**. Por lo cual, se necesitará el Lema 1.

Lema 1: Todo conjunto parcialmente ordenado no vacío y finito, (S, \leq) tiene un elemento minimal.

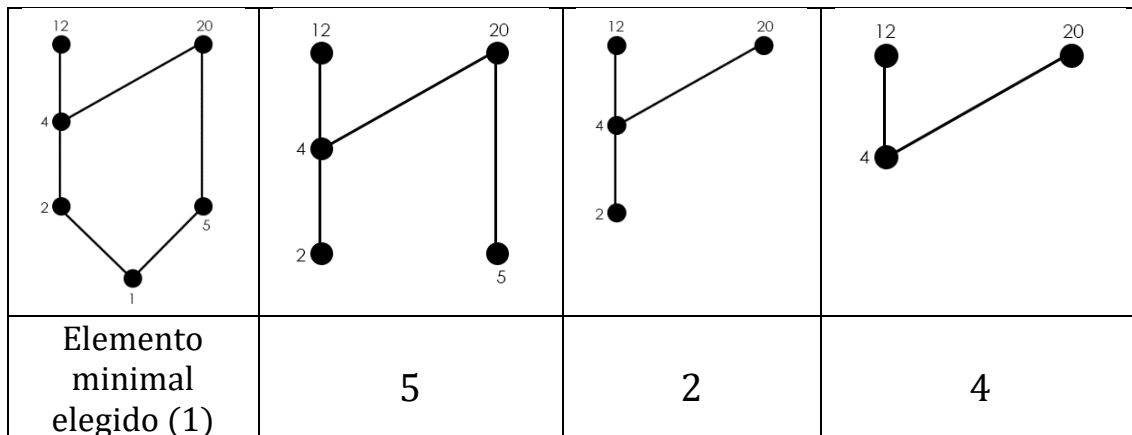
Demostración: Elegimos un elemento a_0 de S . si a_0 no es minimal, entonces hay un elemento a_1 con $a_1 < a_0$. Si a_1 no es minimal, hay un elemento a_2 con $a_2 < a_1$. Se continúa con este proceso, de modo que si a_n no es minimal, hay un elemento a_{n+1} con $a_{n+1} < a_n$. Como hay sólo un número finito de elementos en el conjunto, este proceso debe concluir produciendo un elemento minimal a_n .

Ejemplo: Hallar un orden total compatible para el conjunto parcialmente ordenado $(\{1, 2, 3, 4, 5, 12, 20\}, |)$.

Solución: En primer paso es elegir un elemento minimal. Éste tiene que ser 1, ya que es el único elemento minimal. A continuación, se selecciona un elemento minimal de $(\{2, 3, 4, 5, 12, 20\}, |)$. Hay dos elementos minimales en este conjunto, a saber, 2 y 5. Se elige el 5. Los elementos que quedan son $\{2, 4, 12, 20\}$. El único elemento minimal en esta etapa es el 2. A continuación, se elige el 4, ya que es el único elemento minimal de $(\{4, 12, 20\}, |)$. Dado que tanto 12 como 20 son elementos minimales de $(\{12, 20\}, |)$, cualquiera d ellos dos puede elegirse en esta etapa. Se elige el 20m lo que deja a 12 como único elemento restante. Esto produce el orden total: $1 < 5 < 2 < 4 < 20 < 12$

En la siguiente figura se muestran los pasos que ha seguido este algoritmo de ordenación.

Observación: La ordenación topológica puede aplicarse a la planificación de proyectos.



<div>12</div> <div>●</div>	<div>20</div> <div>●</div>	<div>12</div> <div>●</div>
20		12

UNIDAD 3

¿QUÉ SON LOS GRAFOS?

Un grafo es un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas (edges en inglés) que pueden ser dirigidos o no.

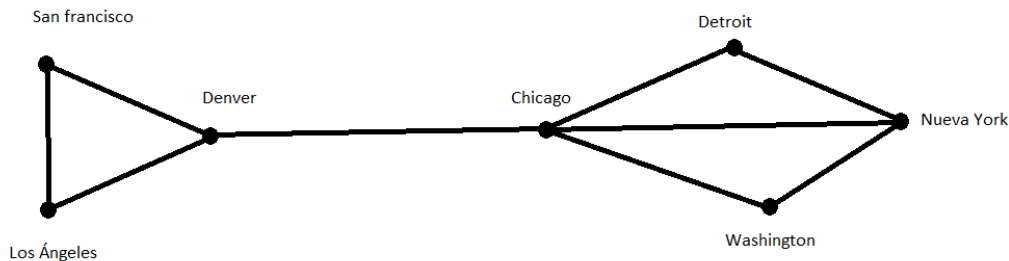
Tipos de grafos:

Grafo simple:

Véase el ejemplo:

Representar un ordenador mediante un punto y cada línea telefónica mediante un segmento.

Véase la siguiente figura:

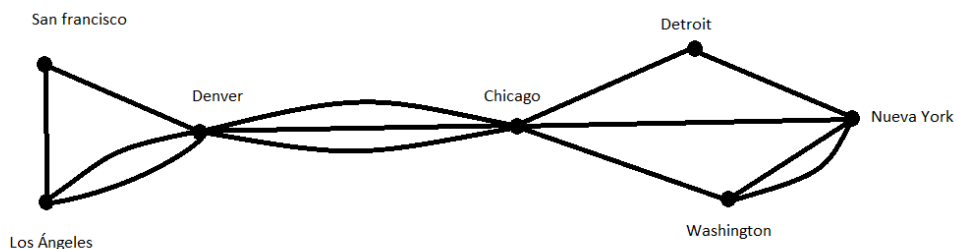


Este es llamado un **grafo simple** el cual consta de vértices que representan a los ordenadores y de aristas no dirigidas, que representan líneas telefónicas

Definición: Un grafo simple $G = (V, E)$ consta de V , un conjunto no vacío de vértices, y de E , un conjunto de pares no ordenados de elementos distintos de V . A estos pares se les llama aristas.

Multígrafo:

Véase la siguiente imagen siguiendo el mismo concepto anterior:

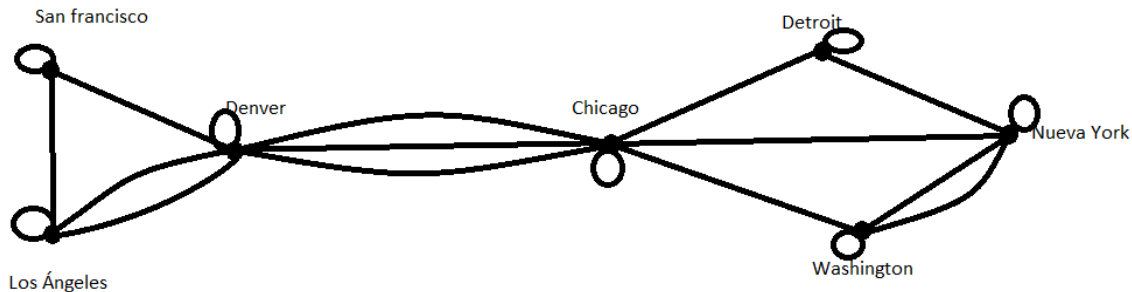


Cuando hay “mucho tráfico de información” no basta con un grafo simple, así que optamos por un **Multígrafo** que consta de aristas no dirigidas entre esos vértices, pero admitiendo la existencia de aristas múltiples entre pares de vértices.

Cabe aclarar que “*Todo grafo simple es un multígrafo. Sin embargo, no todos los multígrafos son grafos simples*”.

Definición: un multígrafo $G = (V, E)$ consta de un conjunto V de vértices, un conjunto E de aristas y una función $f(e)$ en $\{\{u, v\} \mid u, v \in V, u \neq v\}$. Se dice que las aristas e_1 y e_2 son aristas múltiples o paralelas si $f(e_1) = f(e_2)$.

Pseudografos: En la red informática puede haber una línea telefónica que conecte a un ordenador consigo mismo. Por lo tanto, no podemos usar multígrafos pues estos no aceptan el concepto de **bucles**. En este caso una arista puede conectar un vértice consigo mismo.

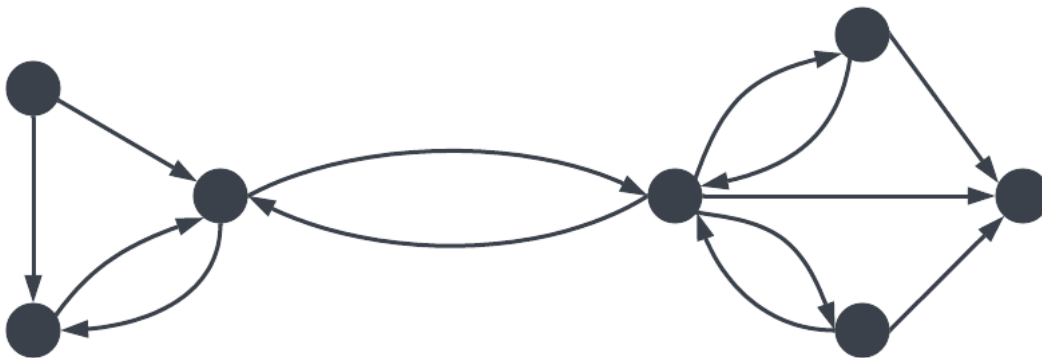


Definición: un pseudografo $G = (V, E)$ consta de un conjunto V de vértices, un conjunto E de aristas y una función $f(e)$ en $\{\{u, v\} \mid u, v \in V\}$. Una arista e es un bucle, o lazo, si $f(e) = \{u, u\} = \{u\}$ para algún $u \in V$.

Grafos dirigidos:

Son pares ordenados, donde se admiten bucles, pero no se admiten aristas múltiples en la misma dirección entre dos vértices.

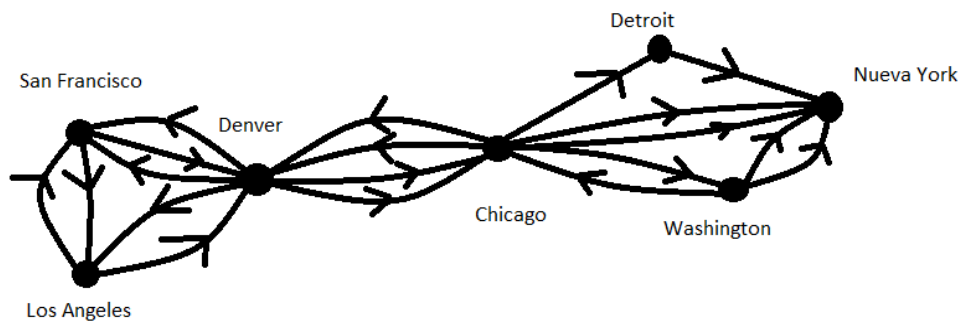
Definición: Un grafo dirigido $G = (V, E)$ consta de un conjunto V de vértices y de un conjunto E de aristas, que son pares ordenados de elementos de V .



Multígrafos dirigidos:

Puede que haya varias líneas unidireccionales desde cada nodo. Utilizaremos **multígrafos dirigidos**, que pueden tener aristas dirigidas múltiples desde un vértice a otro (que, eventualmente, puede coincidir con el primero).

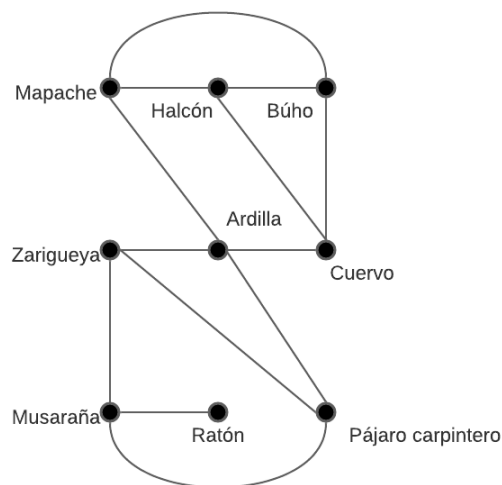
Definición: Un *multígrafo dirigido* $G = (V, E)$ consta de un conjunto V de vértices, un conjunto E de aristas y una función f de E en $\{(u, v) \mid u, v \in V\}$. Se dice que las aristas e_1 y e_2 son aristas múltiples si $f(e_1) = f(e_2)$.



MODELOS CON GRAFOS

1. **Grafos de solapamiento de nichos en Ecología:** Un grafo de solapamiento de nicho es un tipo de grafo que se utiliza para representar las relaciones entre especies que comparten recursos o nichos ecológicos similares en un ecosistema determinado. En un grafo de solapamiento de nicho, cada nodo representa una especie y las aristas representan la relación de solapamiento o compartición de recursos entre ellas.

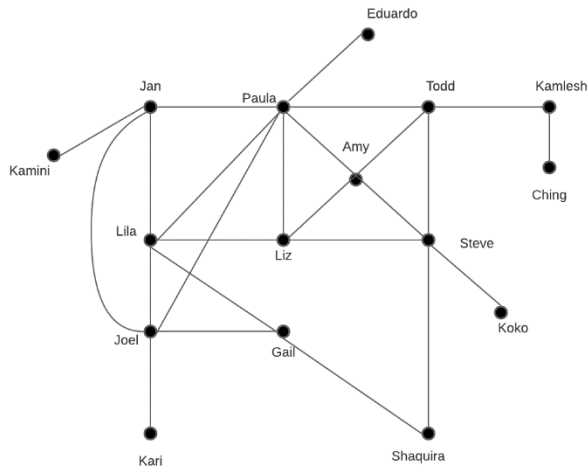
Ejemplo:



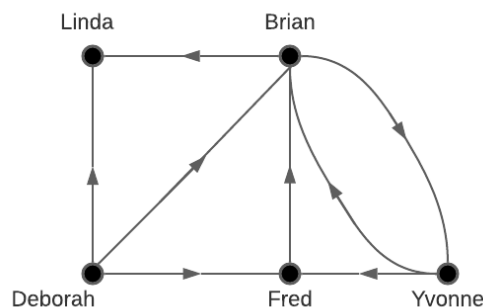
Como se observa, los mapaches y las ardillas compiten entre sí, mientras que las musarañas y los cuervos no lo hacen.

2. **Grafos de conocidos:** Podemos usar un grafo para representar el hecho de que dos personas se conozcan. Cada persona en grupo concreto se representa por un vértice. Se utiliza una arista no dirigida para conectar dos personas cuando éstas se conocen.

Ejemplo:



3. **Grafos de influencias:** Se ha observado en estudios del comportamiento de grupos que ciertas personas pueden influir en la forma en que piensan otras personas. Puede usarse un grafo dirigido, llamado grafo de influencias, para representar este comportamiento. Donde cada persona se representa por un vértice. Hay una arista dirigida del vértice a al b si la persona representada por el vértice a influye sobre la persona representada por el vértice b.



En este ejemplo se puede observar cómo Deborah puede influir sobre Brian, Fred y Linda, pero nadie puede influir en ella. Yvonne y Brian se influyen mutuamente.

TERMINOLOGÍA EN TEORÍA DE GRAFOS

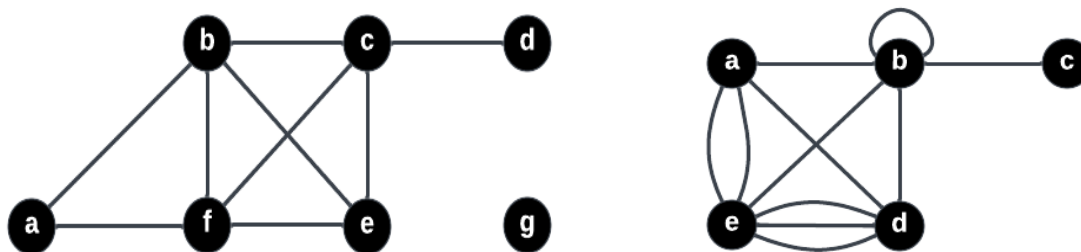
Terminología básica

Veamos en primer lugar la terminología que describe los vértices y las aristas de los grafos no dirigidos.

Definición 1: Se dice que los vértices u y v de un grafo no dirigido G son adyacentes en G si $\{u, v\}$ es una arista de G . Si $e = \{u, v\}$, se dice que la arista e es incidente con los vértices u y v . También se dice que la arista e conecta u y v . Se dice que los vértices u y v son extremos de la arista e .

Definición Dos: El grado de un vértice de un grafo no dirigido es el número de aristas incidentes con él, exceptuando los bucles, cada uno los cuales contribuye con dos unidades al grado del vértice. El grado del vértice v se denota por $\delta(v)$.

Ejemplo: ¿Cuáles son los grados de los vértices de los grafos G y H que se muestran a continuación?



Solución: En G $\delta(a) = 2$, $\delta(b) = 3$, $\delta(c) = 3$, $\delta(f) = 3$, $\delta(d) = 1$, $\delta(e) = 2$ y $\delta(g) = 0$.

En H , $\delta(a) = 2$, $\delta(b) = 3$, $\delta(e) = 2$, $\delta(c) = 1$, $\delta(d) = 3$.

A los vértices de grado cero se les llama aislados. Claramente, un vértice aislado no es adyacente a ningún vértice. El vértice g del grafo G del Ejemplo 1 es un vértice aislado. Se dice que un vértice es colgante, o que es una hoja, si, y sólo si, tiene grado uno. Por tanto, una hoja es adyacente a exactamente un vértice distinto de ella. El vértice d del grafo G del Ejemplo 1 es una hoja.

¿Qué obtenemos cuando sumamos los grados de todos los vértices de un grafo $G = (V, E)$?

Cada arista contribuye con dos unidades a la suma de los grados de los vértices, ya que cada arista es incidente con dos vértices (posiblemente iguales). Esto significa que la suma de los grados de los vértices es el doble del número de aristas.

Enunciamos este resultado en el Teorema 1, al que también se conoce como teorema de los apretones de manos, debido a la analogía entre los dos extremos que tiene cada arista y las dos manos que se estrechan cuando dos personas se saludan.

EL TEOREMA DE LOS APRETONES DE MANOS

Sea $G = (V, E)$ un grafo no dirigido con e aristas. Entonces,

$$2e = \sum_{v \in V} \delta(v)$$

Ejemplo: ¿Cuántas aristas hay en un grafo con diez vértices, cada uno de los cuales tiene grado seis?

Solución: Como la suma de los grados de los vértices es $6 \cdot 10 = 60$, se sigue que $2e = 60$. Por tanto, $e = 30$.

En el teorema 1 nos dice que la suma de los grados de los vértices de un grafo no dirigido es un número par. Este hecho tiene muchas consecuencias, una de las cuales se da en el teorema 2.

Teorema Dos: Todo grafo no dirigido tiene un número par de vértices de grado impar.

Demostración: Sean V_1 y V_2 el conjunto de vértices de grado par y el conjunto de vértices de grado impar, respectivamente, de un grafo no dirigido $G = (V, E)$. Entonces,

$$2e = \sum_{v \in V} \delta(v) = \sum_{v \in V_1} \delta(v) + \sum_{v \in V_2} \delta(v)$$

Como $\delta(v)$ es par si $v \in V_1$, el primer sumando del término de la derecha de la última igualdad es par. Además, la suma de los dos sumandos de dicho término es par, puesto que esa suma es $2e$. Por tanto, el segundo sumando también es par. Como todos los términos que se suman en ese segundo sumando son impares. Tiene que haber un número par de ellos. Por tanto, hay un número par de vértices de grado impar.

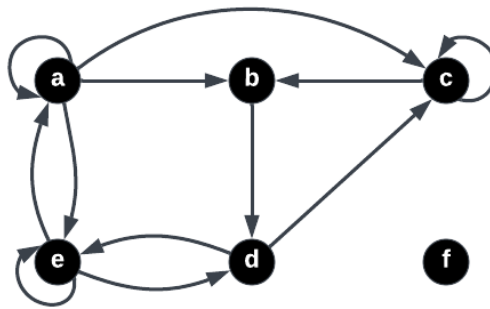
Grafos dirigidos: A las aristas de un grafo dirigido se les asigna un sentido.

Definición: Si (u, v) es una arista del grafo dirigido G , se dice que, u es adyacente a v y que v es adyacente desde u . Al vértice u se le llama vértice inicial de (u, v) y a v se le llama vértice final o terminal de (u, v) . Los vértices inicial y final de un bucle coinciden.

Como las aristas de un grafo dirigido son pares ordenados, podemos refinar la definición de grado de un vértice a fin de reflejar el número de aristas que tienen a ese vértice como vértice inicial o como vértice final.

Definición: En un grafo dirigido, el grado de entrada de un vértice v , denotado por $\delta^-(v)$, es el número de aristas que tienen a v como vértice final. El grado de salida de un vértice v , denotado por $\delta^+(v)$, es el número de aristas que tienen a v como vértice inicial (nótese que un bucle contribuye con una unidad tanto al grado de entrada como al grado de salida del vértice correspondiente).

Ejemplo: Halla los grados de entrada y de salida de cada vértice del grafo dirigido G que se muestra a continuación.

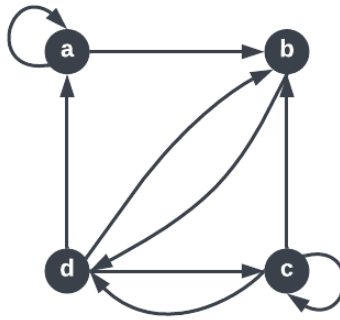


Solución: Los grados de entrada de G son: $\delta^-(a) = 2, \delta^-(b) = 2, \delta^-(c) = 3, \delta^-(d) = 2, \delta^-(e) = 3$ y $\delta^-(f) = 0$. Los grados de salida son: $\delta^+(a) = 4, \delta^+(b) = 1, \delta^+(c) = 2, \delta^+(d) = 2, \delta^+(e) = 3, \delta^+(f) = 0$.

Como cada arista tiene un vértice inicial y un vértice final, la suma de los grados de entrada y la suma de los grados de salida de todos los vértices del grafo dirigido coinciden. Ambas sumas son iguales al número de aristas que tiene el grafo.

Teorema Tres.

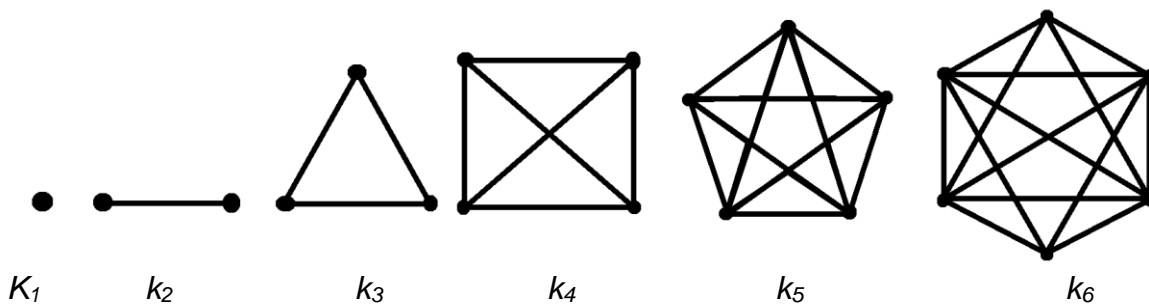
$$\sum_{v \in V} \delta^-(v) + \sum_{v \in V} \delta^+(v) = 2|E|$$



Familias Distinguidas de Grafos Simples

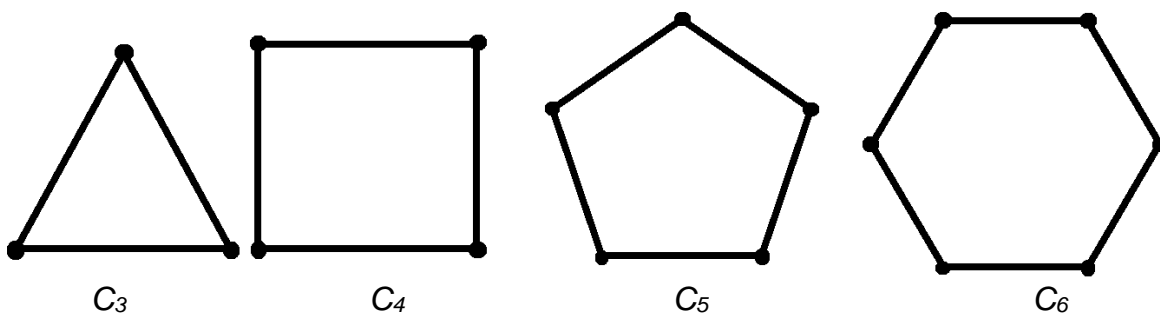
Ejemplo 4

Grafos Complejos: El grafo completo de n vértices, que se denota por K_n , es el grafo simple que contiene exactamente una arista entre cada par de vértices distintos. En la siguiente figura se muestran los grafos K_n para $n = 1, 2, 3, 4, 5, 6$.



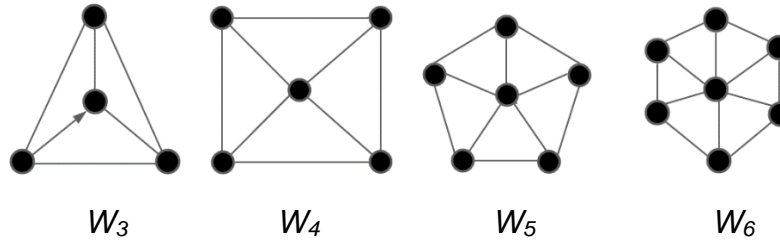
Ejemplo 5

Ciclos: El ciclo C_n , $n \geq 3$, consta de n vértices v_1, v_2, \dots, v_n y aristas $\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{n-1}, v_n\}$ y $\{v_n, v_1\}$. En la siguiente figura se muestran los ciclos C_3, C_4, C_5 y C_6 .



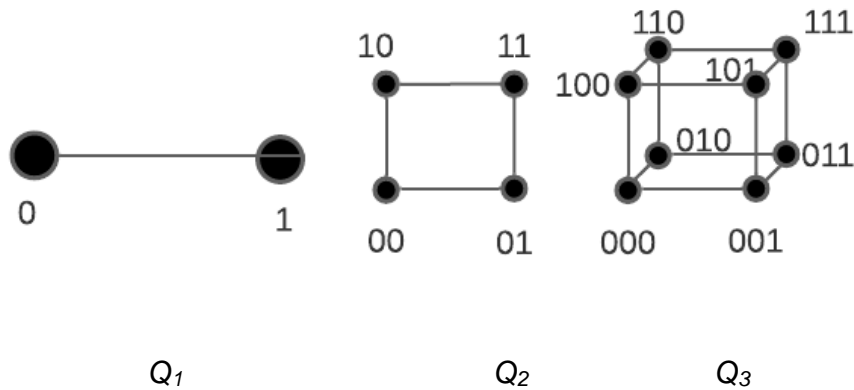
Ejemplo 6

Ruedas: Obtenemos la rueda W_n cuando añadimos un vértice adicional al ciclo C_n , para $n \geq 3$, y conectados este nuevo vértice con cada uno de los n vértices de C_n mediante una nueva arista. En la siguiente figura se muestran las ruedas W_3, W_4, W_5 y W_6 .



Ejemplo 7

n-Cubos: El cubo n -dimensional, o n -cubos, denotado por Q_n , es el grafo cuyos vértices representan las 2^n cadenas de bits de longitud n . Dos vértices son adyacentes si, y sólo si, las cadenas de bits a las que representan difieren exactamente en un bit. En la figura que se verá a continuación se muestran los grafos Q_1 , Q_2 y Q_3 . Nótese que se puede construir el $(n + 1)$ -cubo Q_{n+1} a partir del n -cubo Q_n haciendo dos copias de Q_n , anteponiendo un 0 a cada una de las etiquetas de los vértices de una de las copias de Q_n , anteponiendo un 1 en la otra copia y añadiendo aristas que conecten dos vértices cuyas etiquetas se defirieren únicamente en el primer bit. En la figura que se muestra a continuación, Q_3 , se construye a partir de Q_2 colocando dos copias de Q_2 como las caras superior e inferior de Q_3 , añadiendo un 0 al principio de la etiqueta de cada vértice de la cara inferior y añadiendo un 1 al principio de la etiqueta de cada vértice de la cara superior.



Grafos Bipartitos

En matemáticas discretas, un grafo bipartito es un tipo especial de grafo en el cual los vértices pueden ser divididos en dos conjuntos disjuntos (no se tienen vértices en común) de manera que todas las aristas del grafo conectan vértices de diferentes conjuntos. En otras palabras, un grafo bipartito es un grafo cuyos vértices pueden ser coloreados con dos colores distintos de manera que no haya aristas que conecten vértices del mismo color.

Formalmente, un grafo $G = (V, E)$ se dice que es bipartito si su conjunto de vértices V puede ser particionado en dos conjuntos disjuntos V_1 y V_2 , es decir, $V = V_1 \cup V_2$ y $V_1 \cap V_2 = \emptyset$, de manera que cada arista en el conjunto de aristas E conecta un vértice en V_1 con un vértice en V_2 .

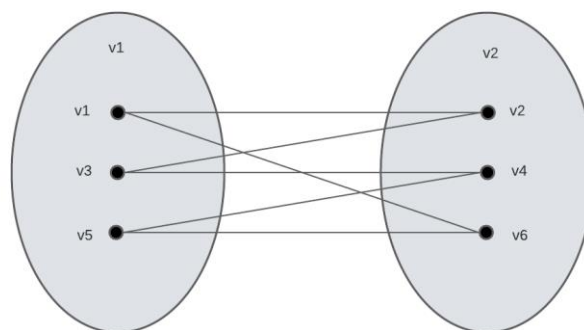
Los grafos bipartitos son ampliamente utilizados en diferentes aplicaciones prácticas, como asignación de tareas, emparejamiento de elementos, modelado de redes sociales, y problemas de flujo máximo, entre otros. Además, tienen propiedades y algoritmos específicos que los hacen de interés en la teoría de grafos y en la resolución de problemas computacionales.

Definición 5

Se dice que un grafo simple G es bipartito si su conjunto de vértices V se puede dividir en dos conjuntos disjuntos V_1 y V_2 tales que cada arista del grafo conecta un vértice de V_1 con un vértice de V_2 .

Ejemplo 8

C_6 es bipartito, como se muestra en la siguiente figura, ya que su conjunto de vértices puede dividirse en dos conjuntos, $V_1 = \{v_1, v_3, v_5\}$ y $V_2 = \{v_2, v_4, v_6\}$, y cada arista de C_6 conecta un vértice de V_1 con un vértice de V_2 .

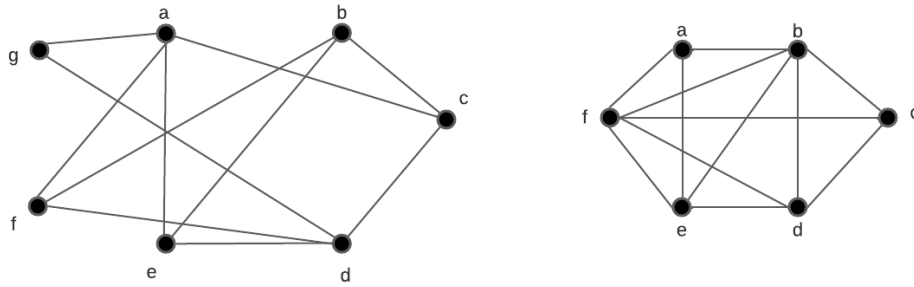


Ejemplo 9

Para verlo, nótese que, si partimos el conjunto de vértices de K_3 en dos conjuntos disjuntos, uno de los conjuntos debe contener dos vértices. Si el grafo fuese bipartito, esos dos vértices no podrían estar conectados por ninguna arista, pero en K_3 cada vértice está conectado con cualquier otro vértice por medio de una arista.

Ejemplo 10

¿Son bipartitos los grafos G y H que se muestran en la siguiente figura?



G

H

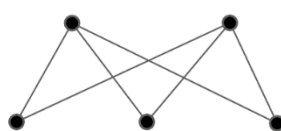
El grafo G es bipartito, puesto que su conjunto de vértices es la unión de dos conjuntos disjuntos, $\{a, b, d\}$ y $\{c, e, f, g\}$, y cada arista conecta un vértice de uno de estos subconjuntos con un vértice del otro subconjunto.

El grafo H no es bipartito, ya que su conjunto de vértices no se puede dividir en dos subconjuntos de modo que las aristas no conecten ningún par de vértices de un mismo subconjunto.

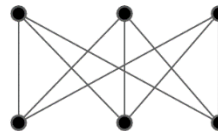
Ejemplo 11

Grafos bipartitos completos: El grafo bipartito completo $K_{m, n}$ es el grafo cuyo conjunto de vértices está formado por dos subconjuntos con m y n vértices, respectivamente, y hay una arista entre dos vértices si, y solo si, un vértice está en el primer subconjunto y el otro vértice está en el segundo subconjunto. En la siguiente figura no se muestra los grafos bipartitos completos $K_{2, 3}$, $K_{3, 3}$, $K_{3, 5}$ y $K_{2, 6}$.

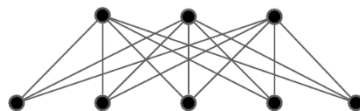
6.



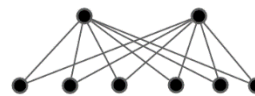
$K_{2, 3}$



$K_{3, 3}$



$K_{3, 5}$



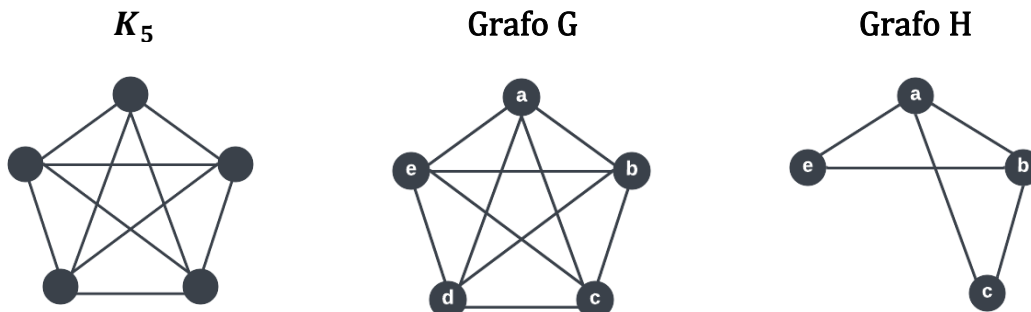
$K_{2, 6}$

GRAFOS DEFINIDOS A PARTIR DE OTROS

Al eliminar algunos vértices y aristas que no sean de interés, sin eliminar ningún extremo de las aristas que quedan, se obtiene un grafo más pequeño. Se dice que este grafo es un **subgrafo** del grafo original

Definición: Un subgrafo de un grafo $G = (V, E)$ es un grafo $H = (W, F)$ con $W \subseteq V$ y $F \subseteq E$.

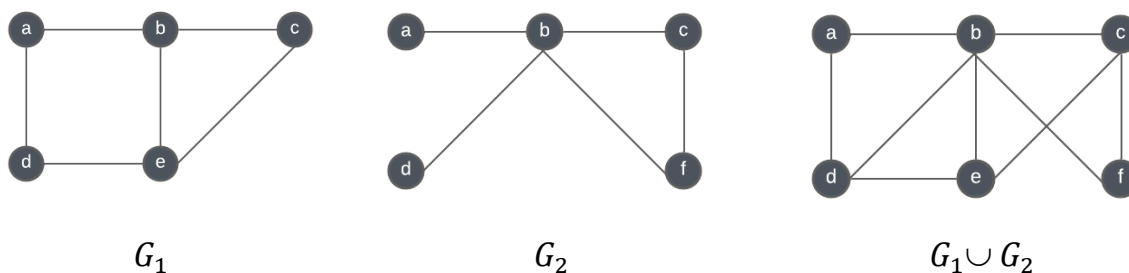
Ejemplo: El grafo G que se muestra en la siguiente figura, es un subgrafo de K_5 .



Podemos combinar dos o más grafos de varias formas. El nuevo grafo que contiene todos los vértices y todas las aristas de estos grafos se llama **unión** de los grafos. A continuación, se verá una definición más formal para la unión de dos grafos simples.

Definición: La unión de dos grafos simples $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ es un grafo simple cuyo conjunto de vértices es $V_1 \cup V_2$ y cuyo conjunto de aristas es $E_1 \cup E_2$ se denota por $G_1 \cup G_2$.

Ejemplo: Halla la unión de grafos $G_1 \cup G_2$ que se muestran en la siguiente figura



REPRESENTACIÓN DE GRAFOS E ISOMORFISMO DE GRAFOS

INTRODUCCIÓN

Hay muchas maneras útiles de representar los grafos. A veces, dos grafos tienen la misma forma, en el sentido de que hay una biyección entre sus conjuntos de vértices que preserva las aristas. En este caso, decimos que los dos grafos son **isomorfos**. El determinar si dos grafos son isomorfos o no es un problema importante en la teoría de grafos que se estudiarán en la siguiente sección.

REPRESENTACIÓN DE GRAFOS

Una forma de representar grafos sin aristas múltiples es enumerar todas las aristas del grafo. Otra forma de representar grafos sin aristas múltiples es utilizar **listas de adyacencia**, que especifican los vértices que son adyacentes a cada uno de los vértices del grafo.

Ejemplo: Utilizar las listas de adyacencia para describir el grafo simple de la siguiente.

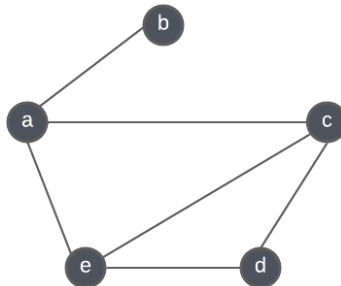


Figura: Un grafo simple.

Tabla: Una lista de aristas para un grafo simple	
Vértices	Vértices adyacentes
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d

Ejemplo: Representar el grafo dirigido de la siguiente figura enumerando todos los vértices que son vértices finales de las aristas que salen de cada uno de los vértices del grafo.

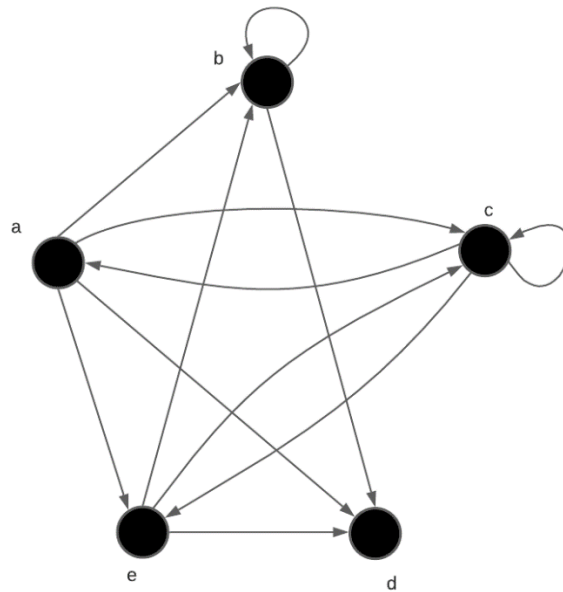


Figura: Un grafo dirigido.

Tabla: Una lista de aristas para un grafo dirigido	
Vértices	Vértices finales
a	b, c, d, e
b	b, d
c	a, c, e
d	
e	b, c, d

MATRICES DE ADYACENCIA

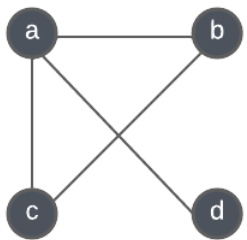
Supongamos que $G = (V, E)$ es un grafo simple $|V| = n$. Supongamos que los vértices de G se enumeran de manera arbitraria como v_1, v_2, \dots, v_n . **La matriz de adyacencia A** (o A_G) de G con respecto a este listado de los vértices es la matriz booleana $n \times n$ que tiene un 1 en la posición (i, j) , si v_i y v_j no son adyacentes. En otras palabras, si su matriz de adyacencia es $A = [a_{ij}]$, entonces:

$$a_{ij} = \begin{cases} 1 & \text{si } \{v_i, v_j\} \text{ es una arista de } G. \\ 0 & \text{en el caso contrario.} \end{cases}$$

Nótese que la matriz de adyacencia del grafo depende del orden elegido para los vértices. Por tanto, hay $n!$ Matrices de adyacencia distintas para un grafo de n vértices, puesto que hay $n!$ Ordenaciones distintas de los n vértices.

Nótese que si hay relativamente pocas aristas en el grafo, la matriz de adyacencia es una matriz **poco densa**, esto es, una matriz con pocos elementos no nulos. Hay técnicas especiales para representar estas matrices, así como para realizar cálculos con ellas. También hay ocasiones en que puede ser más eficiente trabajar con listas de aristas a la hora de representar y de trabajar con este tipo de grafos.

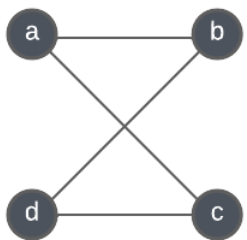
Ejemplo: Utilizar una matriz de adyacencia para representar el grafo que se muestra en la siguiente figura.



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

Figura: Un grafo simple.

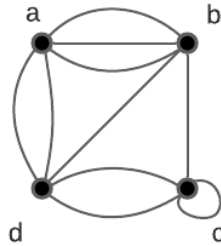
Ejemplo: Dibuja un grafo con matriz de adyacencia con respecto a la ordenación de vértices a, b, c, d .



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

Las matrices de adyacencia pueden usarse también para representar grafos no dirigidos con bucles y con aristas múltiples. Un bucle en el vértice a_i se representa por medio de un 1 en la posición (i,i) de la matriz de adyacencia. Cuando hay aristas múltiples, la la matriz de adyacencia deja de ser booleana, ya que el elemento en la posición (i,j) de esta matriz es igual al número de aristas asociadas con $\{a_i, a_j\}$. Todos los grafos no dirigidos, incluyendo multigrafos y pseudografos, tienen matrices de adyacencia simétricas.

Ejemplo: Utiliza una matriz de adyacencia para representar el pseudografo que se muestra en la siguiente figura.



Solución: La matriz de adyacencia con respecto al orden a, b, c, d es:

$$\begin{bmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 3 & 1 & 2 & 0 \end{bmatrix}$$

La matriz de adyacencia de un grafo dirigido no tiene por qué ser simétrica ya que puede no haber una arista a_j a a_i cuando hay una arista a_i a a_j .

Las matrices de adyacencia se pueden emplear también para representar multigrafos dirigidos. De nuevo, estas matrices ya no son booleanas si hay aristas múltiples en el mismo sentido conectando dos vértices. En la matriz de adyacencia de un multigrafo dirigido, a_{ij} es igual al número de aristas asociadas con (v_i, v_j) .

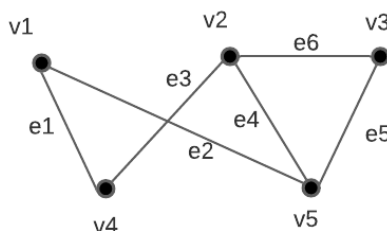
MATRICES DE INCIDENCIA

Otra representación de grafos que se usa con frecuencia es la que emplea **matrices de incidencia**.

Sea $G = (V, E)$ un grafo no dirigido. Supongamos que v_1, v_2, \dots, v_n son los vértices y e, e_2, \dots, e_m las aristas de G . Entonces, la matriz de incidencia con respecto a este ordenamiento de V , y de E , es la matriz $M = [m_{ij}]$ de $n \times m$ dada por:

$$a_{ij} = \begin{cases} 1 & \text{si la arista } e_j \text{ es incidente con } v_i \\ 0 & \text{en el caso contrario.} \end{cases}$$

Ejemplo: Representa por medio de una matriz de incidencia el grafo que se muestra en la siguiente figura:

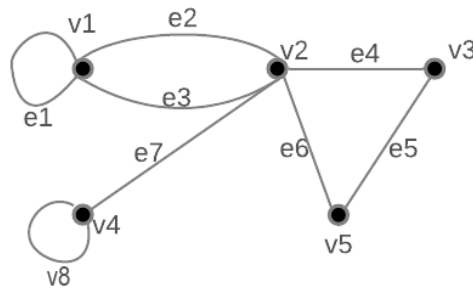


Solución: La matriz de incidencia es:

$$\begin{matrix} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

Las matrices de incidencia pueden usarse también para representar aristas múltiples y bucles. Las aristas múltiples se representan en la matriz de incidencia mediante columnas con todos sus elementos idénticos, puesto que dichas aristas son incidentes con el mismo par de vértices. Los bucles se representan por medio de una columna con un único elemento igual a 1, que corresponde al vértice con el que es incidente el bucle.

Ejemplo: Representar el pseudografo que se muestra en la siguiente figura usando una matriz de incidencia.



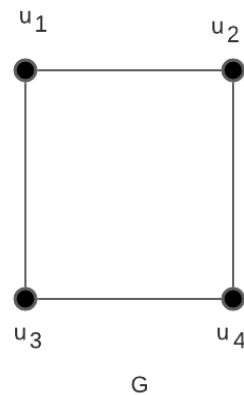
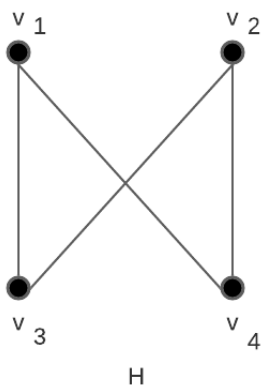
Solución: La matriz de incidencia para este grafo es:

$$\begin{array}{c}
 e_1 \quad e_2 \quad e_3 \quad e_4 \quad e_5 \quad e_6 \quad e_7 \quad e_8 \\
 \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}
 \end{array}$$

ISOMORFISMO DE GRAFOS

Definición 1. Los grafos simples $G_1 = (V, E)$ y $G_2 = (V_2, E_2)$ son isomorfos si hay una función biyectiva f de V_1 en V_2 con la propiedad de que, para cada par de vértices $u, v \in V_1$, u y v son adyacentes en G_1 si, y solo si, $f(u)$ y $f(v)$ son adyacentes en G_2 . Se dice que esta función f es un isomorfismo.

Ejemplo: Demuestra que los grafos $G = (V, E)$ y $H = (W, F)$ que se muestran son isomorfos.

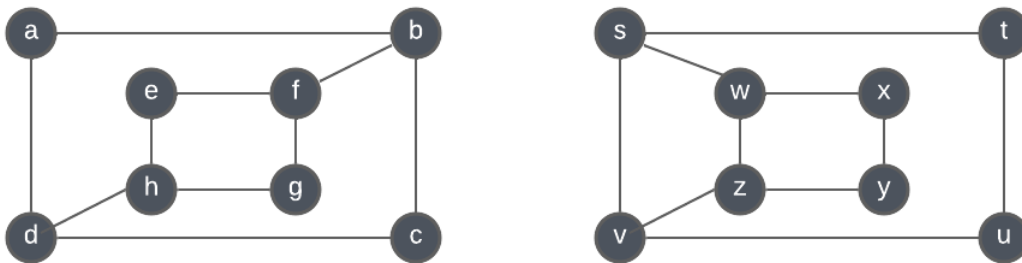


Solución: La función f con $f(u_1) = v_1, f(u_2) = v_4, f(u_3) = v_3$ y $f(u_4) = v_2$ es una función biyectiva entre V y W . Para ver que esta función preserva la adyacencia, nótese que los pares adyacentes en G son u_1 y u_2, u_1 y u_3, u_2 y u_4, u_3 y u_4 y cada uno de los pares $f(u_1) = v_1$ y $f(u_2) = v_4, f(u_1) = v_1$ y $f(u_3) = v_3, f(u_2) = v_4$ y $f(u_4) = v_2, f(u_3) = v_3$ y $f(u_4) = v_2$, son adyacentes en H .

A menudo es difícil determinar si dos grafos simples son o no isomorfos. Hay $n!$ posibles biyecciones entre los conjuntos de vértices de dos grafos simples de n vértices. Comprobar cada una de estas funciones para ver si preserva o no las adyacencias es un método poco práctico cuando n es grande.

Sin embargo, con frecuencia podemos demostrar que dos grafos no son isomorfos demostrando que no comparten alguna propiedad que dos grafos isomorfos deberían tener en común.

Ejemplo: Determina si los grafos que se muestran en la siguiente figura son o no isomorfos.

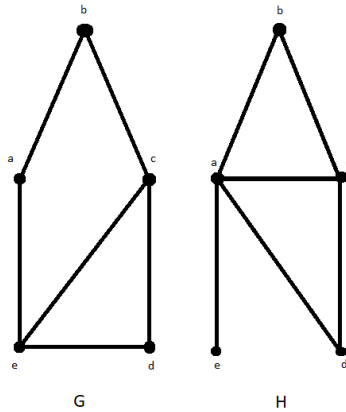


Solución: Ambos grafos G y H tienen ocho vértices y diez aristas. Ambos tienen también cuatro vértices de grado dos y cuatro de grado tres. Como todos estos invariantes coinciden, podemos pensar que estos grafos son isomorfos.

Sin embargo, G y H no son isomorfos. Para verlo, nótese que como $\delta(a) = 2$ en G , a debería corresponder a uno de los vértices t, u, x o y de H , ya que éstos son los vértices de grado dos de H . Sin embargo, cada uno de estos cuatro vértices de H es adyacente a otro vértice de grado de H , lo que no es cierto para a en el grafo G .

Ejemplo

Demuestra que los grafos que se muestran en la siguiente figura no son isomorfos.



Tanto G como H tienen cinco vértices y seis aristas. Sin embargo, H tiene un vértice de grado uno, más concretamente el vértice e, mientras que G no tiene vértices de grado uno. Se sigue que G y H no son isomorfos.

El número de vértices, el número de aristas y los grados de las aristas son invariantes bajo isomorfismo. Si cualquiera de estas cantidades toma valores distintos en dos grafos simples, dichos grafos no pueden ser isomorfos. Sin embargo, el que esos invariantes coincidan no quiere decir que esos dos grafos sean necesariamente isomorfos. En la actualidad, no se conoce ningún conjunto de invariantes que pueda usarse para determinar si dos grafos simples son o no isomorfos.

CONEXIÓN

Hay varios problemas los cuales se pueden representar por medio de caminos he ir recorriendo las aristas de un grafo. Por ejemplo, si se puede enviar o no un mensaje entre dos ordenadores usando enlaces intermedios, planificar de manera eficiente la ruta de repartición de correo, recogida de basura, diagnósticos de redes en ordenadores y muchos más pueden resolverse utilizando modelos que involucran caminos definidos sobre grafos.

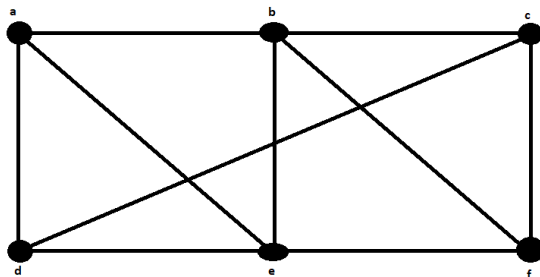
CAMINOS:

Es una secuencia de aristas que comienza en un vértice del grafo y recorre ciertas aristas conectando pares de **vértices adyacentes**.

Definición: Sea n un entero no negativo y sea G un grafo no dirigido. Un camino de longitud n y u a v en G es una secuencia de n aristas a_1, a_2, \dots, a_n de G tal que $f(a_1) = \{x_0, x_1\}$, $f(a_2) = \{x_1, x_2\}$, ..., $f(a_n) = \{x_{n-1}, x_n\}$ donde $x_0 = u$ y $x_n = v$. Si el grafo es simple, denotamos el camino por su secuencia de vértices x_0, x_1, \dots, x_n (ya que al enumerar estos vértices determina el camino de forma única). El camino es un circuito si comienza y termina en el mismo vértice, esto es, si $u = v$, y tiene la longitud mayor que cero. Se dice que el camino o circuito pasa por los vértices x_1, x_2, \dots, x_{n-1} o también que recorre las aristas a_1, a_2, \dots, a_n . Un camino o circuito es simple si no contiene la misma arista más de una vez.

Ejemplo:

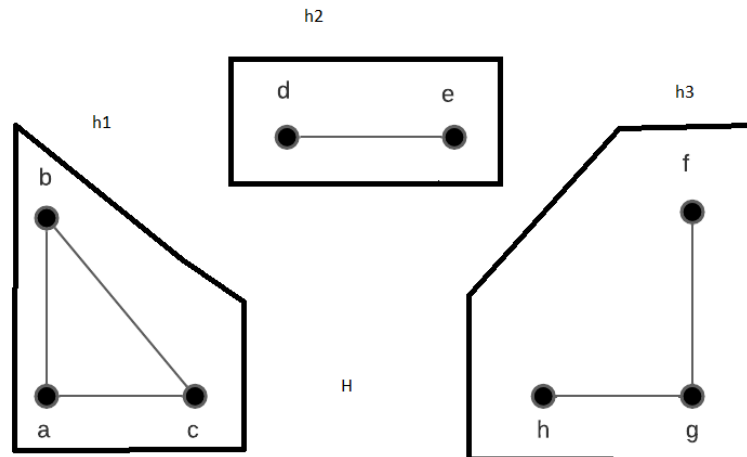
a, d, c, f, e es un camino simple de longitud 4, ya que $\{a, d\}, \{d, c\}, \{c, f\}$ y $\{f, e\}$ son aristas. Sin embargo, d, e, c, a no es un camino, ya que $\{e, c\}$ no es una arista. Nótese que b, c, f, e, b es un circuito de longitud 4, ya que $\{b, c\}, \{c, f\}$, y $\{e, b\}$ son aristas, y este camino comienza y termina en b . El camino a, b, e, d, a, b que tiene longitud 5, no es simple, ya que contiene 2 veces la arista $\{a, b\}$.



Teorema 1: hay un camino simple entre cada par de vértices distintos de un grafo no dirigido conexo.

Un grafo que no es conexo es la unión de dos o más subgrafos conexos que dos a dos no tienen ningún vértice en común. A estos subgrafos conexos disjuntos se les llama “componentes conexas” del grafo.

Ejemplo: ¿Cuáles son las componentes conexas del grafo H de la siguiente figura?



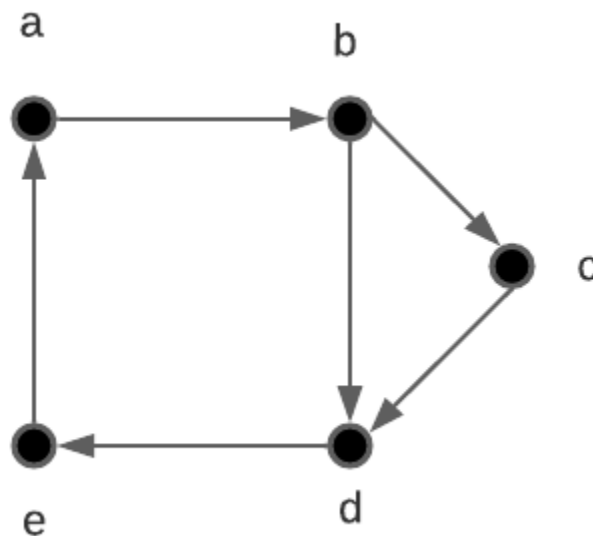
Solución: El grafo H es la unión de tres subgrafos conexos disjuntos h1, h2 y h3.

CONEXIÓN EN GRAFOS DIRIGIDOS:

Existen dos nociones de grafos dirigidos dependiendo si se considera o no las direcciones de las aristas:

- Se dice que un grafo dirigido es **fuertemente conexo** si hay un camino de a a b y un camino de b a a para cualesquiera dos vértices a y b del grafo.
- Se dice que un grafo dirigido **débilmente conexo** si hay un camino entre cada dos vértices del grafo no dirigido subyacente.

Ejemplo: ¿es fuertemente conexo el siguiente grafo dirigido?



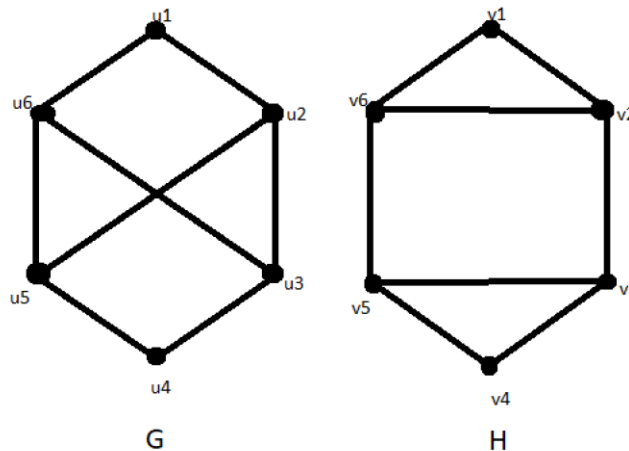
Solución: Es débilmente porque hay un camino entre cada dos vértices de este grafo dirigido, por lo tanto, es débilmente conexo.

CAMINOS E ISOMORFISMO

Hay muchas maneras en las que los caminos y circuitos pueden ayudarnos a determinar si dos grafos son o no isomorfos. Por ejemplo, la Existencia de un circuito simple de una longitud concreta es un invariante útil que se puede emplear a la hora de mostrar que dos grafos no son isomorfos.

Además, podemos hacer uso de los caminos a la hora de construir posibles isomorfismos. Como ya hemos dicho, un invariante bajo isomorfismo útil para grafos simples es la existencia de un circuito simple de longitud K , siendo K un entero positivo mayor que 2.

Ejemplo: Determina si los grafos G y H de la siguiente figura son o no isomorfos.

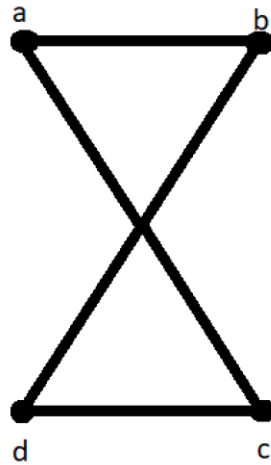


Solución: Tanto G como H tienen seis vértices y ocho aristas. Ambos tienen cuatro vértices de grado tres y dos vértices de grado dos. Por tanto, los tres invariantes (número de vértices, número de aristas y grados de los vértices) coinciden en los dos grafos. Sin embargo, H contiene un circuito simple de longitud tres, a saber, v_1, v_2, v_6, v_1 , mientras que G no contiene ningún circuito simple de longitud tres, lo que puede comprobarse por inspección (todos los circuitos simples de G tienen longitud al menos cuatro). Como la existencia de un circuito simple de longitud tres es un invariante bajo isomorfismo, G y H no son isomorfos.

EL NÚMERO DE CAMINOS ENTRE DOS VERTICES

Teorema 2: Sea G un grafo y sea A su matriz de adyacencia con respecto a la ordenación v_1, v_2, \dots, v_n (se admiten aristas dirigidas o no dirigidas, aristas múltiples o bucles). El número de caminos distintos de longitud r de v_i a v_j , siendo r un entero positivo, es igual al elemento de la posición (i, j) de la matriz A^r .

Ejemplo: ¿Cuántos caminos de longitud 4 hay entre a y d en el grafo simple de la siguiente figura?



Solución: La matriz de la adyacencia de G (ordenado los vértices de la forma a, b, c, d) es:

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

Por lo tanto, el número de caminos de longitud 4 entre a y d es el elemento $(1, 4)$ de A^4 . Como:

$$A^4 = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

Hay exactamente ocho caminos de longitud 4 entre a y d . Por inspección del grafo, vemos que a, b, a, b, d ; a, b, a, c, d ; a, b, d, b, d ; a, b, d, c, d ; a, c, a, b, d ; a, c, d, b, d y a, c, d, c, d son los ocho caminos entre a y d .

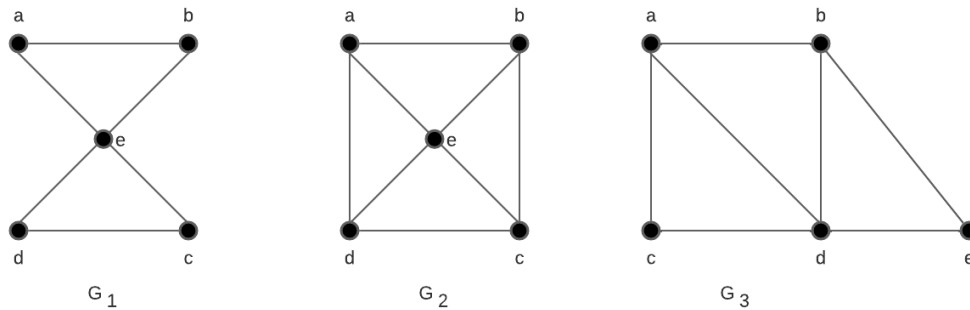
CAMINOS EULERIANOS Y HAMILTONIANOS

Introducción

Un camino euleriano es un recorrido en un grafo que visita cada arista exactamente una vez. En otras palabras, es un camino que comienza y termina en el mismo vértice y recorre todas las aristas del grafo una sola vez. Si un grafo tiene un camino euleriano, se dice que es un grafo euleriano.

Ejemplo 1.

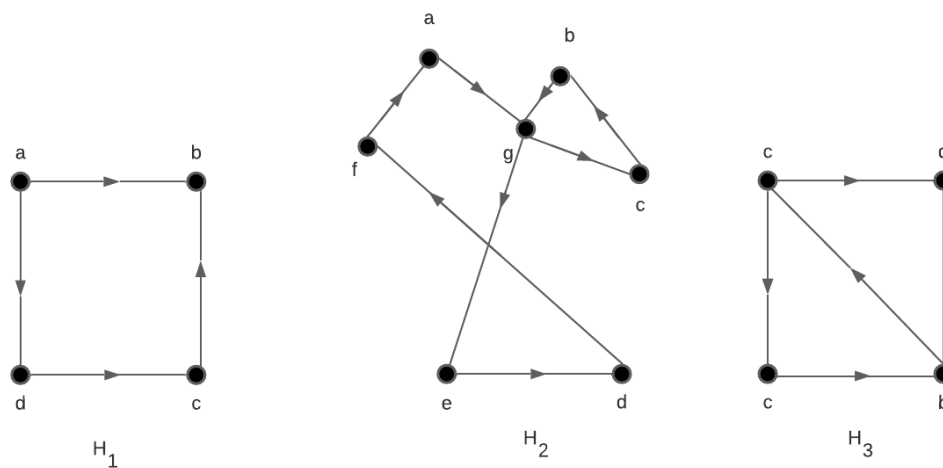
¿Cuáles de los grafos no dirigidos de la Figura 3 contienen un circuito euleriano? Entre aquellos que no lo contienen. ¿Cuáles contienen un camino euleriano?



Solución: El grafo G_1 , contiene un circuito euleriano, por ejemplo, a, e, c, d, e, b, a . Ni G_2 ni G_3 , contienen un circuito euleriano (el lector debería comprobarlo). No obstante, G_3 contiene un camino euleriano, a saber, a, c, d, e, b, d, a, b . El grafo G_2 no contiene ningún camino euleriano (como el lector puede comprobar).

Ejemplo 2.

¿Cuáles de los grafos no dirigidos de la Figura 3 contienen un circuito euleriano? Entre aquellos que no lo contienen. ¿Cuáles contienen un camino euleriano?



Solución: El grafo H_2 contiene un circuito euleriano, por ejemplo, $a, g, c, b, g, e, d, f, a$. Ni H_1 ni H_3 contienen un circuito euleriano. El grafo H_3 contiene un camino no euleriano, a saber, c, a, b, c, d, b , pero H_1 no.

A tener en cuenta: Un multigrafo se considera conexo si existe un camino entre cada par de nodos. Es decir, cualquier nodo del multigrafo se puede alcanzar desde cualquier otro nodo del multigrafo. Un circuito euleriano es un recorrido que visita cada arista del multigrafo exactamente una vez y termina en el mismo nodo donde empezó. Es decir, es un recorrido cerrado que cubre todas las aristas del multigrafo.

CONDICIONES NECESARIAS Y SUFICIENTES PARA LA EXISTENCIA DE CIRCUITOS Y CAMINOS EULERIANOS

Hay criterios sencillos para determinar si un multígrafo contiene un circuito o un camino euleriano. Euler los descubrió al resolver el famoso problema de los puentes de Königsberg. Supondremos que todos los grafos que aparecen en esta sección tienen un número finito de vértices y de aristas.

¿Qué podemos decir de un multígrafo conexo si contiene un circuito euleriano? Lo que podemos demostrar es que todos los vértices tienen grado par. Para ver esto, observamos primero que un circuito euleriano se inicia en un vértice a y continúa con una arista incidente con a . digamos $\{a, b\}$ La arista $\{a, b\}$ contribuye con uno al grado de a . Cada vez que el circuito pasa por un vértice contribuye con dos unidades al grado de ese vértice, ya que el circuito entra en el vértice por una arista incidente con este vértice y sale de él por medio de otra arista incidente. Finalmente, el circuito termina donde comenzó, de nuevo contribuyendo con una unidad al grado de a . Por tanto, $S(a)$ tiene que ser par, puesto que el circuito contribuye con uno cuando comienza, con uno cuando acaba y con dos cada vez que pasa por a (si es que lo hace). Un vértice distinto de a tiene grado par porque el circuito contribuye con dos a su grado cada vez que pasa por el vértice. La conclusión es que, si un grafo conexo contiene un circuito euleriano, entonces todos sus vértices tienen grado par.

¿Esta condición necesaria para la existencia de un circuito euleriano es también suficiente”? Esto es, ¿tiene que existir un circuito euleriano en un multígrafo conexo con todos los vértices de grado par? Vamos a responder afirmativamente a esta pregunta por medio de un argumento constructivo, Supongamos que G es un multigrafo conexo y que el grado de cada uno de los vértices de G es par. Vamos a formar un circuito simple que comienza en un vértice arbitrario a de G . Sea $X_0 = a$. Primero elegimos arbitrariamente una arista (X_0, X_1) incidente con a . Continuamos construyendo un camino simple $\{X_0, X_1\}, \{X_1, X_2\}, \dots, \{X_{n-1}, X_n\}$ tan largo como sea posible, Por ejemplo, en el grafo de la Figura 5 comenzamos en a y elegimos sucesivamente las aristas $\{a, f\}, \{f, c\}, \{c, b\}$ y $\{b, a\}$.

El camino se acaba puesto que el grafo tiene un número finito de aristas. Comienza en a con una arista de la forma $\{a, x\}$ y termina en a con una arista de la forma $\{y, a\}$. Esto es así porque cada vez que el camino atraviesa un vértice con grado par usa una sola arista cuando entra, de manera que queda al menos otra arista para que el camino abandone ese vértice. Puede que este camino haga uso de todas las aristas o puede que no.

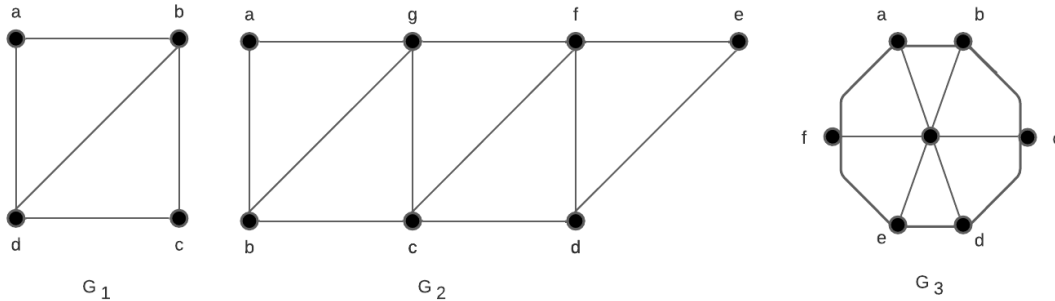
Teorema 1

Un multígrafo conexo contiene un circuito euleriano si, y sólo si, cada uno de sus vértices tiene grado par.

Teorema 2

Un multígrafo conexo contiene un camino euleriano, pero no un circuito euleriano, si y solo si, tiene exactamente dos vértices de grado impar.

Ejemplo: ¿Cuáles de los grados de la Figura 7 contienen un camino euleriano?



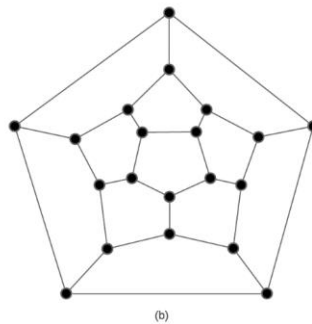
Solución: G_1 tiene exactamente dos vértices de grado impar, que son b y d . Por tanto, contiene un camino euleriano de extremos b y d . Un camino euleriano con esos extremos es d, a, b, c, d, b .

Análogamente, G_2 tiene exactamente dos vértices de grado impar, que son b y d . Por tanto, contiene un camino euleriano cuyos extremos son b y d . Un camino euleriano con esos extremos es $b, a, g, f, e, d, c, g, b, c, f, d$. El grafo G_3 no contiene ningún camino euleriano porque tiene seis vértices de grado impar.

CAMINOS Y CIRCUITOS HAMILTONIANOS

Origen

El origen de esta terminología es un juego, el juego icosiano, inventado en 1857 por el matemático irlandés Sir Willian Rowan Hamilton, Consistía en un dodecaedro de madera [un poliedro de 12 caras, cada una de las cuales es un pentágono regular, como se ve en la Figura con un alfiler saliendo de cada vértice del dodecaedro y en un trozo de cuerda. Los 20 vértices del dodecaedro estaban etiquetados con el nombre de distintas ciudades del mundo. El objetivo del juego era comenzar en una ciudad, viajar siguiendo las aristas del dodecaedro visitando cada una de las otras 19 ciudades exactamente una vez y terminar el viaje en la primera ciudad. El circuito seguido se marcaba utilizando la cuerda y los alfileres. Este es un ejemplo del dodecaedro que mostraba Hamilton.



Definición

Un camino hamiltoniano en un grafo es un recorrido que visita cada vértice exactamente una vez. Es decir, un camino que recorre todos los vértices del grafo sin repetir ninguno. Por otro lado, un circuito hamiltoniano en un grafo es un recorrido que visita cada vértice exactamente una vez y termina en el mismo vértice donde empezó. Es decir, es un camino hamiltoniano cerrado.

¿Hay alguna forma sencilla de determinar si un grafo contiene o no un camino o un circuito hamiltoniano?

A primera vista, parece que debería haber alguna manera sencilla de hacerlo, ya que hay una manera sencilla de responder a la pregunta similar de si un grafo contiene o no un circuito euleriano, Sorprendentemente, no se conocen condiciones necesarias y suficientes sencillas para la existencia de circuitos hamiltonianos. No obstante, se conocen muchos teoremas que dan condiciones suficientes para la existencia de circuitos hamiltonianos. También hay ciertas propiedades que se pueden utilizar para demostrar que un grafo no contiene ningún circuito hamiltoniano. Por ejemplo, un grafo con un vértice de grado uno no puede contener un circuito hamiltoniano, ya que en un circuito hamiltoniano cada vértice es incidente con dos aristas del circuito.

Además, si un vértice del grafo tiene grado dos, entonces las dos aristas que son incidentes con ese vértice tienen que formar parte de cualquier circuito hamiltoniano. También observamos que cuando se construye un circuito hamiltoniano y este circuito ha pasado por un vértice, pueden descartarse todas las aristas incidentes con ese vértice que no sean las dos usadas en el circuito. Además, un circuito hamiltoniano no puede contener un circuito más pequeño dentro de él.

Teorema de DIRAC

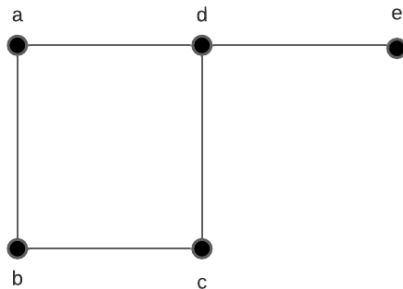
Sea G un grafo simple con n vértices para $n \geq 3$ tal que todos los vértices de G tienen grado mayor o igual que $n / 2$. Entonces, G contiene un circuito hamiltoniano.

Teorema de ORE

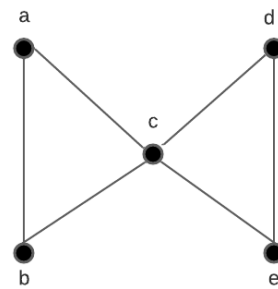
Sea G un grafo simple con n vértices para $n \geq 3$ tal que $\delta(u) + \delta(v) \geq n$, para cada par de vértices no adyacentes u y v de G . Entonces, G contiene un circuito hamiltoniano.

Ejemplo:

Demuestra que ninguno de los grafos que se muestran en la Figura contiene un circuito hamiltoniano.



G



H

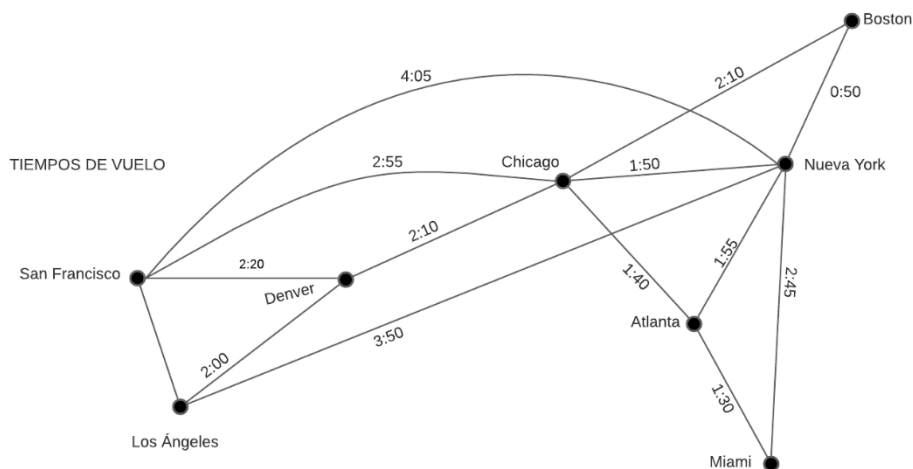
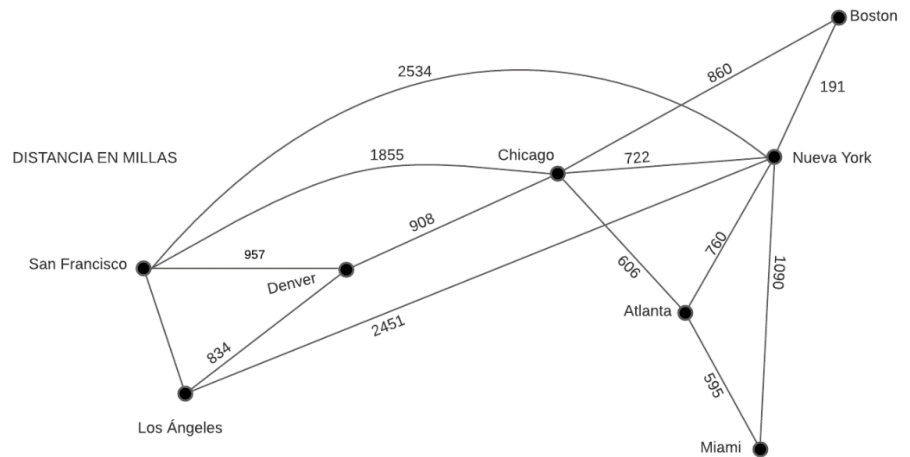
Solución: Solución: No hay ningún circuito hamiltoniano en G , ya que G tiene un vértice de grado uno, que es e . En cuanto a H , como los vértices a , b , d y e son todos de grado dos, cualquier arista incidente con estos vértices tiene que formar parte de cualquier circuito hamiltoniano. Ahora es fácil ver que no puede existir ningún circuito hamiltoniano en H , ya que cualquier circuito hamiltoniano tendría que contener cuatro aristas incidentes con e , lo que es imposible.

CAMINOS DE LONGITUD MÍNIMA

Introducción:

Muchos problemas se pueden representar utilizando grafos en los que se asigna un peso a cada una de las aristas. Consideremos, a modo de ilustración, la forma en que se representa el sistema de vuelos de una línea aérea.

Llamamos grafos ponderados a los grafos en los que se asigna un número a cada una de las aristas. Los grafos ponderados se utilizan para representar redes informáticas y pueden emplearse para estudiar los costes de comunicación (como, por ejemplo, el coste mensual de alquilar una línea telefónica), los tiempos de respuesta de los ordenadores o la distancia entre ordenadores. En la figura se muestra tres formas de asignar pesos a las aristas del grafo de una red informática que corresponden a distancias, tiempos de respuesta y costes.



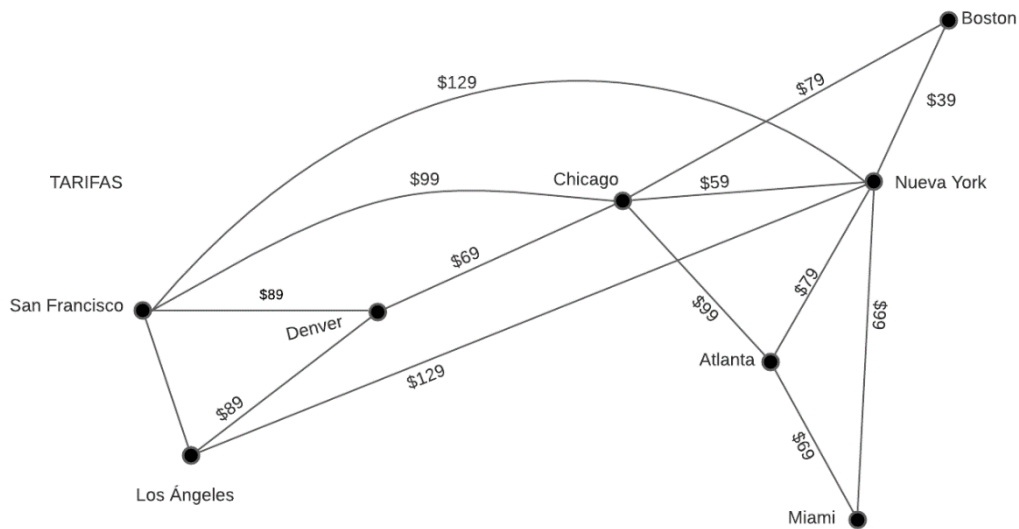
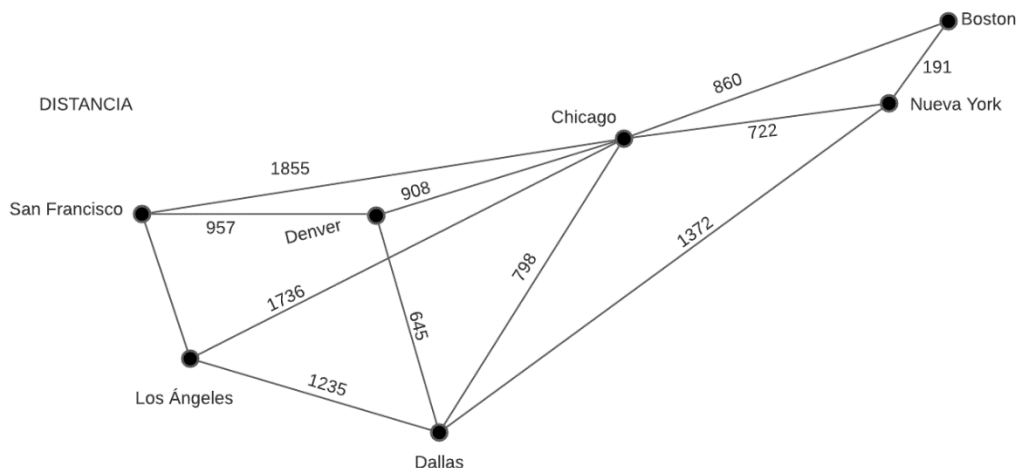
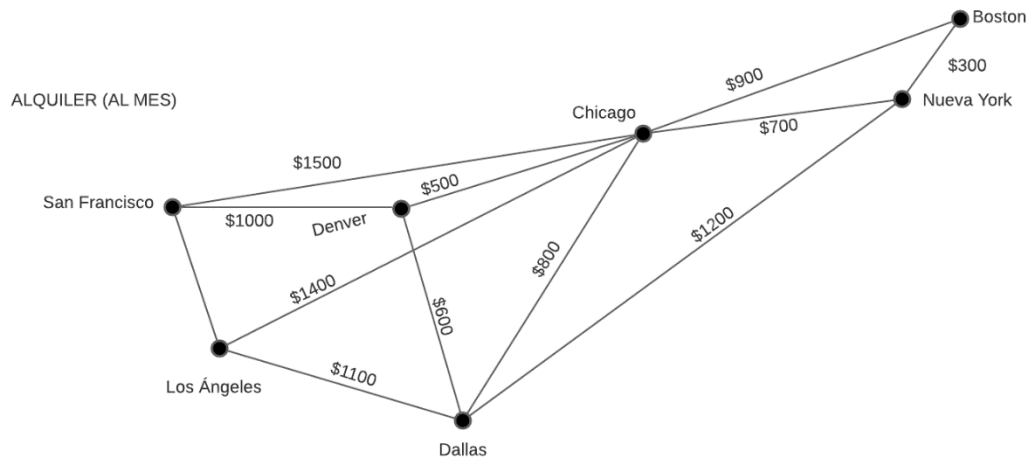
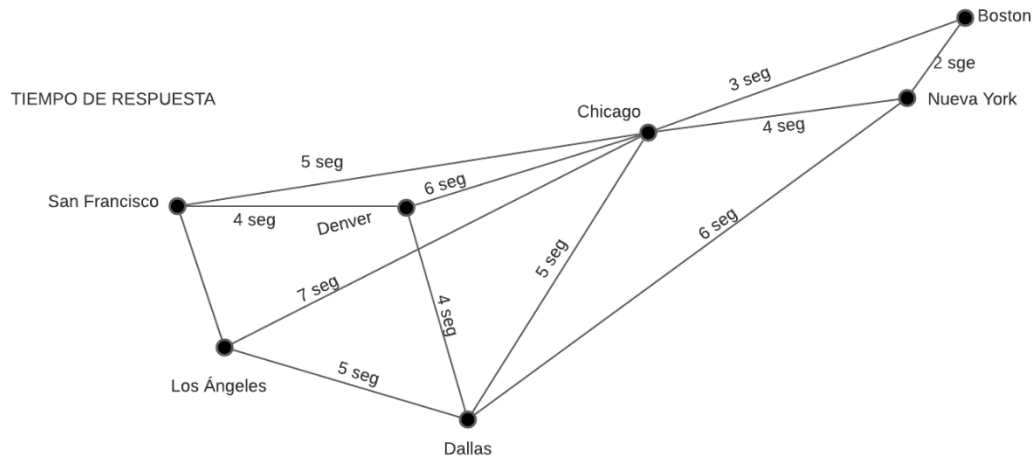


Figura 1, ¿Cuál es el camino más corto en distancia aérea entre Boston y Los Ángeles? ¿Qué combinación de vuelos tiene el menor tiempo total de vuelo (esto es, el tiempo total en el aire sin contar el tiempo entre vuelo y vuelo) entre Boston y Los Angeles? ¿Cuál es la tarifa más barata entre esas dos ciudades?

En la red informática de la Figura 2. ¿Cuál es el conjunto más barato de líneas telefónicas entre todos los que conectan los ordenadores de San Francisco con los de Nueva York?

¿Qué conjunto de líneas telefónicas proporciona una respuesta más rápida al comunicarse entre San Francisco y Nueva York? ¿Qué conjunto de líneas cubre menor distancia en total?





ALGORITMO PARA CALCULAR CAMINOS DE LONGITUD MÍNIMA

Hay diferentes algoritmos para hallar un camino de longitud mínima entre dos vértices de un grafo ponderado. Presentaremos un algoritmo descubierto por el matemático holandés Edsger Dijkstra en 1959. La versión que a describir resuelve este problema para grafos ponderados no dirigidos si todos los pesos son positivos. Este algoritmo puede adaptarse fácilmente para resolver problemas de caminos de longitud mínima en grafos dirigidos.

Antes de presentar el algoritmo formalmente, veremos un ejemplo a modo de motivación.

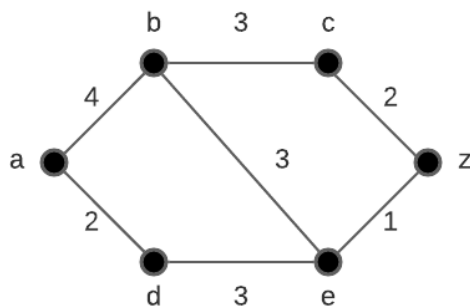


Figura: Un grafo ponderado simple.

Solución: Aunque puede hallarse fácilmente el camino más corto por simple inspección, se desarrollarán algunas ideas que serán útiles para entender el algoritmo de Dijkstra. Resolveremos este problema hallando la longitud de un camino de longitud mínima de a a cada uno de los vértices sucesivos hasta que lleguemos a z .

Los únicos caminos que comienzan en a y no contienen más vértices que a antes de alcanzar el vértice final son a, b y a, d . Como las longitudes de a, b y a, d son 4 y 2, respectivamente, se sigue que d es el vértice más cercano a a .

Se puede hallar el siguiente vértice más cercano examinando todos los caminos que pasan sólo por a y por d antes de alcanzar el vértice final. De entre éstos, el camino más corto a b sigue siendo a, b , con longitud 4, y el camino más corto a e es a, d, e con longitud 5. Por tanto, el siguiente vértice más cercano a a es b .

Para hallar el tercer vértice más próximo a a , se tiene que examinar únicamente aquellos caminos que pasan sólo por a, d y b antes de alcanzar el vértice final. Hay un camino de longitud 7 a c , que es a, b, c y un camino de longitud 6 a z , que es a, d, e, z . Por tanto, z es el siguiente vértice más cercano a a , y la longitud de un camino de longitud mínima a z es 6.

Este ejemplo ilustra los principios generales que se utilizan en el algoritmo de Dijkstra. Nótese que, de pura inspección se puede hallar el camino más corto de a a z . Sin embargo, no es eficiente para los humanos ni ordenadores si el grafo tiene muchas aristas.

Consideremos el problema general de determinar la longitud de un camino de longitud mínima entre a y z en un grafo simple no dirigido. Este algoritmo procede determinando la longitud de un camino de longitud mínima entre a y un segundo vértice, y así sucesivamente, hasta determinar la longitud mínima entre a y z .

ALGORITMO

Veamos ahora con detalle el algoritmo de Dijkstra. Se comienza asignándole ∞ a a la etiqueta 0 y a los demás vértices la etiqueta ∞ . Se utiliza la notación $L_0(a) = 0$ y $L_0(v) = \infty$ para estas etiquetas debe de haber llevado a cabo ninguna iteración (el subíndice 0 indica la iteración “0-ésima”). Estas etiquetas son las longitudes de caminos más cortos entre a y cada uno de los vértices, tomando el mínimo entre aquellos caminos que sólo contienen al vértice a (debido a que aún no existe ningún camino entre a y un vértice distinto de a , la longitud del camino más corto entre a y cada vértice distinto de él es ∞).

EL algoritmo de Dijkstra prosigue formando un conjunto distinguido de vértices. Sea S_k ese conjunto al cabo de k iteraciones del proceso de etiquetado. Comenzamos con $S_0 = \emptyset$.

El conjunto S_k se forma a partir de S_{k-1} . Una vez añadiendo un vértice u cuya etiqueta mínima entre los vértices que no están en S_{k-1} . Una vez añadiendo u a S_k , actualizamos las etiquetas de todos los vértices que no están en S_k , de modo que $L_k(v)$, la etiqueta del vértice v en el paso k , es la longitud de un camino de longitud mínima entre a y v que sólo contiene vértices de S_k (esto es, los vértices que ya estaban en el conjunto distinguido junto con u).

Sea v un vértice que no está en S_k . Para actualizar la etiqueta de v , observamos que $L_k(v)$, es la longitud del camino más corto entre a y v que sólo pasa por los vértices de S_k . La actualización se lleva a cabo de forma eficiente si se hace uso de la siguiente observación: el camino más corto de a y v que sólo pasa por vértices de S_k es bien el camino más corto entre a y v sólo pasa por vértices S_{k-1} (esto es, los vértices distinguidos exceptuando u) o bien es el resultado de añadirle a la arista (u, v) , al camino más corto entre a y u obteniendo en el paso $k - 1$. En otras palabras:

$$L_k(a, v) = \min\{L_{k-1}(a, v), L_{k-1}(a, u) + w(u, v)\}.$$

Este proceso se repite añadiendo vértices sucesivamente al conjunto distinguido hasta que se añade el vértice z . Cuando se añade z al conjunto distinguido, su etiqueta es la longitud del camino más corto entre a y z . Describimos el algoritmo de Dijkstra en el Algoritmo 1. Más adelante demostraremos que este algoritmo es correcto.

ALGORITMO 1
Algoritmo de Dijkstra

Procedure Dijkstra (G : grafo ponderado simple y conexo, con todos los pesos positivos)

{ G tiene vértices $a = v_0, v_1, \dots, v_n = z$ y pesos $w(v_1, v_j)$,

Donde $w(v_1, v_j) = \infty$ si $\{v_1, v_j\}$ no es una arista de G }

for $i := 1$ **to** n

$L(v_1) := \infty$

$L(a) := 0$

$S := \emptyset$

{los valores iniciales de las etiquetas se asignan de modo que la etiqueta de a es 0 y todas las demás etiquetas son ∞ y S es el conjunto vacío}

While $z \notin S$

Begin

$u :=$ vértice con $L(u)$ mínima entre los vértices que no están en S

$S := S \cup \{u\}$

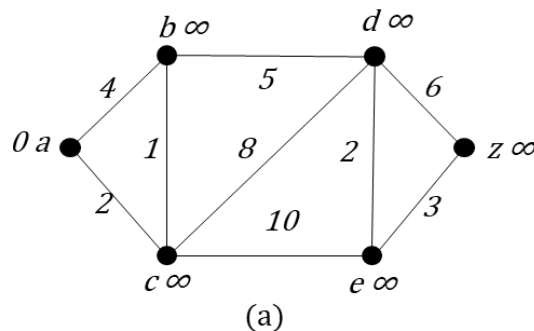
for todos los vértices v que no están en S

if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$

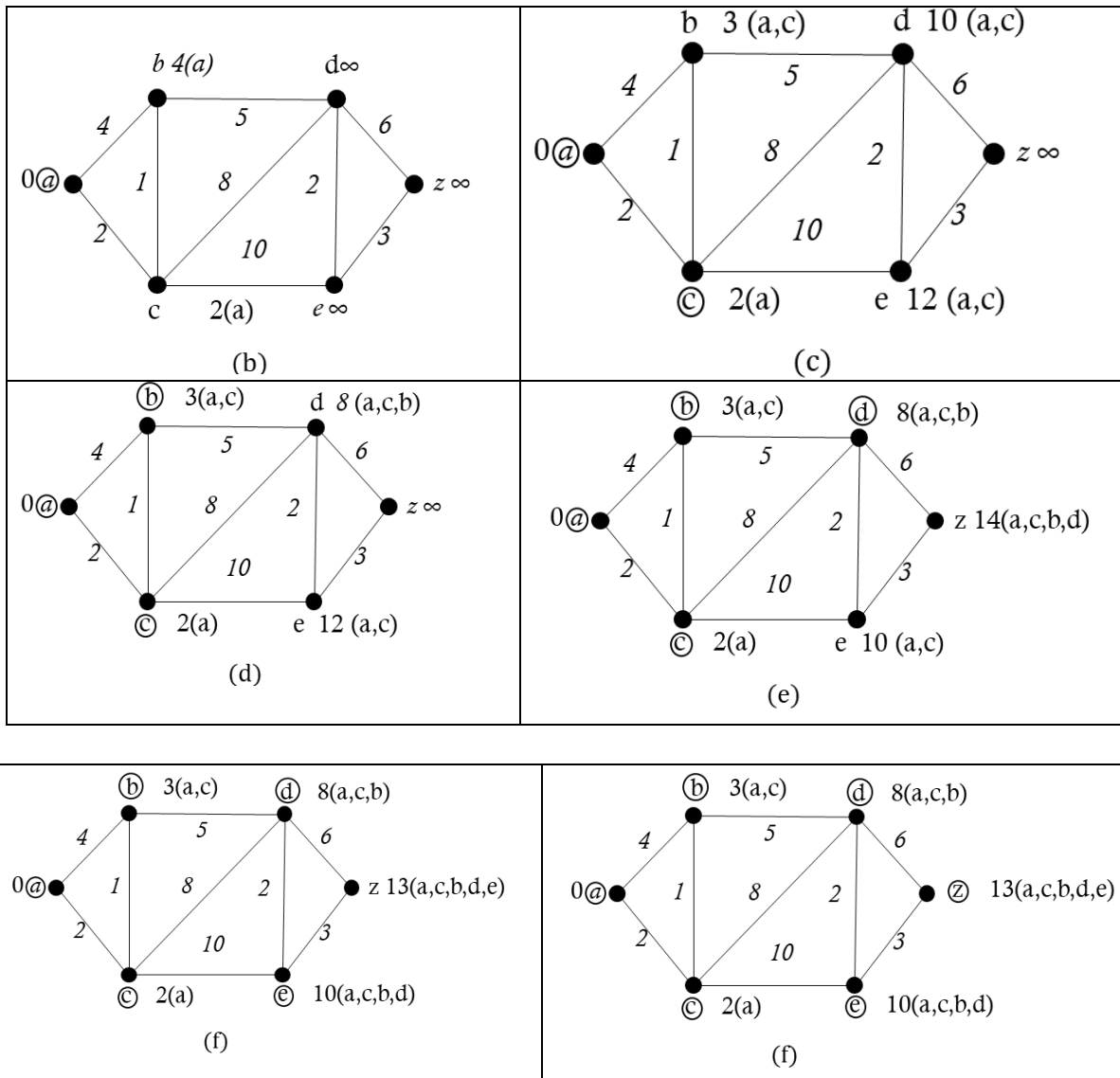
{esto añade a S en un vértice con etiqueta mínima y actualiza las etiquetas de los vértices que no están en S }

End { $L(z)$ = longitud del camino más corto entre a y z }

EJEMPLO: Utiliza el algoritmo de Dijkstra para hallar el camino de longitud mínima entre los vértices a y z del grafo ponderado de la siguiente figura:



Solución: En la figura se muestran los pasos que sigue el algoritmo de Dijkstra para determinar un camino de longitud mínima entre a y z . En cada iteración se indica un camino de longitud mínima entre a y cada uno de los vértices que sólo contiene vértices de S_k . El algoritmo termina cuando se encierra en un círculo a z . El camino de longitud mínima de a a z que se obtiene es a, b, c, d, e, z de encierra en un círculo a z . El camino de longitud 13.



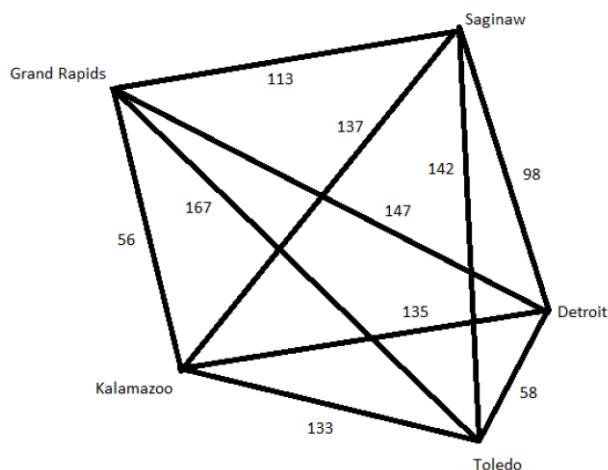
TEOREMA 1: *El algoritmo de Dijkstra determina la longitud del camino más corto entre dos vértices de un grafo simple, conexo y no dirigido.*

TEOERMA 2: *El algoritmo de Dijkstra realiza $O(n^2)$ operaciones (sumas y comparaciones) para determinar la longitud del camino más corto entre dos vértices de un grafo ponderado simple, conexo y no dirigido con n vértices.*

EL PROBLEMA DEL VIAJANTE

Un viajante quiere visitar exactamente una vez cada ciudad de un conjunto de n ciudades para finalmente regresar al punto de partida. Por ejemplo, supongamos que el viajante quiere visitar Detroit, Toledo, Saginaw, Grand Rapids y Kalamazoo. ¿En qué orden debería visitar estas ciudades para recorrer la mínima distancia total? Para resolver el problema suponemos que el viajante comienza por Detroit y examinamos todas las posibilidades formas en que puede visitar las otras cuatro ciudades y regresar a Detroit. Al realizar las combinaciones se pueden sacar 24 respuestas diferentes, pero como es el mismo recorrido devuelta, solo consideramos 12 para determinar la mínima distancia total que el viajante debe recorrer. Se enumeran los 12 circuitos y la distancia que se recorre en cada uno.

Ruta	Distancia Total (En millas)
Detroit-Toledo-Grand Rapids-Saginaw-Kalamazoo-Detroit	610
Detroit-Toledo-Grand Rapids-Kalamazoo-Saginaw-Detroit	516
Detroit-Toledo-Kalamazoo-Saginaw-Grand Rapids-Detroit	588
Detroit-Toledo-Kalamazoo-Grand Rapids-Saginaw-Detroit	458
Detroit-Toledo-Saginaw-Kalamazoo-Grand Rapids-Detroit	540
Detroit-Toledo-Saginaw-Grand Rapids-Kalamazoo-Detroit	504
Detroit-Saginaw-Toledo-Grand Rapids-Kalamazoo-Detroit	598
Detroit-Saginaw-Toledo-Kalamazoo-Grand Rapids-Detroit	576
Detroit-Saginaw-Kalamazoo-Toledo-Grand Rapids-Detroit	682
Detroit-Saginaw-Grand Rapids-Toledo-Kalamazoo-Detroit	646
Detroit-Grand Rapids-Saginaw-Toledo-Kalamazoo-Detroit	670
Detroit-Grand Rapids-Toledo-Saginaw-Kalamazoo-Detroit	728



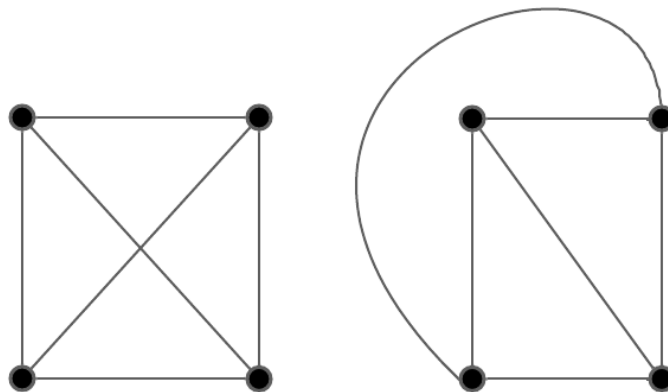
El problema del viajante consiste en encontrar el circuito de peso total mínimo de un grafo ponderado, completo y no dirigido que visita cada vértice exactamente una vez y regresa al punto de partida. Para resolver un caso particular de este problema, se pueden examinar todos los posibles circuitos hamiltonianos y elegir uno de longitud total mínima. Sin embargo, el número de circuitos hamiltonianos distintos crece muy rápidamente y tratar de resolver el problema exhaustivamente para grafos con muchas aristas es casi imposible. No se conoce ningún algoritmo de complejidad polinómica en el peor caso que resuelva el problema del viajante, pero existen algoritmos de aproximación que garantizan una solución cercana a la solución exacta. Si se encontrara un algoritmo que proporcionara siempre una solución cuyo peso total fuera menor o igual que k veces el peso total de una solución óptima, se demostraría que la clase P coincide con la clase NP , lo que es uno de los problemas abiertos más famosos de la teoría de complejidad de algoritmos.

GRAFOS PLANOS

Definición: Se dice que un grafo es *plano* si puede dibujarse en el plano de manera que ningún par de sus aristas se corte (por corte de aristas se entiende la intersección de las líneas que representan a las aristas en un punto distinto de sus extremos). A ese dibujo se le llama representación plana en del grafo.

Observación: Un grafo puede ser *plano*, aunque habitualmente se dibuje con cortes, ya que cabe la posibilidad de que se pueda dibujar diferente sin ningún corte, en pocas palabras, representarlo de manera equivalente.

EJEMPLO: El siguiente grafo simple se representa en forma de grafo plano.



k_4 con cortes

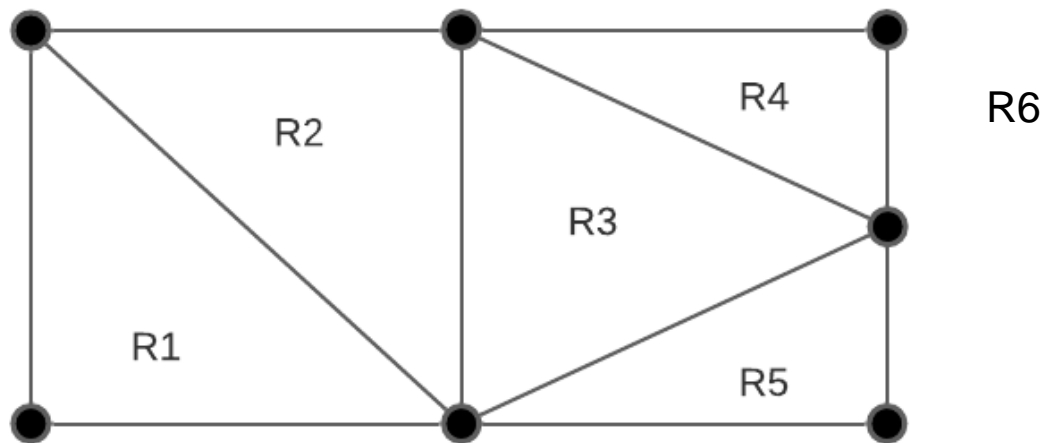
k_4 sin cortes

LA FÓRMULA DE EULER

Una representación plana de un grafo divide el plano en **regiones**, entre ellas una región no acotada. Por ejemplo, la representación plana que se muestra en la siguiente figura divide el plano en seis regiones, que están etiquetadas en la figura. Euler demostró que todas las representaciones planas de un mismo grafo dividen el plano en igual número de regiones. Esto lo hizo hallando una relación entre el número de regiones, el número de vértices y el número de aristas de un grafo plano.

FÓRMULA DE EULER: Sea G un grafo simple conexo con e aristas y v vértices. Sea r el número de regiones de una representación planas de G . Entonces, $r = |e| - |v| + 2$.

Ejemplo: En el siguiente grafo plano tenemos en cuenta: $e = 11$, $v = 7$, lo cual nos quedan $r = 11 - 7 + 2$, $r = 6$.



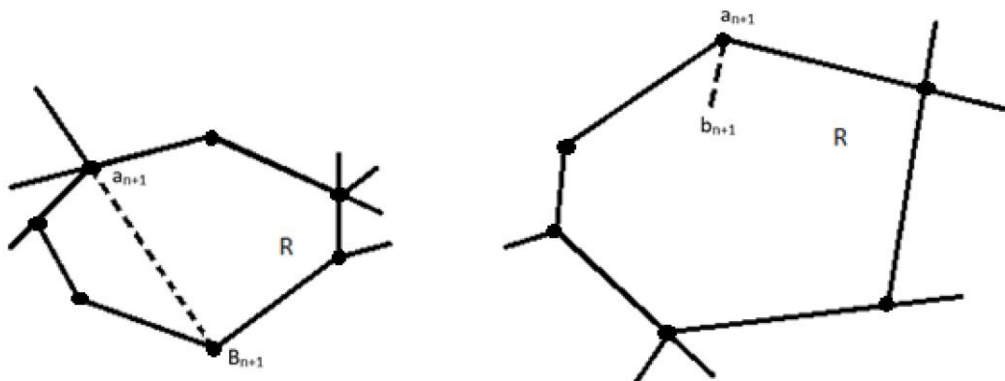
Ejemplo: Supongamos que un grafo simple y conexo tiene 20 vértices, cada uno de los cuales tiene grado 3. ¿En cuántas regiones divide el plano una representación plana de ese grafo?

Solución: El grafo tiene 20 vértices, cada uno de grado 3, de modo que: $v = 20$. Como la suma de los grados de los vértices, $3v = 3(20) = 60$, es igual al doble del número de aristas, se tiene que $2e = 60$, o que $e = 30$. Por tanto, según la fórmula de Euler, el número de regiones es: $r = 30 - 20 + 2$, $r = 12$.

Observación: La fórmula de Euler se puede utilizar para establecer desigualdades que tiene que cumplir cualquier grafo plano. Una de estas desigualdades es la que se da en el siguiente corolario.

Corolario 1

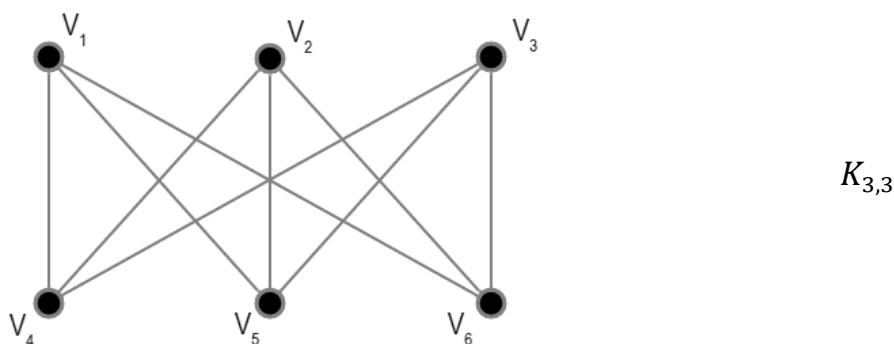
Sea G un grafo plano simple y conexo con e aristas, v vértices y $v \geq 3$. Entonces, $e \leq 3v - 6$.



Corolario 2: Sea G un grafo simple y conexo. Entonces G tiene un vértice de grado menor o igual que cinco.

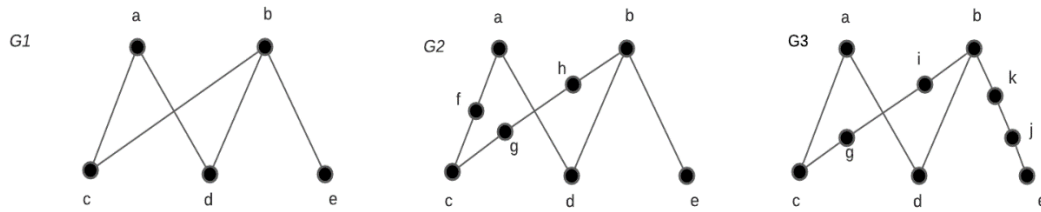
Sea G un grafo plano simple y conexo con e aristas, v vértices y $v \geq 3$, entonces $e \leq 3v - 6$.

Corolario 3: Si un grafo plano simple y conexo tiene e aristas y v vértices con $v \geq 3$ y no contiene circuitos de longitud tres, entonces $e \leq 2v - 4$.



TEOREMA DE KURATOWSKI

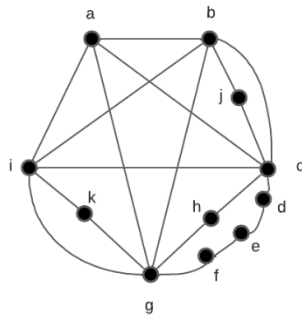
Hemos visto que $K_{3,3}$ y K_5 no son planos. Claramente, cualquier grafo que contenga como subgrafo a uno de esos dos grafos tampoco será plano. Sin embargo, todos los grafos que no son planos tienen que contener un subgrafo que se puede obtener a partir de $K_{3,3}$ o de K_5 por medio de ciertas operaciones.



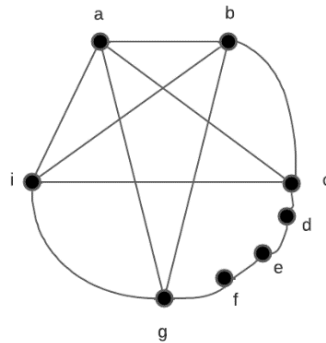
Observación: en G_1 hay una arista $\{c, b\}$, en cambio en el grafo G_2 hay tres subdivisiones elementales, si solo analizamos la subdivisión elemental g , podemos identificar que se eliminó la arista del principio y se añadió el vértice g y las aristas $\{c, g\}$ y $\{g, b\}$.

Si un grafo es plano, también lo será cualquier grafo que se obtenga de él eliminando una arista $\{u, w\}$ y añadiendo un nuevo vértice w junto con las aristas $\{u, w\}$ y $\{w, v\}$. Se dice que esta operación es una **subdivisión elemental**. Ahora, Se dice que los grafos $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ son **homeomorfos** si se pueden obtener a partir de un mismo grafo por medio de una secuencia de subdivisiones elementales.

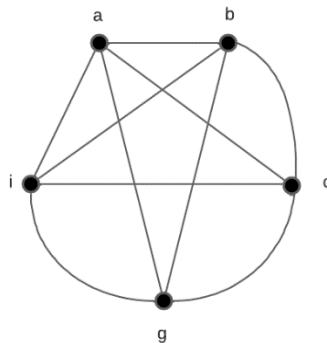
Ejemplo: Determinar si el grafo G de que se muestra en la siguiente figura es o no un grafo plano.



G



H



K_5

Solución: G tiene un subgrafo H homeomorfo a K_5 , que se obtiene eliminando h, j y k junto con todas las aristas incidentes con dichos vértices. El grafo H es homeomorfo a K_5 , ya que puede obtenerse a partir de K_5 (con vértices a, b, c, g, i) por medio de una secuencia de subdivisiones elementales, añadiendo a los vértices $d, e, y f$. Por tanto, G no es plano.

COLOREADO DE GRAFOS

Definición 1

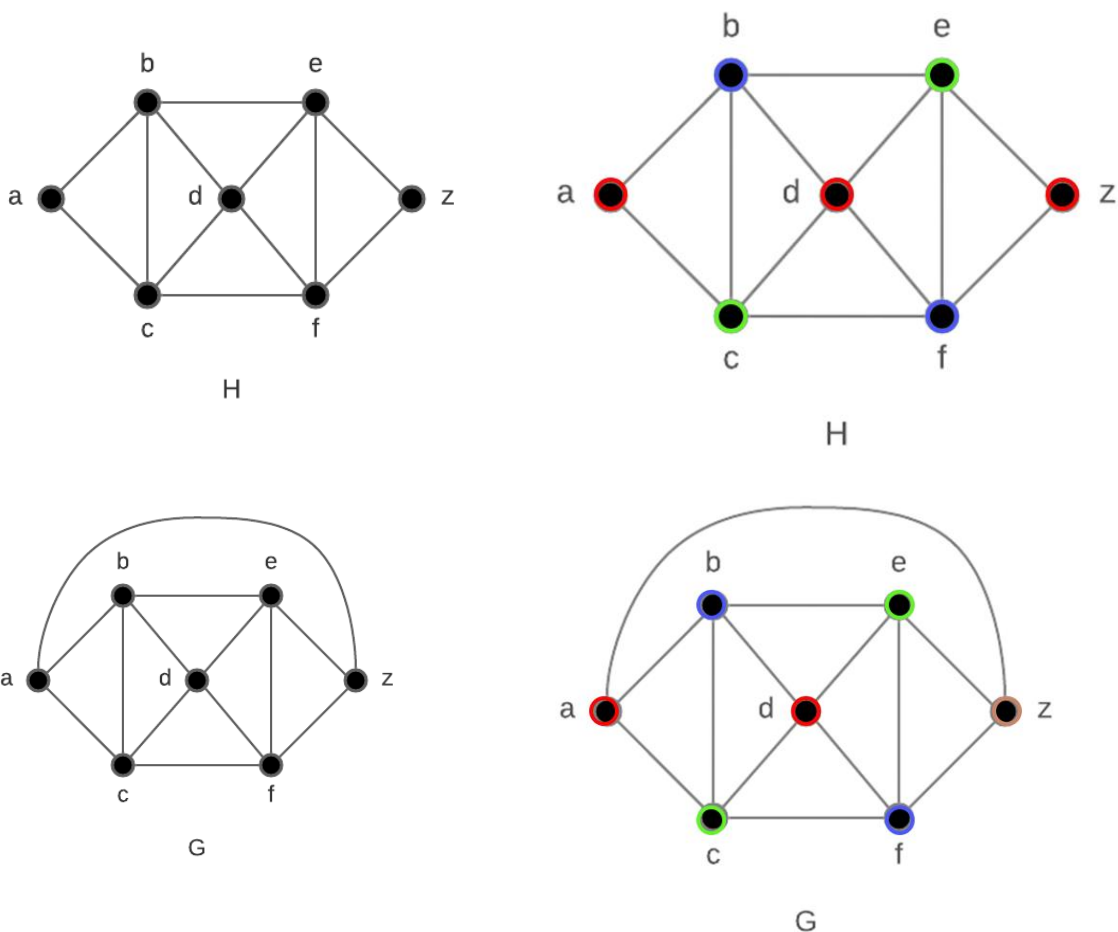
Una coloración de un grafo simple consiste en asignarle un color a cada vértice del grafo de manera que a cada dos vértices adyacentes se les asignan colores distintos.

Definición 2

El número cromático de un grafo es el número mínimo de colores que se requieren para una coloración del grafo.

Teorema 1: El teorema de los 4 colores, El número cromático de un grafo plano menor o igual a cuatro.

Ejemplos:



APLICACIONES DEL COLOREADO DE GRAFOS

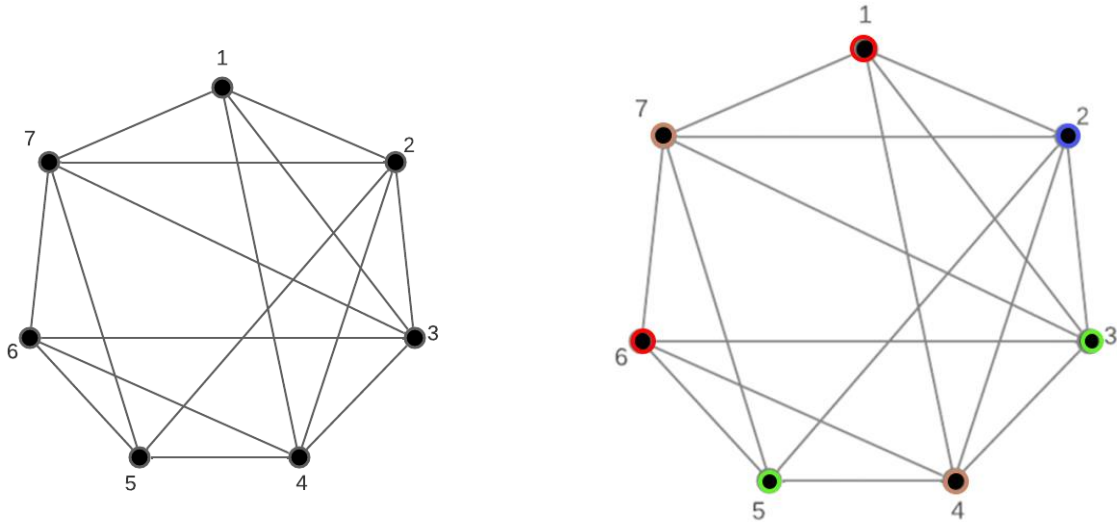
El coloreado de grafos tiene múltiples aplicaciones en diversos campos, pero estos tienen un enfoque que tiende a la planificación y asignación, sin embargo, al no existir algoritmos eficientes para el coloreado de grafos, no se usan mucho en sistemas de computación.

Programación de exámenes finales: ¿Cómo se pueden programar exámenes finales de una universidad de modo que ningún estudiante tenga 2 exámenes al mismo tiempo?

Solución: Usando un modelo con grafos donde los vértices representan las asignaturas y hay una arista entre 2 vértices si hay un estudiante matriculado en las asignaturas representadas por dichos vértices. Cada segmento horario reservado para un examen final se representa mediante un color diferente.

Se supone que, se quieren programar siete exámenes finales. Supongamos que las asignaturas se numeran del 1 al 7 y que las siguientes parejas de alumnos en común: 1 y 2, 1 y 3, 1 y 4, 1 y 7, 2 y 3, 2 y 4, 2 y 5, 2 y 7, 3 y 4, 3 y 6, 3 y 7, 4 y 5, 4 y 6, 5 y 6, 5 y 7, y 6 y 7. En la siguiente figura se muestra el grafo asociado a este conjunto de asignaturas, Una programación consiste en la coloración de este grafo.

Como el número cromático de este grafo es 4, se necesitan cuatro segmentos horarios.



UNIDAD 4

ÁRBOLES

Introducción

Un grafo conexo que no contiene ciclos se llama árbol. Los árboles comenzaron a emplearse en 1857, cuando el matemático inglés Arthur Cayley los utilizó para contar cierto tipo de componentes químicos. Desde ese momento, los árboles se han empleado para resolver problemas en una gran variedad de disciplinas, como veremos en los ejemplos de este capítulo. Los árboles son particularmente útiles en informática, pues son empleados en un amplio espectro de algoritmos. Por ejemplo, los árboles se usan para construir algoritmos eficientes que localizan elementos en una lista. También pueden utilizarse en algoritmos, como los códigos de Huffman, que construyen códigos compresores eficientes, ahorrando costes en la transmisión de datos. y en su posterior almacenamiento. Los árboles se pueden emplear para estudiar juegos como las damas y el ajedrez y pueden ayudar a determinar estrategias ganadoras para cada uno de estos juegos. Los árboles pueden utilizarse para modelar procedimientos que se llevan a cabo mediante una secuencia de decisiones. Construir estos modelos puede ayudar a determinar la complejidad computacional de los algoritmos basados en una secuencia de instrucciones como los algoritmos de ordenación.

Los procedimientos para construir árboles que contengan todos los vértices de un grafo, incluyendo la búsqueda en profundidad y la búsqueda con anchura, pueden usarse para explorar sistemáticamente los vértices de un grafo. Recorrer los vértices de un grafo haciendo una búsqueda en profundidad, también conocida como de vuelta atrás", es la base de la búsqueda sistemática de soluciones en una amplia variedad de problemas, tales como determinar el modo de colocar ocho reinas en un tablero de ajedrez de modo que dos cualesquiera no se amenacen.

Podemos modelar muchos problemas asignando pesos a las aristas de un árbol. Por ejemplo, utilizando árboles ponderados podemos desarrollar algoritmos para construir redes que contengan el conjunto menos costoso de líneas de teléfono que conectan distintos nodos de la red.

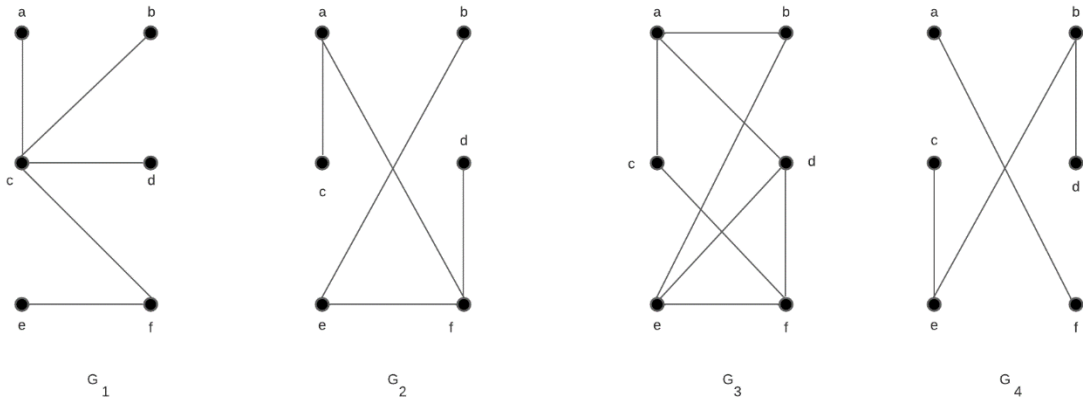
Definición 1

Un árbol es un grafo no dirigido, conexo y sin ciclos.

Puesto que un árbol no puede contener ciclos, es un grafo acíclico, tampoco puede tener bucles o aristas múltiples. Por tanto, un árbol es necesariamente un grafo simple.

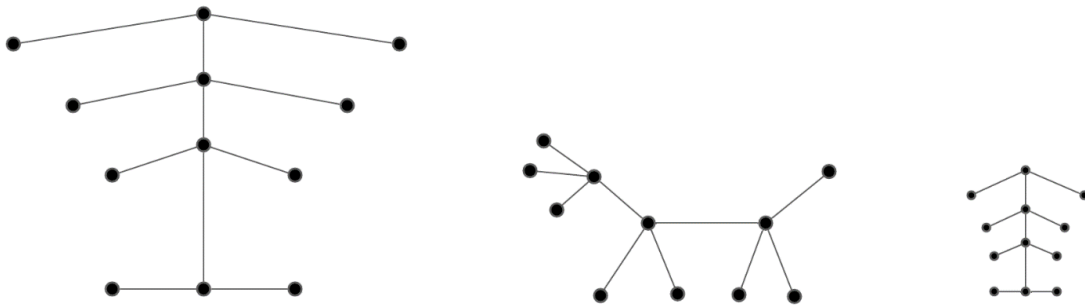
Ejemplo 1

¿Cuáles de los grafos de la figura son árboles?



Solución: G_1 y G_2 son árboles, puesto que ambos son grafos conexos y acíclicos. G_3 no es un árbol, porque e, b, a, d, e es un ciclo del grafo. Finalmente, G_4 no es un árbol, porque no es conexo.

Todo grafo conexo y acíclico es un árbol. ¿Qué podemos decir de los grafos acíclicos, pero no necesariamente conexos? Estos grafos se llaman bosques y tienen la propiedad de que cada una de sus componentes conexas es un árbol.



En muchas ocasiones, los árboles se definen como grafos no dirigidos con la propiedad de cada pareja de vértices está conectada por un único camino. El siguiente teorema prueba que esta definición alternativa es equivalente a la nuestra.

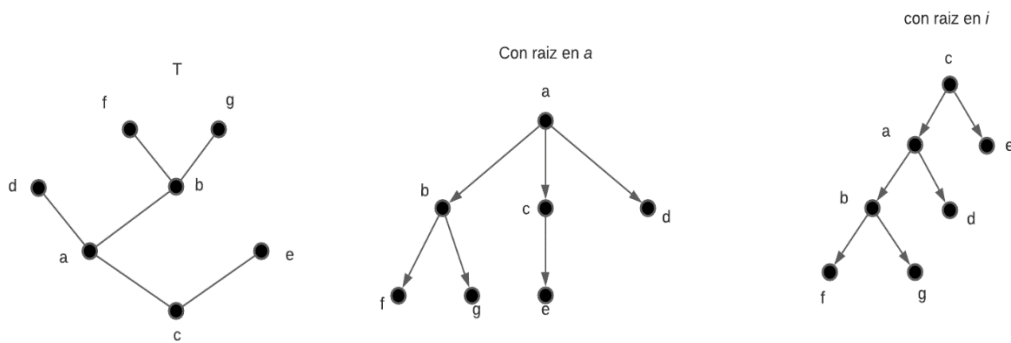
Teorema 1

Un grafo no dirigido es un árbol si, y solo si, hay un único camino entre cada pareja de vértices.

Definición 2

Un árbol con raíz es un árbol en el que uno de sus vértices ha sido designado como la raíz y todas están orientadas de modo que se alejan de la raíz. (Cada nodo tiene un solo padre, excepto el nodo raíz que no tiene padre).

Los árboles con raíz también se pueden definir recursivamente (véase la Sección 3.4). Podemos transformar un árbol en un árbol con raíz sin más que elegir un vértice como raíz. Nótese que las distintas elecciones de la raíz producen diferentes árboles con raíz. Por ejemplo, la Figura 4 muestra los árboles con raíz formados al designar como raíces los vértices *a* y *c*, respectivamente, en el árbol *T*. Normalmente, dibujaremos un árbol con raíz situando la raíz en la parte superior del grafo. Las flechas que indican el sentido de las aristas pueden omitirse en un árbol con raíz, puesto que la elección de la raíz determina las direcciones de las aristas.



La terminología de los árboles tiene orígenes botánicos y genealógicos. Supongamos que T es un árbol con raíz. Si v es un vértice de T distinto de la raíz, el padre de v es el único vértice u tal que hay una arista dirigida de u a v (el lector debería demostrar que tal vértice es único). Cuando u es el padre de v , se dice que v es hijo de u . Los vértices con el mismo padre se llaman hermanos. Los antecesores de un vértice diferente de la raíz son todos los vértices que aparecen en el camino desde la raíz hasta ese vértice, excluyendo a este último e incluyendo a la raíz.

Esto es, su padre, el padre de su padre y así hasta alcanzar la raíz. Los descendientes de un vértice v son aquellos vértices para los que v es un antecesor. Un vértice de un árbol se llama hoja si no tiene hijos.

Los vértices que tiene hijo se llaman vértices internos. La raíz es un vértice interno sino es el único vértice del grafo. En este caso, será una hoja.

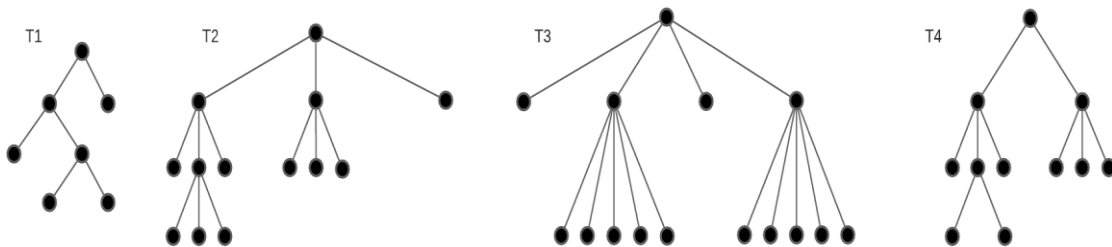
Si a es un vértice de un árbol, el subárbol con raíz en a es el subgrafo del árbol que contiene el vértice a , a todos sus descendientes y a todas las aristas incidentes en dichos descendientes.

Definición 3

Un árbol con raíz se llama árbol m -ario si todos los vértices internos tienen, a lo sumo, m hijos. El árbol se llama árbol m -ario completo si todo vértice interno tiene exactamente m hijos, un árbol m -ario con $m = 2$ se llama árbol binario.

Ejemplo 3

Los árboles con raíz de la figura, ¿Son árboles m -arios completos para algún entero positivo m ?



Solución: T_1 , es un árbol binario completo, pues cada uno de sus vértices internos tiene dos hijos.

T_2 es un árbol ternario completo, pues sus vértices internos tienen tres hijos. En T_3 , cada vértice interno tiene cinco hijos, de modo que T_3 es un árbol 5-ario completo. T_4 , no es un árbol completo para ningún m , pues algunos nodos internos tienen dos hijos, mientras que otros tienen tres.

Un árbol ordenado con raíz es un árbol con raíz en el que los hijos de cada vértice interno están ordenados. Los árboles ordenados con raíz se dibujan de modo que los hijos de cada nodo interno se colocan ordenados de izquierda a derecha. Nótese que una representación de un árbol con raíz en el modo convencional determina un orden para sus aristas. Utilizaremos estas ordenaciones de las aristas en los dibujos sin mencionar explícitamente que estamos considerando un árbol ordenado con raíz.

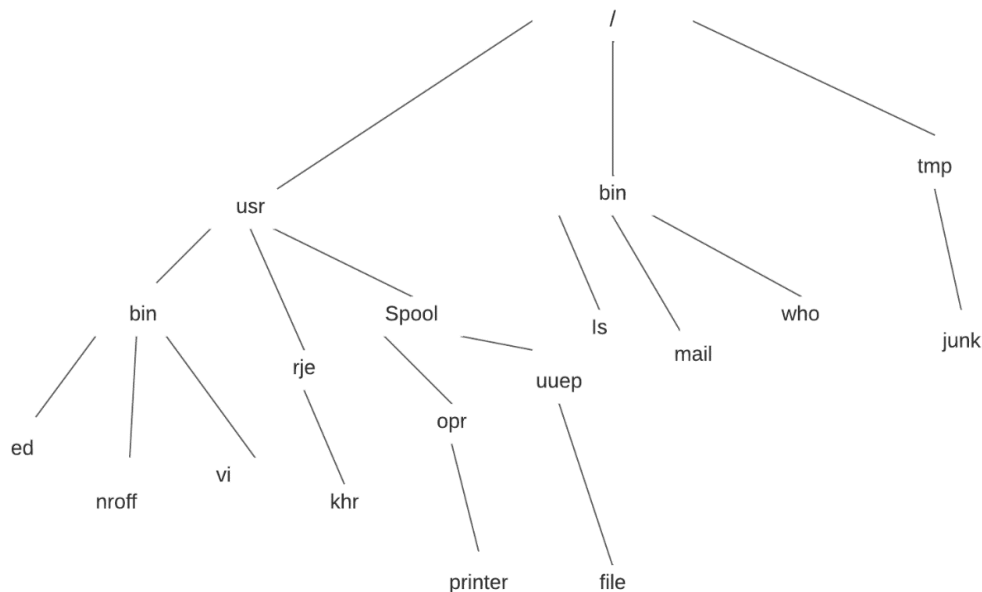
En un árbol binario ordenado (normalmente, llamado simplemente árbol binario), si cada vértice interno tiene dos hijos, el primer hijo se llama hijo izquierdo y el segundo se llama hijo derecho. El árbol con raíz en el hijo izquierdo de un vértice se llama subárbol izquierdo de este vértice y el árbol con raíz en el hijo derecho subárbol derecho.

ÁRBOLES COMO MODELOS

Los árboles se utilizan como modelos en ciencias tan diversas como la informática, la química, la geología, la botánica y la psicología. A continuación, describiremos varios modelos basados en árboles.

Sistema de ficheros de ordenadores

Sistemas de ficheros de ordenadores En la memoria de un ordenador, los ficheros se organizan en directorios. Un directorio puede contener tanto ficheros como subdirectorios. La raíz de un directorio contiene el sistema de ficheros completo. Así, un sistema de ficheros puede representarse mediante un árbol con raíz, donde la raíz representa el directorio raíz, los vértices internos representan los subdirectorios y las hojas representan los ficheros ordinarios o subdirectorios vacíos. Uno de tales sistemas de ficheros se muestra en la siguiente figura. En este sistema, el fichero *khr* está en el directorio *rje*. (Nótese que los enlaces con ficheros pueden dar lugar a circuitos si hay ficheros con más de una ruta de acceso).



PROPIEDADES DE LOS ÁRBOLES

Con frecuencia se necesitan resultados que relacionen los números de vértices de aristas en diferentes tipos de árboles.

TEOREMA 2: Un árbol de n vértices, tiene $n - 1$ aristas.

Demostración: Se demostrará usando principio de inducción. Nótese que para cualquier árbol podemos elegir una raíz y considerar el árbol con raíz resultante.

Paso de inducción: La hipótesis de inducción afirma que todo árbol de k vértices tiene $k - 1$ aristas, donde k es un entero positivo. Supongamos que un árbol T tiene $k + 1$ vértices y v es una hoja de T (que debe existir puesto que el árbol es finito) y sea w el padre de v . Si eliminamos T tanto el vértice v como la arista que conecta a w y v se obtiene un árbol T' tiene $k - 1$ aristas. De ahí se deduce que T tiene k aristas, ya que tiene una más que T' , la arista que conecta a v y w . Esto completa el paso de inducción.

El número de vértices de árbol m -ario completo con un número prefijado de vértices internos está determinado, como muestra el teorema siguiente. Como el Teorema 2, denotaremos por n al número de vértices de un árbol.

TEOREMA 3: Un árbol m -ario completo con i vértices internos tiene $n = mi + 1$ vértices.

TEOREMA 4: Un árbol m -ario completo con

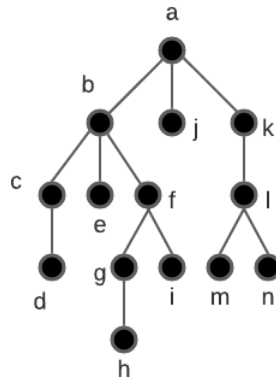
1. n vértices tiene $i = \frac{n-1}{m}$ vértices internos y $l = [(m-1)n + 1]/m$ hojas.
2. i vértices internos tiene $n = mi + 1$ vértices y $l = (m-1)i + 1$ hojas.
3. l hojas tiene $n = (ml - 1)/(m - 1)$ vértices e $i = (m-1)/(m-1)$ vértices internos.

EJEMPLO: Supongamos que alguien comienza una cadena de cartas. A cada persona que recibe una de esas cartas se le pide que la envíe a otras cuatro. Algunas personas lo hacen, pero otras no envían ninguna carta. ¿Cuántas personas han leído la carta, incluyendo a la primera persona, si nadie recibe más de una carta y si la cadena finaliza después de 100 personas que han visto la carta no hayan enviado ninguna? ¿Cuántas personas enviaron la carta?

Solución: La cadena de cartas se puede representar mediante un árbol 4-ario. Los vértices internos se corresponden con aquellas personas que enviaron la carta y las hojas con aquellas otras no la enviaron. Puesto que 100 personas no enviaron la carta, el número de hojas de este árbol con raíz es $l = 100$. De ahí, por el tercer apartado del Teorema 4, se tiene que el número de personas que han visto la carta es $n = \frac{4 \times 100 - 1}{4 - 1} = 133$. También se tiene que el número de vértices internos es $133 - 100 = 33$, así que 33 personas enviaron la carta.

Con frecuencia es preferible utilizar árboles con raíz que son “equilibrados” en el sentido de que los subárboles de cada vértice contienen caminos de aproximadamente la misma longitud. Posteriormente, se presentarán definiciones que clarificarán este concepto. El **nivel** de un vértice v en un árbol con raíz es la longitud del único camino desde la raíz a dicho vértice. El nivel de la raíz, por definición, es cero. La **altura** de un árbol con raíz es la longitud del camino más largo desde la raíz a cualquier vértice.

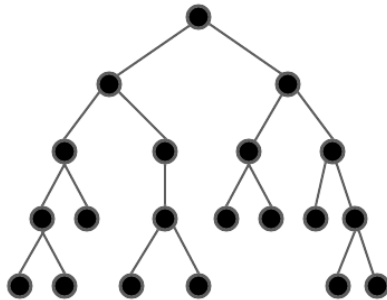
Ejemplo: Calcular el nivel de vértice en el árbol con raíz de la siguiente la siguiente figura. ¿Cuál es la altura de este árbol?



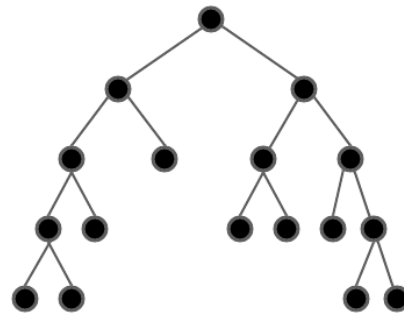
Solución: La raíz a está en el nivel 0. Los vértices b, j y k están en el nivel 1. Los vértices c, e, f y l están en el nivel 2. Los vértices d, g, i, m y n están en el nivel 3. Finalmente, el vértice h está en el nivel 4. Puesto que el mayor nivel es el 4, el árbol tiene altura 4.

Un árbol con raíz m -ario de altura h está **equilibrado** o **balanceado** si todas sus hojas están en los niveles h o $h - 1$.

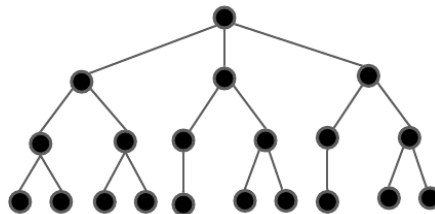
Ejemplo: ¿Cuáles de los árboles de la siguiente figura están equilibrados?



T_1



T_2



T_3

Solución: T_1 está equilibrado, pues todas sus hojas están en los niveles 3 y 4. Sin embargo, T_2 no es un árbol equilibrado, pues sus hojas están en los niveles 2, 3 y 4. Finalmente, T_3 está equilibrado, pues todas sus hojas están en el nivel 3.

TEOREMA 5: Un árbol m -ario de altura h tiene, a lo sumo, m^h hojas.

COROLARIO 1: Si un árbol m -ario de altura h tiene l hojas, entonces $h \geq \lceil \log_m l \rceil$. (Estamos utilizando la función *parte entera por exceso*. Recordamos $\lceil x \rceil$ es el menor entero mayor o igual que x).

APLICACIONES DE LOS ÁRBOLES

Introducción

Discutiremos tres problemas que pueden estudiarse utilizando árboles. El primer problema es: ¿Cómo se pueden almacenar elementos en una lista de manera que todo elemento pueda ser localizado fácilmente? El segundo problema es: ¿Qué serie de decisiones deberíamos tomar para encontrar un objeto con una determinada propiedad en una colección de objetos de un cierto tipo? El tercer problema es: ¿Cómo se puede codificar de manera eficiente un conjunto de caracteres mediante una cadena de bits?

Árboles Binarios de Búsqueda

Los árboles binarios de búsqueda son una estructura de datos comúnmente utilizada en la informática para almacenar y buscar elementos. En matemáticas discretas, los árboles binarios de búsqueda se pueden entender en términos de teoría de conjuntos y relaciones de orden.

Un árbol binario de búsqueda es una estructura de datos que consta de nodos organizados en una estructura jerárquica. Cada nodo tiene un valor asociado y puede tener hasta dos hijos: un hijo izquierdo y un hijo derecho. Los valores de los hijos izquierdo y derecho son menores y mayores que el valor del nodo padre, respectivamente.

Podemos pensar en los nodos de un árbol binario de búsqueda como elementos de un conjunto ordenado. El nodo raíz del árbol representa el elemento más grande del conjunto, mientras que los nodos hoja representan los elementos más pequeños. Cada nodo intermedio representa un elemento que se encuentra entre el valor del nodo padre y los valores de sus hijos.

Para buscar un elemento en un árbol binario de búsqueda, comenzamos en el nodo raíz y comparamos el valor del nodo con el valor buscado. Si el valor del nodo es igual al valor buscado, hemos encontrado el elemento. Si el valor del nodo es mayor que el valor buscado, buscamos en el hijo izquierdo del nodo. Si el valor del nodo es menor que el valor buscado, buscamos en el hijo derecho del nodo. Repetimos este proceso hasta que encontremos el elemento buscado o lleguemos a un nodo hoja, lo que significa que el elemento no está presente en el árbol.

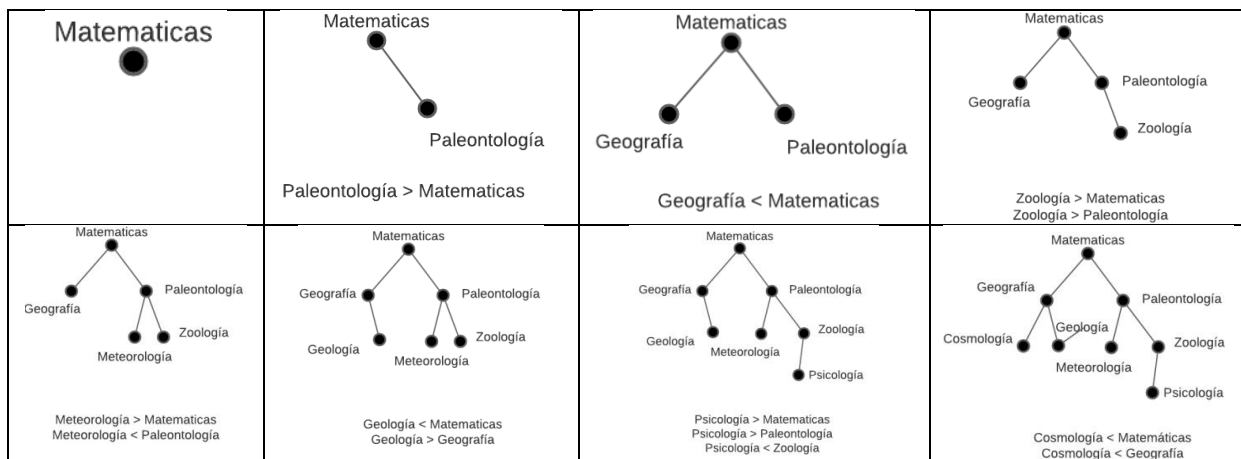
En conclusión, los árboles binarios de búsqueda son una herramienta importante en la informática para almacenar y buscar elementos de manera eficiente. En matemáticas discretas, se pueden entender en términos de conjuntos ordenados y relaciones de orden, lo que nos permite aplicar conceptos de teoría de conjuntos y relaciones de orden para entender mejor su funcionamiento.

Ejemplo 1

Construye un árbol binario de búsqueda para las palabras matemáticas, paleontología, geografía, zoología, meteorología, geología, psicología y cosmología (utilizando el orden alfabético).

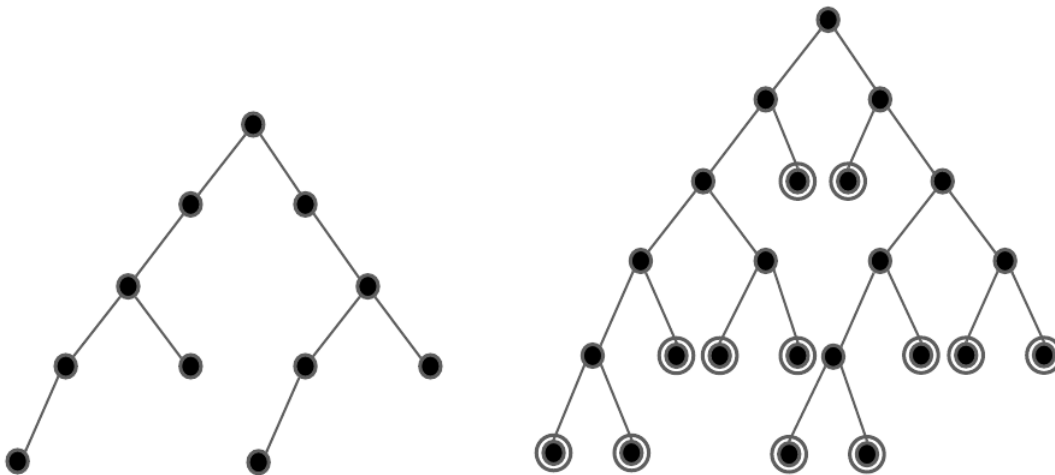
Solución: La Figura 1 muestra los pasos realizados en la construcción del árbol binario de búsqueda. La palabra matemáticas es la clave de la raíz. Puesto que paleontología es posterior a matemáticas (en el orden alfabético), añadimos el hijo derecho de la raíz con la clave paleontología. Puesto que geografía es anterior a matemáticas, añadimos el hijo izquierdo de la raíz con la clave geografía. Seguidamente, añadimos el hijo derecho del vértice etiquetado con paleontología y le asignamos la clave zoología, puesto que en el orden alfabético zoología es posterior a matemáticas y a paleontología. Análogamente, añadimos el hijo izquierdo del vértice de clave paleontología y le asignamos a este nuevo vértice la clave meteorología. Añadimos el hijo derecho al vértice con clave zoología y le asignamos la clave psicología. Por último, incorporamos el hijo izquierdo del vértice con clave geografía y le asignamos la clave cosmología.

Para localizar un elemento intentamos añadirlo a un árbol de búsqueda binario. Lo localizamos si está presente en el árbol. El Algoritmo 1 proporciona un pseudocódigo para localizar un elemento en un árbol binario de búsqueda y añadir un nuevo vértice con ese elemento como clave si éste no se encontrara en el árbol. El Algoritmo 1 localiza a x si es la clave de un vértice. Cuando x no es una clave, se añade un nuevo vértice con clave x . En el pseudocódigo, v es el vértice que tiene a x como su clave y $etiqueta(v)$ representa la clave del vértice v .



Vamos a determinar la complejidad computacional de este procedimiento. Supongamos que tenemos un árbol binario de búsqueda T par una lista de n elementos. Podemos construir, a partir de T , un árbol binario completo U sin más que añadir vértices no etiquetados cuando sea necesario de modo que cada vértice con clave tenga dos hijos. Este proceso se ilustra en la siguiente figura. Una vez hecho esto, podemos localizar fácilmente un elemento o añadir uno nuevo mediante una clave sin tener que añadir un vértice.

El mayor número de comparaciones necesarias para añadir un nuevo elemento es la longitud del camino más largo en U desde la raíz hasta una hoja. Los vértices internos de U son los vértices de T . De ahí se deduce que U tiene n vértices internos. Podemos utilizar la parte (2) del Teorema 4 de la sección 9.1 para concluir que U tiene $n + 1$ hojas. Utilizando el Corolario I de la sección 9.1 obtenemos que la altura de U es mayor o igual que $h = \lceil \log(n + 1) \rceil$. En consecuencia, es necesario realizar al menos $\lceil \log(n + 1) \rceil$ comparaciones para añadir un elemento. Nótese que si U está equilibrado, su altura es $\lceil \log(n + 1) \rceil$ (por el Corolario I de la sección 9.1). Así, si un árbol binario de búsqueda está equilibrado, localizar o insertar un elemento no requiere más de $\lceil \log(n + 1) \rceil$ comparaciones. Un árbol binario de búsqueda puede dejar de estar equilibrado si se insertan nuevos elementos. Puesto que los árboles de búsqueda binarios y equilibrados proporcionan complejidad óptima en el peor caso para la búsqueda binaria, los algoritmos han sido diseñados de manera que el árbol de búsqueda se reequilibra a medida que se insertan elementos.



ALGORITMO 1 Algoritmo para árboles binarios de búsqueda

```
procedure insertar(T: árbol binario de búsqueda, x: elemento)
  v := raíz de T
  {los vértices que no son de T tienen valor null}
  while v ≠ null y etiqueta(v) ≠ x
  begin
    if x < etiqueta(v) then
      if hijo izquierdo de v ≠ null then v := hijo izquierdo de v
      else añadir nuevo vértice como el hijo izquierdo de v a T y asignar v := null
    else
      if hijo derecho de v ≠ null then v := hijo derecho de v
      else añadir nuevo vértice como el hijo derecho de v a T y asignar v := null
  end
  if raíz de T = null then añadir un vértice v al árbol y etiquetarlo como x
  else if v es null o etiqueta(v) ≠ x then etiquetar nuevo vértice con x y sea v este nuevo vértice
  {v = posición de x}
```

ÁRBOLES DE DECISIÓN

Los árboles con raíz pueden utilizarse para modelar problemas en los que una serie de decisiones llevan a una solución. Por ejemplo, un árbol binario de búsqueda puede emplearse para localizar elementos basándose en una serie de comparaciones, donde cada una de ellas nos dice si hemos localizado o no el elemento o si deberíamos ir hacia el subárbol derecho o hacia el subárbol izquierdo. Un árbol con raíz en el que cada vértice interno corresponde a una decisión, con un subárbol en dichos vértices para cada posible resultado de la decisión, se llama árbol de decisión. Las posibles soluciones del problema corresponden a los caminos desde la raíz hasta las hojas. El en siguiente ejemplo se ilustra una aplicación de los árboles de decisión.

Ejemplo: Supongamos que tenemos siete monedas. todas de igual peso, y una moneda falsa que pesa menos que las restantes. ¿Cuántas pesadas, utilizando una balanza, son necesarias para determinar cuál de las ocho monedas es la moneda falsa? Da un algoritmo para encontrar esa moneda.

Solución: Hay tres posibilidades para cada pesada en la balanza. Los dos platillos pueden tener igual peso. el primer platillo puede pesar más o puede pesar más el segundo platillo. Por tanto, el árbol de decisión, para la secuencia de pesadas es ternario. Hay, al menos, ocho hojas en el árbol de decisión, puesto que hay ocho posibles resultados (ya que cada una de las ocho monedas puede ser la falsa), y cada posible resultado debe estar representado por, al menos, una hoja. El mayor número de pesadas necesario para encontrar la moneda falsa es la altura del árbol de decisión. Por el Corolario 1 se sabe que la altura del árbol es al menos $\lceil \log_3 8 \rceil = 2$. De ahí, son necesarias al menos dos pesadas.

Se puede determinar qué moneda es la falsa utilizando dos pesadas. En la siguiente figura aparece el árbol de decisión que ilustra cómo hacerlo.

COROLARIO 1: Si un árbol m -ario de altura h tiene l hojas, entonces $h \geq \lceil \log_m l \rceil$. (Estamos utilizando la función *parte entera por exceso*. Recordamos $\lceil x \rceil$ es el menor entero mayor o igual que x .)

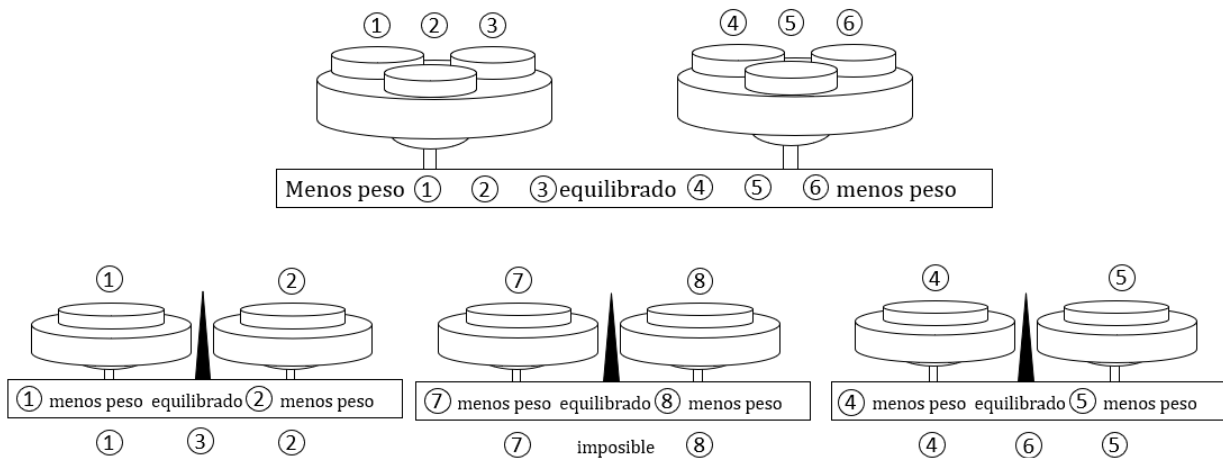


Figura: Un árbol de decisión para localizar una moneda falsa. La moneda falsa se señala en negrita debajo de cada una de las pesadas finales.

LA COMPLEJIDAD DE LOS ALGORITMOS DE ORDENACIÓN

Con el tiempo, se han desarrollado muchos y variados algoritmos de ordenación. Para decidir si un algoritmo de ordenación es eficiente o no, debemos calcular su complejidad. Utilizando los árboles de decisión como modelos, podemos calcular una cota inferior de la complejidad de los algoritmos de ordenación para el peor caso.

Nótese que dados n elementos, hay $n!$ formas posibles de ordenarlos, puesto que cada una de las $n!$ permutaciones podría ser la ordenación correcta. Los algoritmos de ordenación con mayor frecuencia, están basados en comparaciones binarias, esto es, la comparación de dos elementos en cada paso. El resultado de dicha comparación reduce el conjunto de órdenes posibles. Así, un algoritmo de comparación basado en comparaciones binarias se puede representar mediante un árbol de decisión en el que cada vértice interno representa una comparación de dos elementos. Cada hoja representa una de las $n!$ permutaciones de los n elementos.

Ejemplo: en la siguiente figura se muestra un árbol de decisión que ordena los elementos de la lista a, b, c .

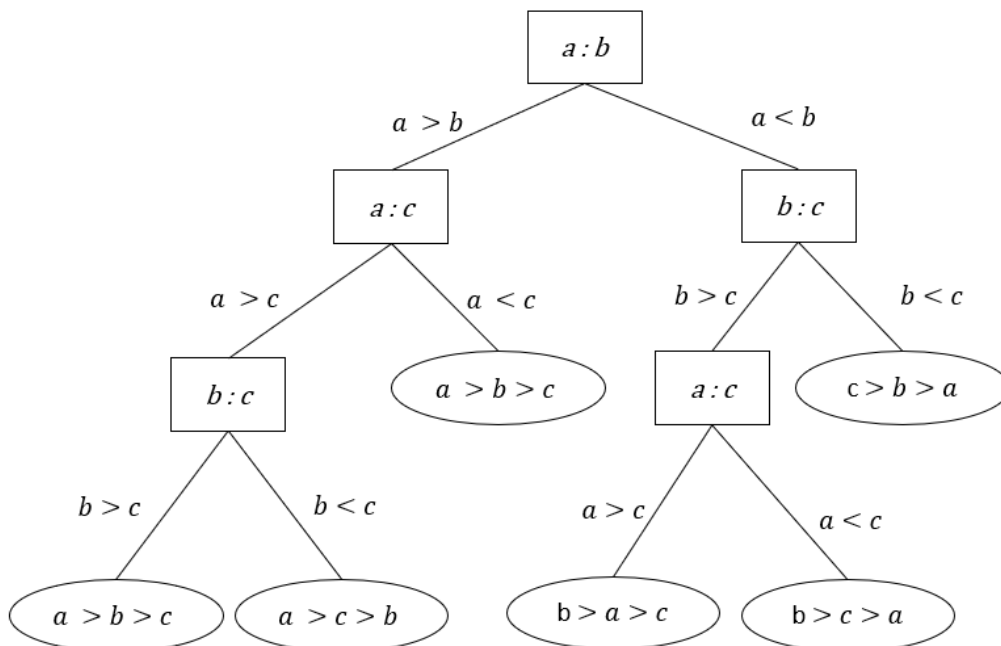


Figura: Un árbol de decisión para ordenar tres elementos distintos.

La complejidad de un algoritmo de ordenación basado en comparaciones binarias se mide en términos del número de comparaciones realizadas. El mayor número de comparaciones necesaria para ordenar una lista cualquiera de n elementos proporciona el peor caso en la ejecución del algoritmo. El número máximo de tales comparaciones coincide con la longitud del camino más largo en el árbol de decisión que representa al procedimiento de ordenación. En otras palabras, el mayor número de comparaciones necesarias para ordenar cualquier lista es precisamente la altura del árbol de decisión. Puesto que la altura de un árbol binario con $n!$ hojas es al menos $\lceil \log n! \rceil$ (utilizando el Corolario 1), se necesitan al menos $\lceil \log n! \rceil$ comparaciones, como se establece en el Teorema 1.

TEOREMA 1: Un algoritmo de ordenación basado en comparaciones binarias requiere al menos $\lceil \log n! \rceil$ comparaciones.

Podemos utilizar el Teorema 1 para dar una estimación en notación Ω del número de comparaciones realizadas por un algoritmo de ordenación basado en comparaciones binarias. Basta con observar que $\lceil \log n! \rceil$ es $\Theta(n \log n)$, una de las funciones de referencia más comunes a la hora de estimar la complejidad computacional de algoritmos.

TEOREMA 2: El promedio del número de comparaciones realizadas por un algoritmo que ordena n datos haciendo comparaciones binarias es $\Omega(n \log n)$.

CÓDIGOS INSTANTÁNEOS

En teoría de la información, los códigos instantáneos son códigos prefijos que se utilizan para codificar secuencias de símbolos de longitud variable, de tal manera que no haya ninguna secuencia que sea prefijo de otra secuencia codificada.

En otras palabras, los códigos instantáneos aseguran que cada símbolo de la secuencia se codifica con una cantidad mínima de bits, sin la necesidad de utilizar un carácter especial para indicar el final de cada símbolo. Esto permite una compresión de datos más eficiente, ya que se pueden representar secuencias de símbolos más largas con menos bits que si se utilizaran códigos fijos de longitud constante.

Los códigos instantáneos se pueden representar gráficamente mediante un árbol, donde cada símbolo se asocia con un camino desde la raíz hasta una hoja. La longitud del camino que lleva a cada símbolo determina su longitud de codificación, y se garantiza que no hay ningún camino que sea prefijo de otro camino. De esta manera, se puede obtener una codificación sin ambigüedades y eficiente en términos de espacio de almacenamiento.

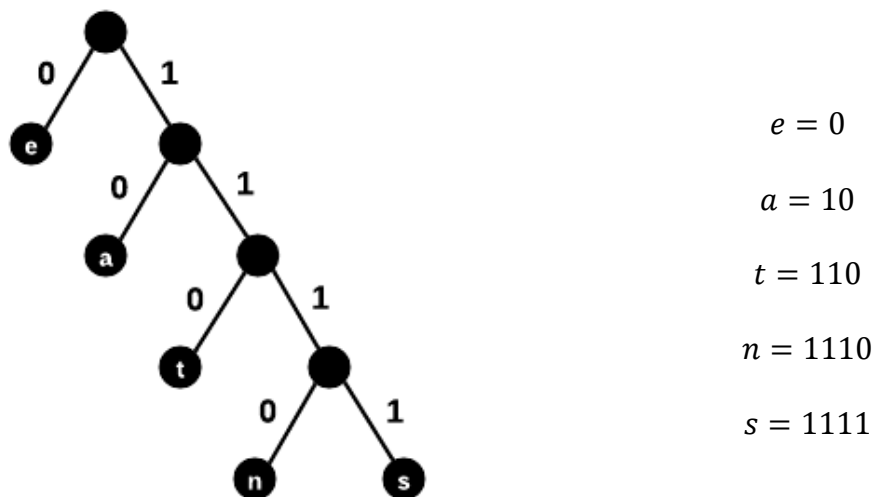


Figura: El árbol binario de código instantáneo.

CÓDIGOS DE HUFFMAN

Los códigos de Huffman son un tipo de códigos de longitud variable utilizados en compresión de datos. Fueron inventados por David A. Huffman en 1952 y son muy utilizados en la compresión de archivos de texto y otros tipos de datos.

El objetivo de los códigos de Huffman es asignar códigos más cortos a los símbolos que aparecen con mayor frecuencia en un conjunto de datos y códigos más largos a los símbolos menos frecuentes. De esta manera, se puede lograr una compresión más eficiente, ya que los símbolos más frecuentes tendrán códigos más cortos y, por lo tanto, se representarán con menos bits.

El proceso para generar los códigos de Huffman implica construir un árbol de codificación basado en la frecuencia de aparición de cada símbolo en el conjunto de datos. Los símbolos se colocan en las hojas del árbol y se fusionan gradualmente hacia arriba, generando códigos de longitud variable a medida que se sube por el árbol. Los códigos se asignan a cada símbolo de forma tal que los códigos asignados a cada símbolo sean únicos y no haya códigos que sean prefijos de otros códigos.

Una vez que se ha construido el árbol de codificación, se pueden utilizar los códigos de Hoffman para comprimir los datos. Cada símbolo en el conjunto de datos se reemplaza por su correspondiente código de Hoffman antes de ser almacenado o transmitido. Como los símbolos más frecuentes tienen códigos más cortos, esto reduce la cantidad de bits necesarios para representar los datos, lo que resulta en una mayor eficiencia de almacenamiento o transmisión.

ALGORITMO 2 Codificación de Huffman

```
procedure Huffman(C: los símbolos  $a_i$  con frecuencias  $w_i, i = 1, \dots, n$ )  
B := bosque de  $n$  árboles con raíz, cada uno de los cuales consta de un único vértice  $a_i$  con  
peso  $w_i$   
while F no sea un árbol  
begin  
    Sustituir los árboles de F de menor peso, T y T', con  $w(T) \geq w(T')$ , por un árbol que tie-  
    ne una nueva raíz, y que tiene a T como subárbol izquierdo y a T' como subárbol dere-  
    cho. Etiquetar la arista nueva de la raíz a T con 0 y la nueva arista de la raíz a T' con 1.  
    Asignar  $w(T) + w(T')$  como el peso del nuevo árbol.  
end  
{La codificación de Huffman para el símbolo  $a_i$  es la concatenación de las etiquetas de las  
aristas en el único camino desde la raíz hasta el vértice  $a_i$ }
```

Ejemplo: Utiliza la codificación de Huffman para cifrar los siguientes símbolos con las frecuencias dadas: A:0.08, B:0.10, C:0.12, D:0.15, E:0.20, F:0.35. ¿Cuál es el promedio de bits utilizado para codificar un carácter?

Solución: En la siguiente figura se muestran los pasos realizados para codificar estos símbolos. Las codificaciones resultantes son las siguientes: la de A es 111 la de B es 110, la de C es 011, la de D es 010, la de E es 10y la de F es 00. El número promedio de bits utilizado para codificar un símbolo es:

$$3 \cdot 0,08 + 3 \cdot 0,10 + 3 \cdot 0,12 + 3 \cdot 0,15 + 2 \cdot 0,20 + 2 \cdot 0,35 = 2,45.$$

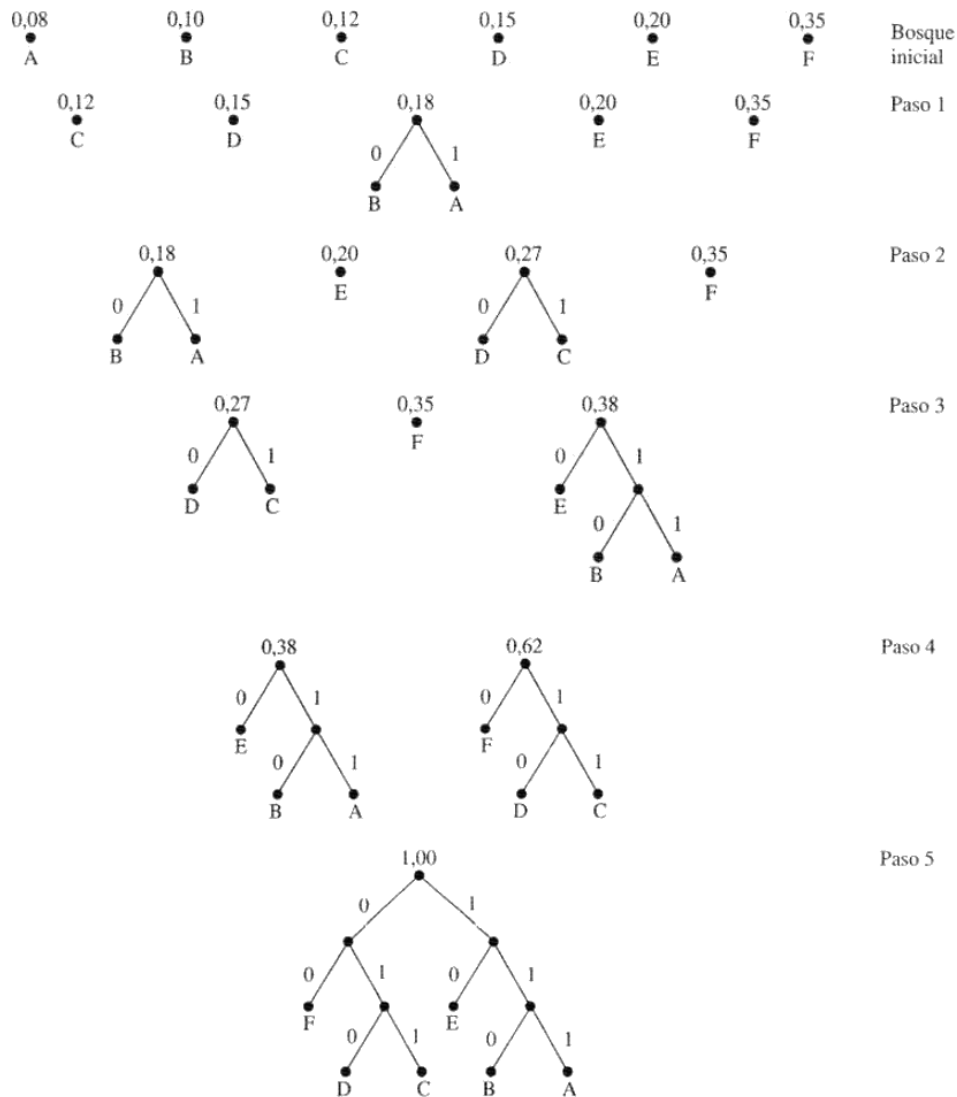


Figura: Codificación de Huffman de los símbolos del anterior ejemplo.

ARBOLES DE JUEGOS

En teoría de juegos, los árboles de juegos son estructuras de datos que se utilizan para representar gráficamente un juego, con el fin de analizar las diferentes opciones y estrategias disponibles para los jugadores.

Un árbol de juegos se construye mediante la representación de cada estado posible del juego en un nodo del árbol. A partir de cada nodo, se dibujan ramas que representan las diferentes jugadas posibles que se pueden realizar desde ese estado. Cada rama está etiquetada con la jugada que se realiza y con el resultado que se obtiene al realizar esa jugada.

Los árboles de juegos se utilizan para analizar diferentes estrategias y tomar decisiones informadas. Por ejemplo, un jugador puede explorar el árbol de juegos para determinar la mejor jugada posible en cada estado del juego. También se pueden utilizar técnicas como la estrategia de minimax y la poda alfa-beta para buscar el camino óptimo en el árbol de juegos.

En general, los árboles de juegos son útiles para representar y analizar juegos de dos o más jugadores, en los que cada jugador toma decisiones que afectan el resultado final del juego. Algunos ejemplos de juegos que se pueden representar mediante árboles de juegos son el ajedrez, el póker y el tic-tac-toe.

Tres en raya. El árbol de juego para el tres en raya es extremadamente largo y no puede dibujarse aquí, aunque cualquier ordenador puede construirlo fácilmente. Mostramos una parte del juego en la Figura G. Nótese que, si eliminamos las posiciones simétricas equivalentes, sólo necesitamos las tres posiciones iniciales de la Figura G. También mostramos un subárbol de este árbol de juego que lleva a la posición terminal de la Figura H, donde un jugador que puede ganar realiza un movimiento ganador.

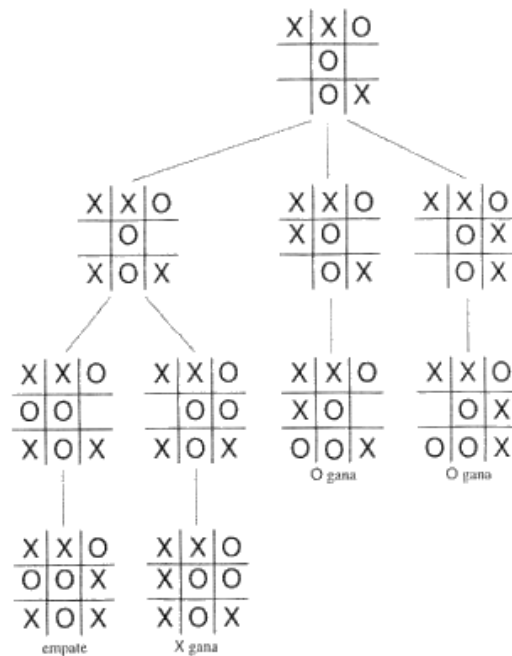


Figura G.

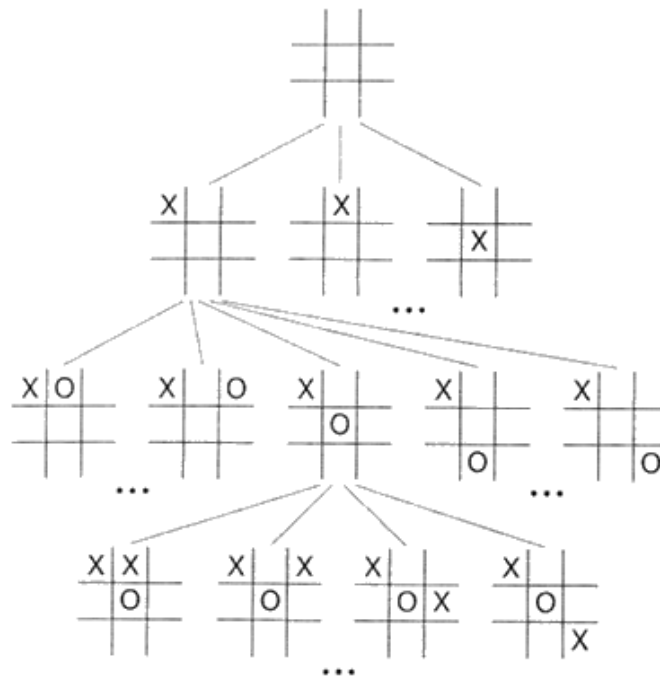


Figura H.

Definición 1: El valor de un vértice en un árbol de juego se define recursivamente como:

1. El valor de una hoja es el pago del primer jugador cuando el juego finaliza en la posición representada por esa hoja.
2. El valor de un vértice interno en un nivel par es el máximo de los valores de sus hijos y el valor de un vértice interno de nivel impar es el mínimo de los valores de sus hijos.

TEOREMA 3: El valor de un vértice de un árbol de juego nos da la puntuación obtenida por el primer jugador si ambos jugadores siguen la estrategia de minimax y el juego comienza desde la posición representada por dicho vértice.

MINIMAX

La estrategia de minimax es un algoritmo utilizado en teoría de juegos para encontrar la mejor jugada en juegos de suma cero con información perfecta, donde dos jugadores compiten por obtener el mejor resultado posible en un juego determinado.

En la estrategia de minimax, se representa el juego mediante un árbol de búsqueda, donde cada nodo representa un estado del juego y las ramas que salen del nodo corresponden a las posibles jugadas que puede realizar el jugador en ese estado. En cada nodo del árbol, se calcula el valor minimax, que representa la mejor respuesta que el jugador contrario puede dar a la jugada actual.

El jugador maximizador trata de maximizar el valor minimax, es decir, elegir la jugada que maximiza su resultado, mientras que el jugador minimizador trata de minimizar el valor minimax, es decir, elegir la jugada que minimiza el resultado del jugador maximizador.

De esta manera, la estrategia de minimax se basa en la suposición de que el oponente también está tratando de maximizar sus ganancias, por lo que el jugador debe considerar las posibles respuestas del oponente para tomar la mejor decisión posible en cada turno.

En resumen, la estrategia de minimax en árboles de juegos es un enfoque para encontrar la mejor jugada en juegos de dos jugadores, donde el jugador maximizador trata de maximizar.

RECORRIDOS EN ÁRBOLES

Introducción

Los árboles ordenados con raíz se utilizan frecuentemente para almacenar la información. Necesitamos procedimientos que permitan visitar cada uno de los vértices de dichos árboles para acceder a los datos. Describiremos varios algoritmos importantes para recorrer los vértices de un árbol ordenado con raíz. Los árboles ordenados con raíz también pueden utilizarse para representar varios tipos de expresiones, tales como expresiones aritméticas que involucran números, variables y operadores. A la hora de evaluar una expresión, puede ser útil disponer de todas las posibles listas de vértices de los árboles ordenados con raíz que se usan para representar dicha expresión.

Sistema de etiquetado universal

Los procedimientos para recorrer todos los vértices de un árbol ordenado con raíz se basan en las ordenaciones definidas en los hijos. En este tipo de árboles, los hijos de un vértice interno se muestran de izquierda a derecha al dibujar estos grafos dirigidos.

Describiremos un modo de ordenar totalmente los vértices de un árbol ordenado con raíz.

Para construir este orden total, primero debemos etiquetar todos los vértices. Esto lo hacemos recursivamente:

1. Etiquetamos la raíz con el entero 0. Luego etiquetamos sus k hijos (al nivel 1) de izquierda a derecha con los enteros 1, 2, 3, ..., k .
2. Para cada vértice v del nivel n con etiqueta A , etiquetamos sus k_v hijos, de izquierda a derecha, como $A.1, A.2, \dots, A.k$.

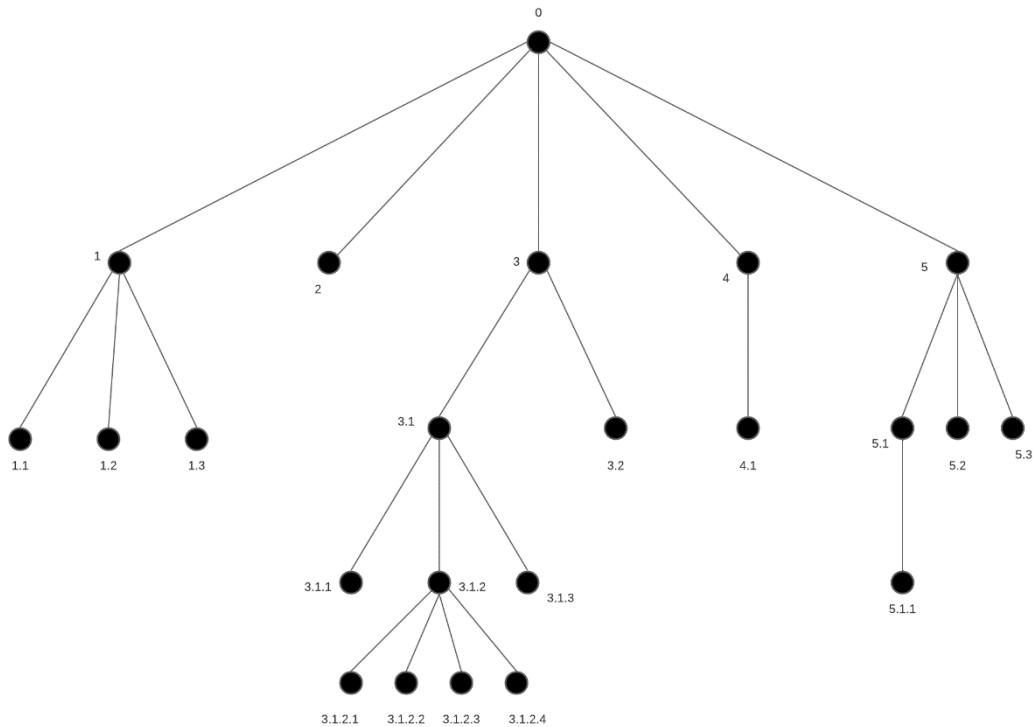
Siguiendo este proceso, un vértice v del nivel n , para $n \geq 1$, está etiquetado por $x_1 \cdot x_2 \cdot \dots \cdot x_n$ donde el único camino desde la raíz hasta v pasa por el vértice x_1 en el nivel 1, por el vértice x_2 en el nivel 2, etc. Este modo de etiquetar se denomina **sistema de etiquetado universal o canónico** del árbol ordenado con raíz.

Podemos ordenar totalmente los vértices utilizando el orden lexicográfico de las etiquetas asignadas con el sistema de etiquetado universal. El vértice por $x_1 \cdot x_2 \cdot \dots \cdot x_n$, es menor que el vértice $y_1 \cdot y_2 \cdot \dots \cdot y_m$ si existe un índice i , $0 \leq i \leq n$, con $x_1 = y_1, x_2 = y_2, \dots, x_{i-1} = y_{i-1}$ y $x_i < y_i$, o bien si $n < m$ y $x_i = y_i$, para $i = 1, 2, \dots, n$.

Ejemplo 1

En el árbol ordenado de la figura 1 se muestran junto a los vértices las etiquetas proporcionadas por el sistema de etiquetado universal. El orden lexicográfico de las etiquetas es:

$0 < 1 < 1.1 < 1.2 < 1.3 < 2 < 3 < 3.1 < 3.1.1 < 3.1.2 < 3.1.2.1 < 3.1.2.2 < 3.1.2.3 < 3.1.2.4 < 3.1.3 < 3.2 < 4 < 4.1 < 5 < 5.1 < 5.1.1 < 5.2 < 5.3$.



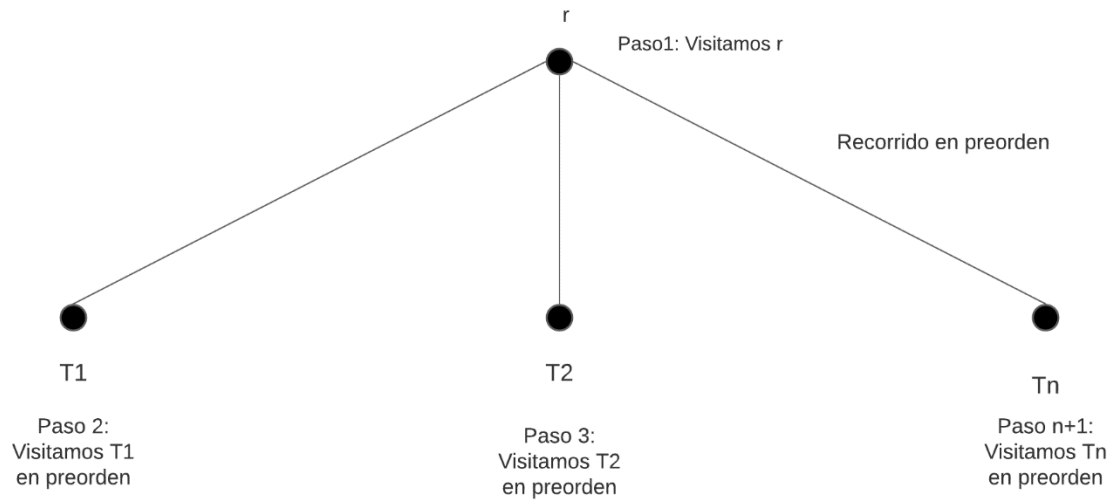
Algoritmos de recorrido

Los procedimientos para el recorrido sistemático de los vértices de un árbol ordenado con raíz se llaman algoritmos de recorrido de un árbol. Describiremos tres de estos algoritmos. Los que se utilizan con mayor frecuencia, llamados recorrido en preorden, recorrido en inorden y recorrido en postorden. Cada uno de estos algoritmos se puede definir de modo recursivo. En primer lugar, definimos el recorrido en preorden.

Definición 1

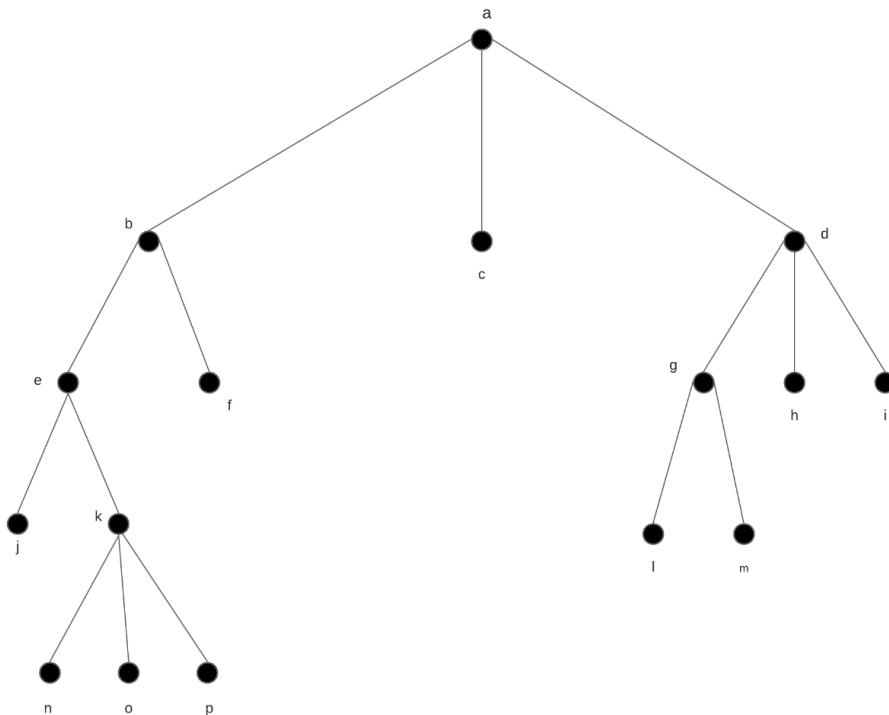
Sea T un árbol ordenado con raíz r . Si T consta sólo de r , entonces r es el recorrido en preorden de T . En otro caso, supongamos que T_1, T_2, \dots, T_n , son los subárboles de r listados de izquierda a derecha en T . El recorrido en preorden comienza visitando r . continúa recorriendo T_1 , en preorden, luego T_2 , en preorden y así sucesivamente hasta recorrer T , en preorden.

El lector puede verificar que el recorrido en preorden de un árbol ordenado con raíz proporciona el mismo orden en los vértices que el obtenido con el sistema de etiquetado universal. La figura indica como se lleva a cabo dicho recorrido.



Ejemplo 2

¿En qué orden visita un recorrido en preorden los vértices del árbol ordenado con la raíz de la figura 3?



Solución: Los pasos realizados en el recorrido en preorden de T se muestran en la Figura 4, Atravesamos T en preorden listando, en primer lugar, la raíz a, seguida de la lista recorrida en preorden del subárbol con raíz b. la lista recorrida en preorden del subárbol de raíz c (que es simplemente c) y la lista recorrida en preorden del subárbol con raíz d.

La lista recorrida en preorden del subárbol con raíz b comienza en b, seguida de la lista recorrida en preorden del subárbol con raíz e y luego la lista recorrida en preorden del subárbol con raíz f (que es simplemente f). La lista recorrida en preorden del subárbol con raíz d comienza en d, seguida de la lista recorrida en preorden del subárbol con raíz g. seguida de la lista recorrida en preorden del subárbol con raíz h (que es sólo h), finalizando con el subárbol con raíz i (que es simplemente i).

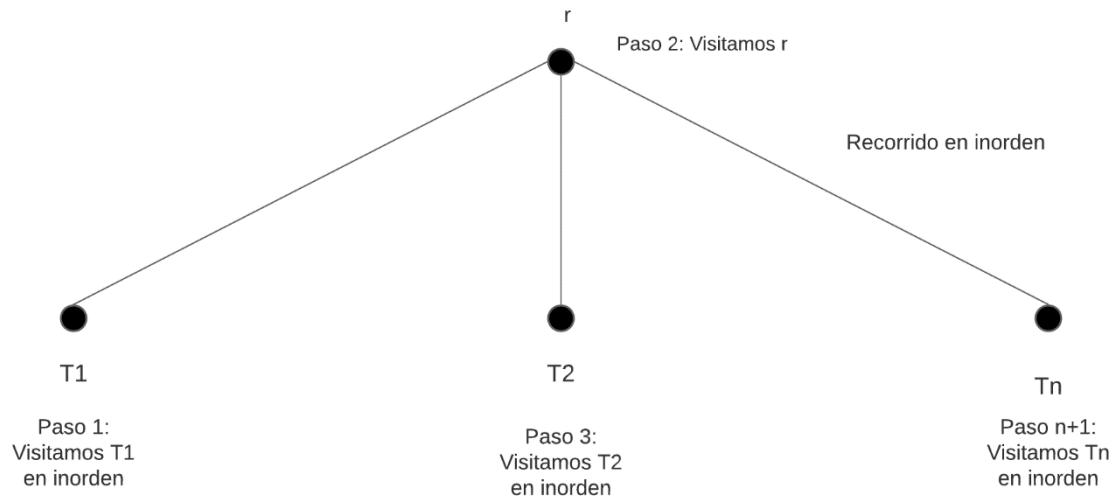
La lista recorrida en preorden del subárbol con raíz e comienza con el vértice e. seguida por la lista recorrida en preorden del subárbol con raíz j (que es sólo j), seguida por la lista recorrida en preorden del subárbol con raíz k. La lista recorrida en preorden del subárbol con raíz g es g. seguida de l y a continuación de m. La lista, en preorden, relativa al subárbol con raíz k es k, n, o, p.

En consecuencia, el recorrido en preorden de T es:

a, b, e, j, k, n, o, p, f, c, d, g, l, m, h, i.

Definición recorrido en inorden

Sea T un árbol ordenado con raíz r. Si T consta sólo de r, entonces r es el recorrido en inorden de T. En otro caso, supongamos que T_1, T_2, \dots, T_n son los subárboles de r listados de izquierda a derecha en T. El recorrido en inorden comienza recorriendo T_1 , en inorden y continúa visitando r, a continuación, recorre T_2 en inorden, después T_3 en inorden y así sucesivamente hasta recorrer T_n en inorden. En términos prácticos, esto significa que, para cada nodo visitado en el recorrido, primero se visitan todos los nodos de su subárbol izquierdo, luego se visita el propio nodo y finalmente se visitan todos los nodos de su subárbol derecho.

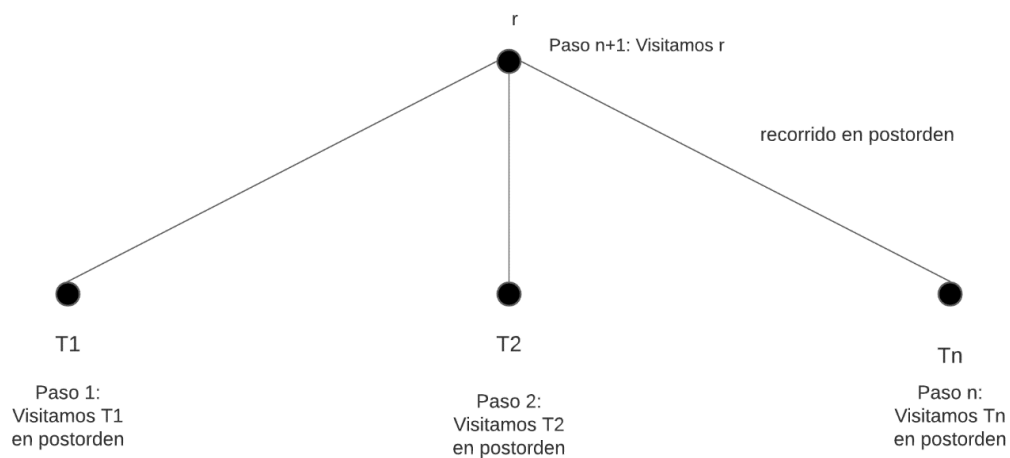


Tomando en cuenta el ejemplo del árbol figura 3, si revisamos como seria la lista en inorden, quedaría de esta manera: j, e, n, k, o, p, b, f, a, c, l, g, m, d, h, i.

Definición “recorrido en postorden”

Sea T un árbol ordenado con raíz r . Si T consta sólo de r , entonces r es el recorrido en postorden de T . En otro caso, supongamos que T_1, T_2, \dots, T_n , son los subárboles de r listados de izquierda a derecha en T . El recorrido en postorden comienza recorriendo T_1 en postorden, luego recorre T_2 en postorden y así sucesivamente hasta recorrer T_n en postorden y finaliza visitando r .

En términos prácticos, esto significa que, para cada nodo visitado en el recorrido, primero se visitan todos los nodos de su subárbol izquierdo, luego se visitan todos los nodos de su subárbol derecho y finalmente se visita el propio nodo.



Tomando en cuenta el ejemplo del árbol figura 3, si revisamos como sería la lista en postorden, quedaría de esta manera: j, n, o, p, k, e, f, b, c, l, m, g, h, i, d, a.

Hay métodos sencillos para enumerar los vértices de un árbol con cualquiera de los tres órdenes presentados anteriormente. Para ello, primero dibujamos una curva alrededor de un árbol ordenado con raíz comenzando en la raíz y moviéndose a lo largo de las aristas como se muestra en el ejemplo de la Figura 9.

Podemos enumerar los vértices en preorden sin más que listar cada vértice la primera vez que la curva pasa por él. Podemos hacerlo en inorden cuando listamos una hoja la primera vez que la curva pasa por ella y listando cada vértice interno la segunda vez que la curva pasa por él. Podemos enumerar los vértices en postorden sin más que listar cada vértice la última vez que pasamos por él en el camino de vuelta hacia su padre. Cuando se hace esto en el árbol de la Figura 9, se obtiene que el recorrido en preorden da la lista a, b, d, h, e, i, j, c, f, g, k; en inorden es h, d, b, i, e, j, a, f, c, k, g y en postorden es h, d, i, j, e, b, f, k, g, c, a.

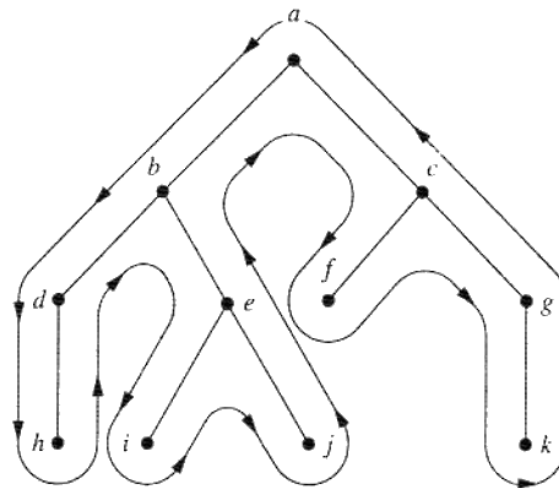


Figura 9. Una forma sencilla de recorrer un árbol ordenado con raíz en preorden, inorden y postorden.

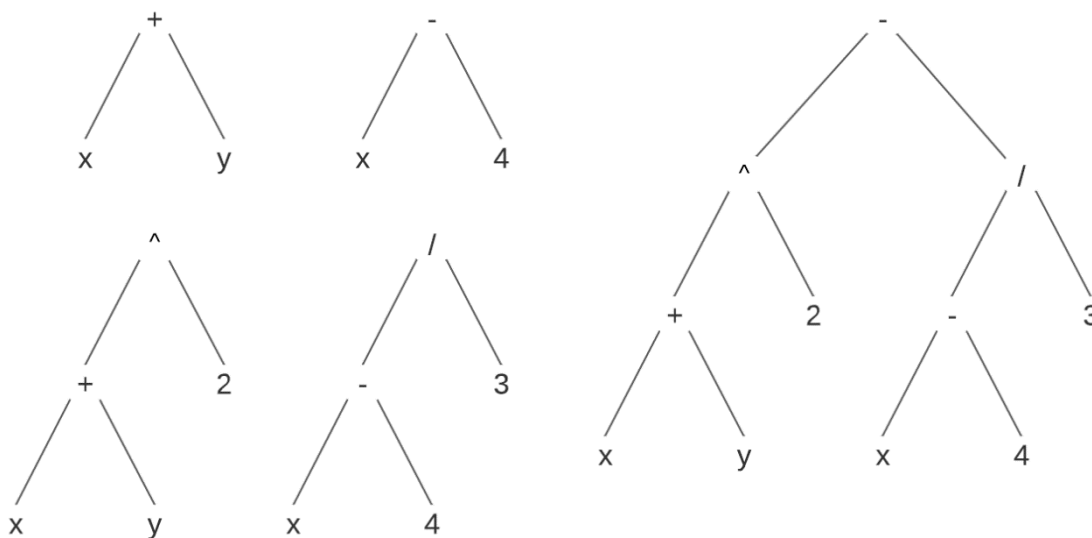
Notación Infija, Prefija y Postfija

Podemos presentar expresiones complejas, tales como fórmulas proporcionales, combinaciones de conjuntos y expresiones aritméticas, mediante árboles ordenados con raíz. Por ejemplo, consideremos la representación de una expresión aritmética que involucra los operadores $+$ (suma), $-$ (resta), $*$ (multiplicación), $/$ (división) y $^$ (exponenciación). Utilizamos los paréntesis para indicar el orden de las operaciones. Se puede utilizar un árbol ordenado con raíz para representar dichas expresiones, en el que los vértices internos representan operaciones y las hojas representan variables o números. Cada operación actúa sobre sus subárboles izquierdo y derecho.

Ejemplo 5

¿Cuál es el árbol ordenado con raíz que representan a la expresión $((x + y)^2) + ((x - 4)/3)$?

El árbol binario para esta expresión se puede construir de abajo a arriba. Primero, se construye un subárbol para la expresión $x + y$. Posteriormente, este subárbol se incorpora a otro mayor que representa a la expresión $(x + y)^2$. También se construye un subárbol para $x - 4$ y luego se incorpora al subárbol que representa a $(x - 4)/3$. Finalmente, los subárboles que representan a $(x + y)^2$ y $(x - 4)/3$ se combinan para formar un árbol ordenado con raíz que representa a la expresión $((x + y)^2) + ((x - 4)/3)$. Los diferentes pasos del proceso se muestran en la siguiente figura.



Ejemplo 6

¿Cuál es la forma prefija para $((x + y)^2) + ((x - 4)/3)$?

Obtenemos la forma prefija de esa expresión sin más que recorrer del árbol binario que la representa, mostrado en la anterior figura. El resultado es $+ \wedge + x y 2 / - x 4 3$.

Ejemplo 7

¿Cuál es el valor de la expresión prefija $+ - * 2 3 5 / ^ 2 3 4$?

Las evaluaciones de esta expresión, leída de derecha a izquierda y realizando las operaciones sobre los operadores a la derecha, se muestran en la siguiente figura. El valor de la expresión es 3.

$$\begin{array}{r}
 + \quad - \quad * \quad 2 \quad 3 \quad 5 \quad / \quad \underbrace{\quad \quad \quad}^{2 \quad 3 \quad 4} \\
 2 \wedge 3 = 8 \\
 + \quad - \quad * \quad 2 \quad 3 \quad 5 \quad / \quad \underbrace{\quad \quad \quad}_{8 \quad 4} \\
 8 / 4 = 2 \\
 + \quad - \quad \underbrace{\quad \quad \quad}_{2 \quad 3 \quad 5} \quad 2 \\
 2 * 3 = 6 \\
 + \quad - \quad \underbrace{\quad \quad \quad}_{6 \quad 5 \quad 2} \\
 6 - 5 = 1 \\
 \underbrace{\quad \quad \quad}_{+ \quad 1 \quad 2} \\
 1 + 2 = 3
 \end{array}$$

Ejemplo 8

¿Cuál es la forma postfija de la expresión $((x + y)^2) + ((x - 4) / 3)$?

La forma postfija de la expresión se obtiene llevando a cabo un recorrido en postorden por el árbol binario asociado a esta expresión, que se mostró anteriormente. El resultado es $x y + 2 ^ x 4 - 3 / +$.

Ejemplo 9

¿Cuál es el valor de la expresión postfija $7 2 3 * - 4 ^ 9 3 / +$?

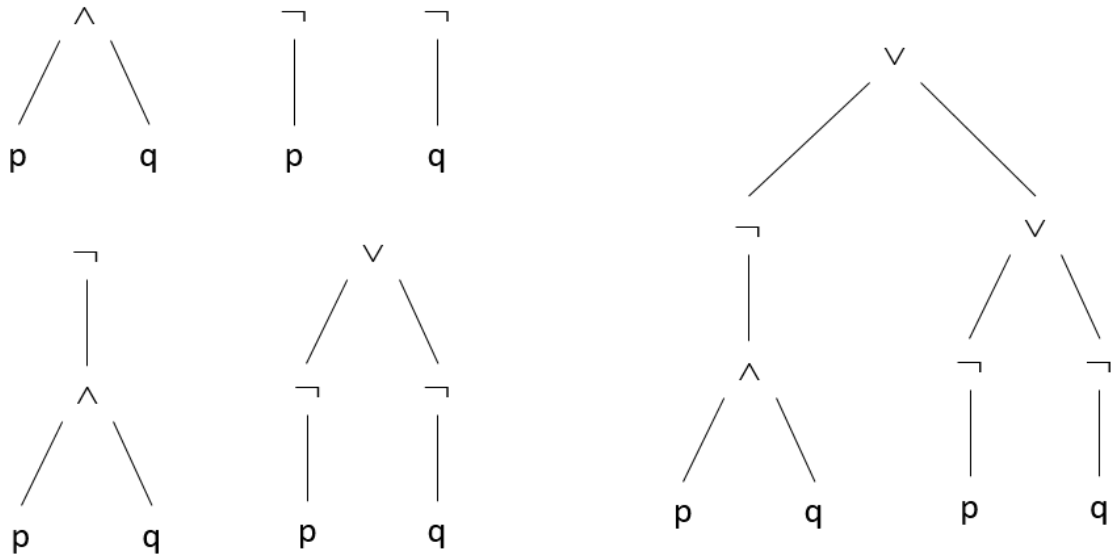
Las evaluaciones de esta expresión, leída de izquierda a derecha, y realizando las operaciones cuando dos operandos van seguidos de un operador se muestran en la siguiente figura. El valor de la expresión es 4.

$$\begin{array}{r} 7 \quad 2 \quad 3 \quad * \quad - \quad 4 \quad ^ \quad 9 \quad 3 \quad / \quad + \\ \hline 2 * 3 = 6 \\ 7 \quad 6 \quad - \quad 4 \quad ^ \quad 9 \quad 3 \quad / \quad + \\ \hline 7 - 6 = 1 \\ 1 \quad 4 \quad ^ \quad 9 \quad 3 \quad / \quad + \\ \hline 1 ^ 4 = 1 \\ 1 \quad 9 \quad 3 \quad / \quad + \\ \hline 9 / 3 = 3 \\ 1 \quad 3 \quad + \\ \hline 1 + 3 = 4 \end{array}$$

Ejemplo 10

Obtén el árbol ordenado con raíz que representa a la fórmula $(\neg(p \wedge q)) \leftrightarrow (\neg p \vee \neg q)$. Utiliza este árbol para obtener las formas prefija, postfija e infija de esta expresión.

El árbol con raíz de esta fórmula se construye de abajo para arriba. Primero se construye los subárboles de $\neg p$ y $\neg q$ (donde la negación se considera un conectivo unario). Posteriormente, el subárbol de $p \wedge q$. Después, los subárboles de $\neg(p \wedge q)$ y $(\neg p) \vee (\neg q)$. Finalmente, estos dos subárboles se utilizan en la construcción del árbol con raíz de la expresión. Los pasos de este proceso se muestran en la siguiente figura.



ÁRBOLES GENERADORES

Introducción: Considera la siguiente red de carreteras del estado de Maine representada por el grafo simple de la Figura 1(a). El único modo de que las carreteras puedan mantenerse abiertas en invierno es despejar con frecuencia el camino con una máquina quitanieves. El departamento de mantenimiento de carreteras quiere limpiar el menor número posible de carreteras de tal manera que dos ciudades cualesquiera queden siempre conectadas. ¿De qué modo se puede realizar esta labor?

Al menos cinco carreteras deben limpiarse de nieve para poder asegurar que hay un camino entre dos ciudades cualesquiera. La Figura 1(b) muestra uno de tales conjuntos de carreteras. Nótese que el subgrafo que representa estas carreteras es un árbol, puesto que es conexo y tiene seis vértices y cinco aristas.

Este problema se ha resuelto utilizando un subgrafo conexo con el mínimo número de aristas

y con todos los vértices del grafo original. Tal grafo debe ser un árbol.

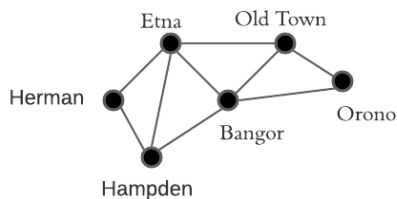


Figura 1(a)

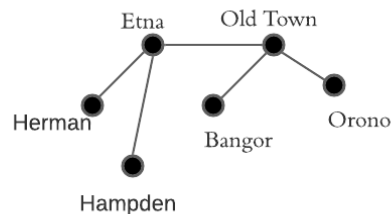


Figura 1(b)

DEFINICIÓN 1: Sea G un grafo simple, un *árbol generador* (o recubridor) de G es un subgrafo G que es un árbol y contiene todos los vértices de G .

Un grafo simple que admite un árbol generador necesariamente es conexo, puesto que existe un camino en el árbol generador entre dos vértices cualesquiera. El recíproco también es cierto, esto es, todo grafo simple conexo tiene un árbol generador. Daremos un ejemplo antes de demostrar este resultado.

EJEMPLO: Obtener un árbol generador del grafo simple G de la siguiente figura:

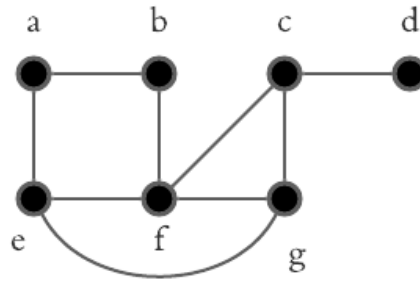
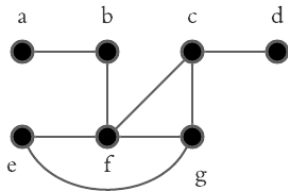
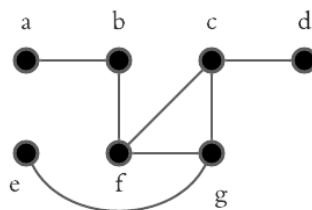


Figura: Grafo simple G .

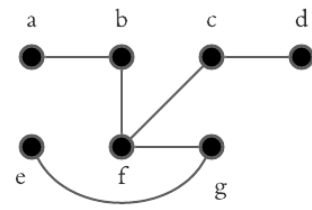
Solución: El grafo G es conexo, pero no es un árbol porque contiene ciclos. Quitamos la arista $\{a, e\}$. De este modo, eliminamos un ciclo y el subgrafo resultante todavía es conexo y contiene todos los vértices de G . Seguidamente, quitamos la arista $\{e, f\}$ para eliminar otro ciclo. Finalmente, quitamos la arista $\{c, g\}$ para obtener un grafo simple sin circuitos. Este subgrafo es un árbol generador, puesto que es un árbol que contiene todos los vértices de G . La sucesión de aristas eliminadas en el proceso de obtención del árbol generador se muestra en la siguiente figura.



Arista eliminada: $\{a, e\}$.
(a)



$\{e, f\}$
(b)



$\{c, g\}$
(c)

Figura: Construcción de un árbol generador de G eliminando aristas que forman ciclos.

OBSERVACIÓN: El árbol mostrado en la Figura no es el único árbol generador de G . Por ejemplo, cada uno de los árboles de la Figura 4 también es un árbol generador de G .

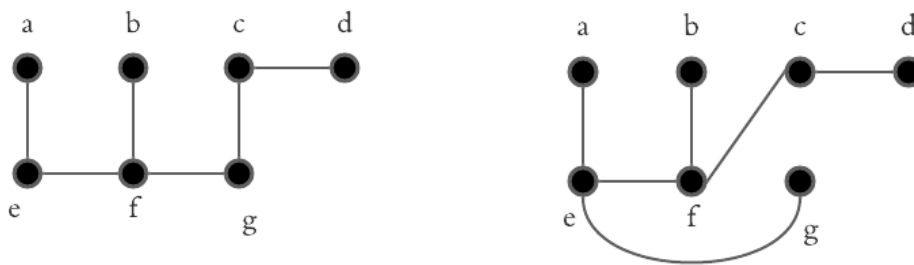


Figura 4: Árboles generadores de G .

TEOREMA 1: Un grafo simple es conexo si, y sólo si, admite un árbol generador.

BÚSQUEDA EN PROFUNDIDAD

La demostración del Teorema 1 proporciona un algoritmo para obtener árboles generadores mediante la supresión de aristas de los circuitos simples. Este algoritmo no es eficiente, puesto que requiere identificar los circuitos simples de algún modo. En lugar de construir árboles generadores eliminando aristas, los árboles generadores se pueden obtener añadiendo aristas. Presentaremos en lo que sigue dos algoritmos basados en esta estrategia.

Podemos construir un árbol generador para un grafo simple conexo utilizando una **búsqueda en profundidad**. Construiremos un árbol con raíz, y el árbol generador será el grafo no dirigido subyacente. Elegimos un vértice arbitrario como raíz del árbol. Formamos un camino que comienza en este vértice añadiendo sucesivamente vértices y aristas, siendo cada nueva arista incidente con el último vértice del camino y un vértice que no está en el camino.

Añadimos a este camino tantos vértices y aristas como sea posible. Si el camino pasa por todos los vértices del grafo, el árbol generador es dicho camino. En caso contrario, se deben añadir más vértices y aristas. Retrocedemos al penúltimo vértice del camino y, si es posible, formamos un nuevo camino comenzando en este vértice y que pase por los nodos no visitados. Si esto no se puede hacer, retrocedemos al vértice anterior en el recorrido hacia la raíz y lo intentamos de nuevo.

Repetimos el proceso, comenzando en el último vértice visitado, retrocediendo un vértice en cada ocasión, construyendo caminos de longitud tan grande como sea posible hasta que no se puedan añadir más aristas. Puesto que el grafo tiene un número finito de aristas y es conexo, este proceso acaba generando un árbol recubridor. Cada vértice final de un camino en una etapa del algoritmo es una hoja del árbol con raíz, y cada vértice que sirve para comenzar un camino es un nodo interno.

El lector observará la naturaleza recursiva de este algoritmo. También cabe señalar que, si los vértices del grafo están ordenados, las elecciones de las aristas en cada paso del procedimiento están todas determinadas cuando elegimos el primer vértice en el orden establecido. Sin embargo, no siempre ordenaremos los vértices de un grafo explícitamente.

La búsqueda en profundidad también se llama **vuelta atrás** o **retroceso** (en inglés, backtracking), porque el algoritmo vuelve a visitar vértices por los que ya ha pasado para añadir caminos. El siguiente ejemplo ilustra la técnica del backtracking.

EJEMPLO:

Utiliza la búsqueda en profundidad para obtener un árbol recubridor del grafo G mostrado en la Figura 6

Solución: Los pasos realizados con la búsqueda en profundidad para generar un árbol recubridor de G se muestran en la Figura 7. Elegimos, de modo arbitrario, un vértice inicial f . Construimos un camino, tan largo como sea posible, añadiendo sucesivamente aristas que son incidentes con vértices que no están en el camino. El resultado es el camino f, g, h, k, j (nótese que se pueden construir otros caminos). Seguidamente, retrocedemos hasta k . No hay ningún camino que comience en k que contenga vértices no visitados, así que retrocedemos hasta h . Construimos el camino h, i . Volvemos hacia atrás hasta f . Desde f construimos el camino f, d, e, c, a . Luego retrocedemos al vértice c y construimos c, b . Con eso obtenemos el árbol generador.

Las aristas de un grafo seleccionadas en la búsqueda en profundidad se llaman **aristas del árbol**. Todas las demás aristas del grafo tienen que conectar un vértice con un antecesor o un descendiente de este vértice en el árbol. A estas aristas se les llama aristas de retroceso.

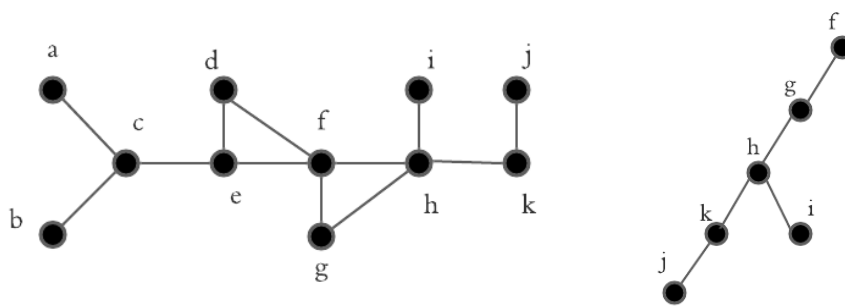
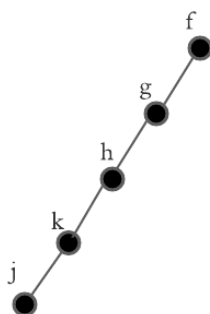


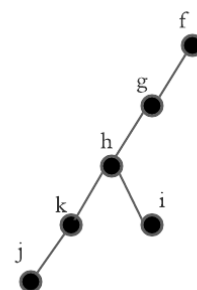
Figura 6: El grafo G .



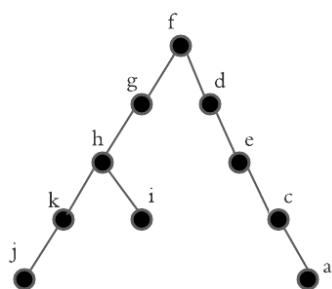
a



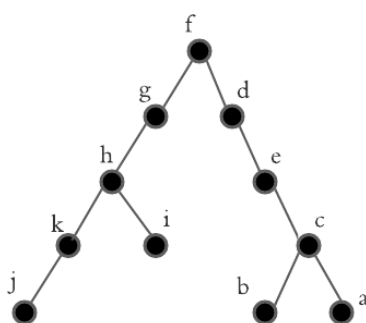
b



c



d



e

Figura 7: Una búsqueda en profundidad del grafo G .

EJEMPLO: En la figura 8 se destaca con trazo grueso las aristas del árbol detectadas por la búsqueda en profundidad que comienza en el vértice f . Las aristas de retroceso son (e, f) y (f, h) y se muestran con trazo más fino.

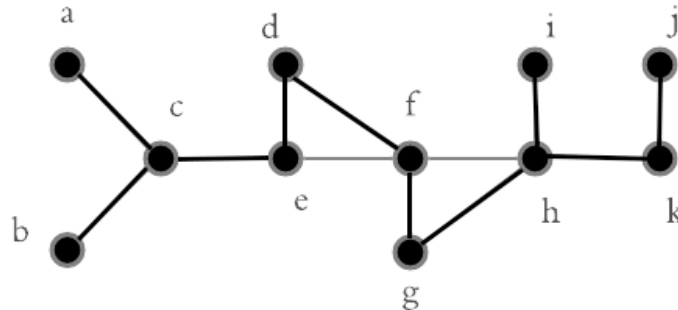


Figura: Las aristas del árbol y las aristas de retroceso de la búsqueda en profundidad del grafo G .

BÚSQUEDA DE ANCHURA

También podemos generar un árbol recubridor de un grafo simple mediante la búsqueda en **anchura** o por **niveles**. Nuevamente, se construye un árbol con raíz, y el grafo no dirigido subyacente es el árbol generador. Elegimos un vértice arbitrario como raíz. Luego añadimos todas las aristas incidentes en ese vértice. Los nuevos vértices añadidos en esa fase forman los vértices del nivel 1 del árbol generador. Los ordenamos con un orden cualquiera. Para cada vértice del nivel 1, visitados en orden, añadimos todos los vértices incidentes con él, siempre que no formen un ciclo. Ordenamos los hijos de los vértices del nivel 1 con un orden cualquiera, generando así los vértices de nivel 2 del árbol. Seguimos el mismo procedimiento hasta que se hayan añadido todos los vértices al árbol.

Este procedimiento termina, ya que los grafos tienen un número finito de aristas. Lo que se genera es un árbol recubridor porque hemos obtenido un subgrafo que es un árbol que contiene a todos los vértices del grafo. En el siguiente ejemplo se muestra el procedimiento de búsqueda en anchura.

EJEMPLO: Obtén un árbol generador para el grafo de la figura 9 utilizando la búsqueda de anchura.

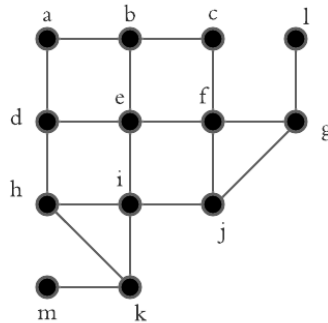


Figura 9. Un grafo G

Solución: Los pasos del algoritmo de búsqueda en anchura se muestran en la Figura 10. Elegimos un vértice e como raíz. Luego añadimos las aristas incidentes con todos los vértices adyacentes a e , esto es, las aristas eb , ed , ef y ei . Los cuatro vértices extremos de las anteriores aristas se añaden al nivel 1 del árbol. Después, se añaden las aristas que conectan los vértices de nivel 1 con vértices que aún no están en el árbol. Por tanto, se añaden las aristas ba , bc , dh , fj , fg , ik . Los nuevos vértices a , c , h , j , g y k forman el nivel 2. Seguidamente, añadimos las aristas que unen estos vértices con vértices adyacentes que no están aún en el árbol, Estas aristas son gl y km .

Describimos la búsqueda en anchura en pseudocódigo en el Algoritmo 2. En este algoritmo suponemos que los vértices del grafo conexo G están ordenados según las etiquetas, v_1, v_2, \dots, v_n .

En el algoritmo utilizamos el término «proceso» para describir el procedimiento de añadir nuevos vértices y las correspondientes aristas al árbol adyacente al vértice procesado mientras no se generen ciclos.

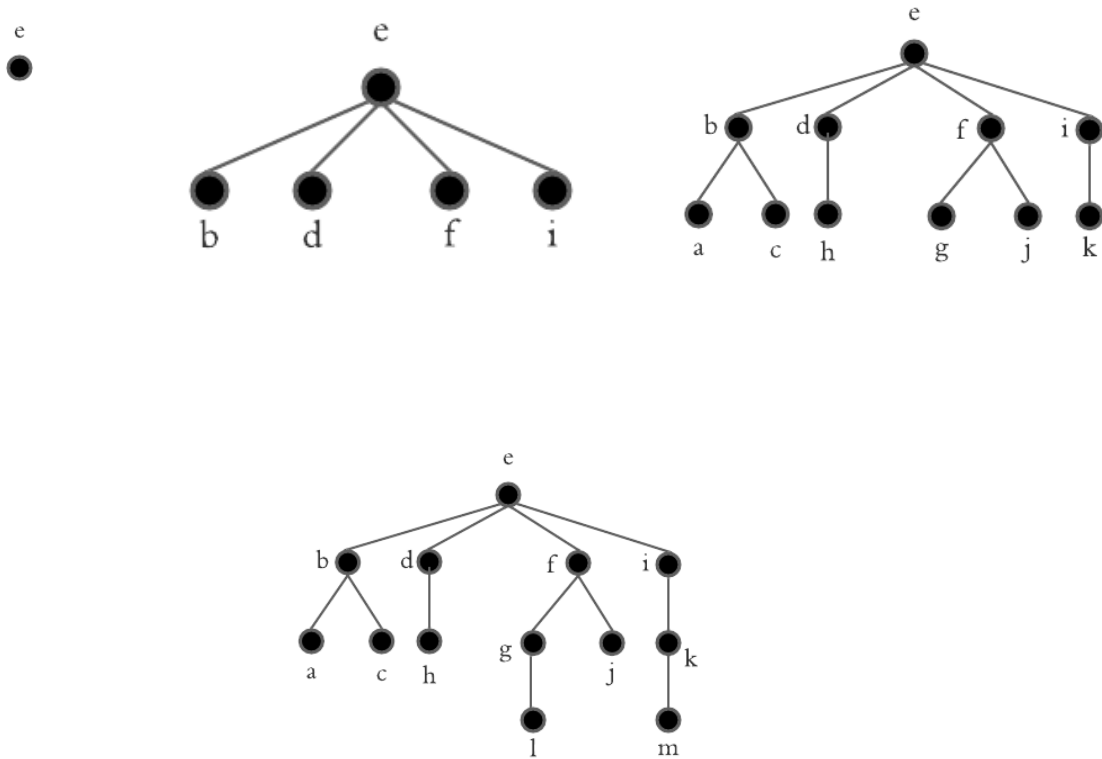


Figura 10. Una búsqueda en anchura de G .

ALGORITMO 2 Búsqueda en anchura

procedure $BA(G: \text{grafo conexo de vértices } v_1, v_2, \dots, v_n)$

$T :=$ árbol que consta de un único vértice v_1

$L :=$ lista vacía

poner v_1 en la lista L de los vértices no procesados

while L no esté vacía

begin

borrar el primer vértice v de L

for cada vértice w adyacente a v

if w no está en L y no está en T **then**

begin

añadir el vértice w al final de la lista L

añadir el vértice w y la arista $\{v, w\}$ a T

end

end

APLICACIONES DE LAS VUELTAS

Hay problemas que se pueden resolver realizando una búsqueda exhaustiva de todas las soluciones posibles. Un método de búsqueda sistemática de una solución es utilizar un árbol de decisión, donde cada vértice interno representa una decisión y cada hoja una posible solución. Para encontrar una solución mediante la vuelta atrás, primeramente, construimos una secuencia de decisiones intentando alcanzar una solución mientras sea posible. Esta serie de decisiones se puede representar por un camino en el árbol de decisión. Una vez que se sabe que no existe solución en las hojas que son descendientes del vértice que se está procesando, se vuelve atrás hacia el padre del vértice en curso y se trabaja en la búsqueda de soluciones realizando otra serie de decisiones. El procedimiento continúa hasta que se consigue una decisión o se establece que tal solución no existe. Los siguientes ejemplos ilustran la utilidad de la vuelta atrás.

EJEMPLO: Coloración de grafos ¿Cómo se puede utilizar la vuelta atrás para decidir cuándo se puede colorear un grafo usando n colores?

Solución: Podemos resolver este problema utilizando la vuelta atrás del modo siguiente. Primero elegimos un vértice, pongamos a , y le asignamos el color 1. Luego seleccionamos otro vértice b , y si b no es adyacente a a , le asignamos el color 1. En caso contrario, le asignamos a b el color 2. Continuamos con el tercer vértice c . Utilizamos el color 1 como primera asignación, si fuera posible. En caso contrario, seguimos con el color 2 como posible asignación para c . Sólo si no se pudiera utilizar ninguno de los colores anteriores, se asignará a c el color 3. Continuamos este proceso, mientras podamos, como estrategia para asignar los colores a n los vértices, utilizando siempre el primer color disponible en la lista. Si se alcanza algún vértice que no pueda ser coloreado con ninguno de los n colores, volvemos hacia atrás a la última asignación realizada y cambiamos el último vértice coloreado, si fuera posible, utilizando el siguiente color disponible en la lista. Si no fuera posible cambiar este color, retrocederíamos a las asignaciones previas, una en cada ocasión, hasta que sea posible cambiar la coloración de un vértice. Luego continuamos la asignación de los restantes vértices siempre que podamos. Si existe una coloración del grafo utilizando n colores, el procedimiento de vuelta atrás la genera. (Desgraciadamente, este procedimiento es en general muy poco eficiente).

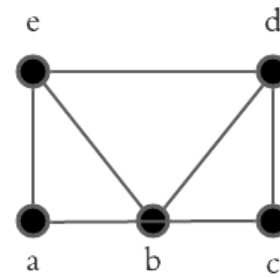
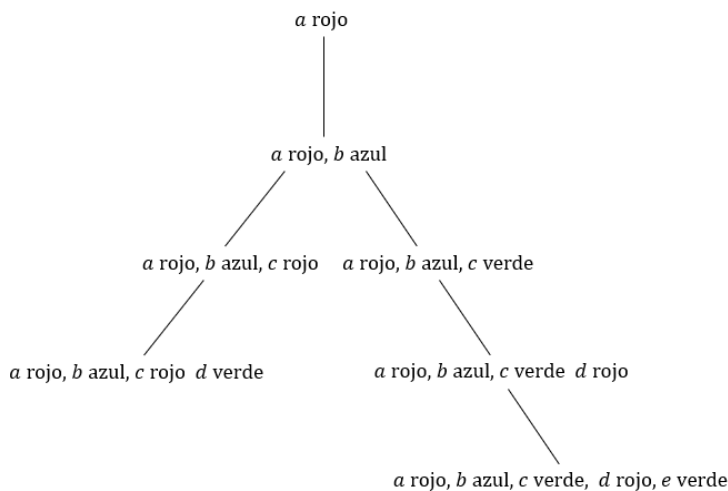


Figura 11: Coloración de un grafo utilizando vuelta atrás.

EJEMPLO: El problema de las n reinas: El problema de las n reinas El problema consiste en determinar cómo se pueden colocar n reinas en un tablero de ajedrez $n \times n$ de manera que dos reinas cualesquiera no puedan amenazarse mutuamente. ¿Cómo se puede utilizar la vuelta atrás para resolver el problema de las n reinas?

Solución: Para resolver este problema debemos encontrar a posiciones sobre un tablero $n \times n$ de manera que dos posiciones cualesquiera no estén en la misma fila o en la misma columna o en la misma diagonal [una diagonal consiste en todas las posiciones (i, j) , con $i + j = m$ para algún $i - j = m$, para algún m]. Utilizaremos la vuelta atrás para resolver el problema. Comenzamos con un tablero vacío. En el estado $k + 1$ intentamos colocar una reina más en la columna $(k + 1)$ -ésima del tablero, donde ya se han colocado k reinas en las k primeras columnas. Examinamos los cuadrados en la columna $(k + 1)$ -ésima comenzando con el cuadrado de la primera fila. buscando una posición en la que colocar la reina de manera que ésta no figure en la misma fila o diagonal que otra de las reinas ya colocadas (ya sabemos que no está en la misma columna). Si es imposible encontrar una posición de la columna $(k + 1)$ -ésima en la que colocar la reina, hacemos retroceder la posición de la reina a la columna k -ésima y colocamos esta reina en el siguiente lugar disponible, si tal lugar existe. Si no existiera, retrocederíamos una vez más.

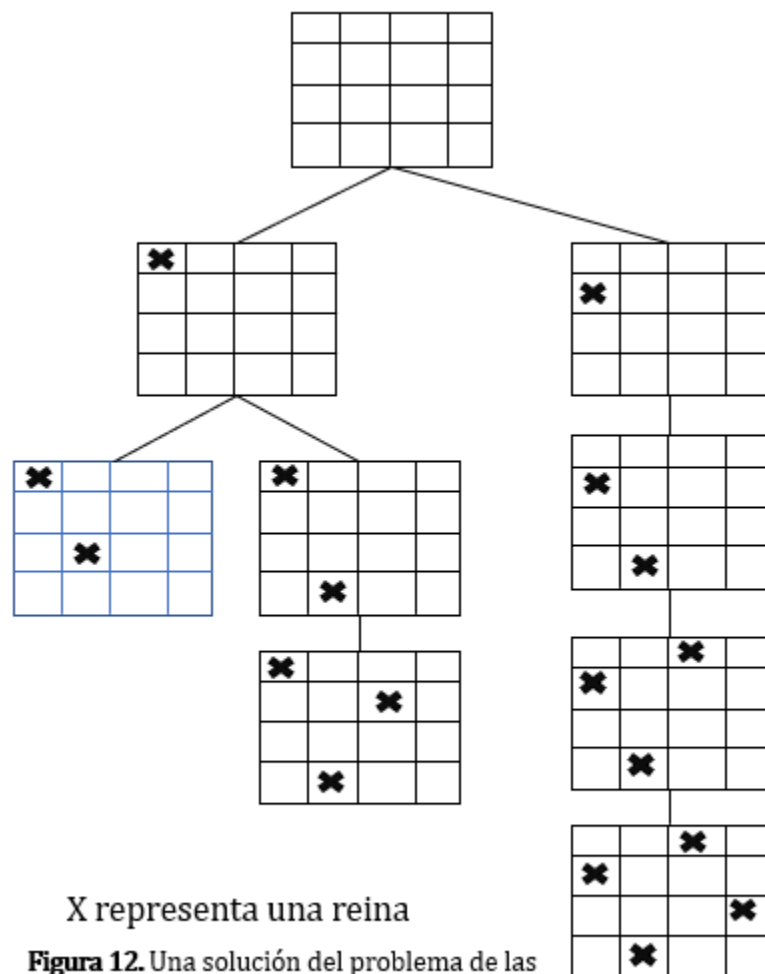


Figura 12. Una solución del problema de las cuatro reinas generada con vuelta atrás.

En particular, la Figura 12 muestra una solución mediante la vuelta atrás para el problema de cuatro reinas. En esta solución, colocamos una reina en la primera fila y columna, Luego colocamos una reina en la tercera fila de la segunda columna. Sin embargo, esta elección hace que resulte imposible colocar una reina en la tercera columna, Así que retrocedemos y colocamos la segunda reina en la cuarta fila de la segunda columna. Una vez hecho esto, podemos poner una reina en la segunda fila de la tercera columna. Pero así no hay modo de añadir la cuarta reina. Esto prueba que no hay solución colocando la primera reina en la primera fila y columna. Retrocedemos al tablero vacío y ponemos una reina en la segunda fila de la primera columna. Esto lleva a una solución como se muestra en la Figura 12.

EJEMPLO: Sumas de subconjuntos Consideremos el problema siguiente. Dado un conjunto de enteros positivos x_1, x_2, \dots, x_n , obtener un subconjunto de este conjunto de enteros de modo que la suma de sus elementos sea M . ¿Cómo podemos utilizar la vuelta atrás para resolver este problema?

Solución: Comenzamos con una suma cuyo conjunto de sumandos es vacío, Construimos la suma añadiendo sumandos de manera sucesiva. Un entero de la sucesión se incluye si la suma permanece menor que M cuando este entero se añade al valor de la suma acumulada. Si se obtiene una suma tal que al añadir cualquier término se supera el valor M , entonces se vuelve atrás eliminando el último sumando considerado.

La Figura 13 muestra una solución con vuelta atrás del problema de obtener un subconjunto de $\{31, 27, 15, 11, 7, 5\}$ suma 39.

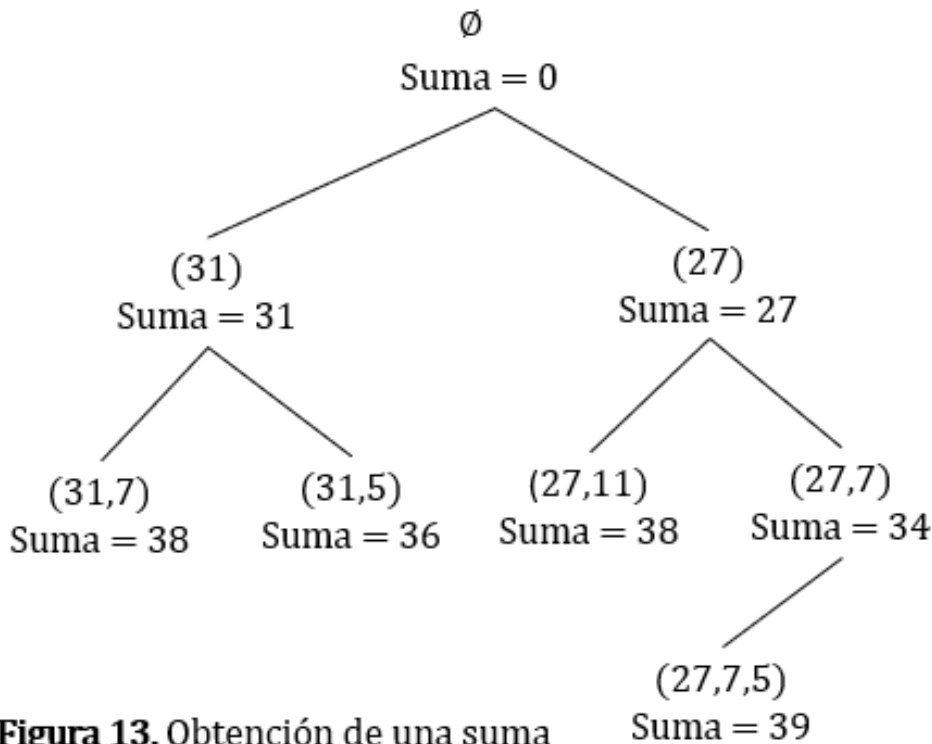


Figura 13. Obtención de una suma igual a 39 utilizando vuelta atrás

BÚSQUEDA EN PROFUNDIDAD EN GRAFOS DIRIGIDOS

Los algoritmos de búsqueda en profundidad y búsqueda en anchura se pueden modificar para que puedan aplicarse a grafos dirigidos. Sin embargo, el resultado final no será necesariamente un árbol generador, sino un bosque generador. En ambos algoritmos podemos añadir una arista sólo cuando ésta se aleja del vértice en curso hacia otro que no se ha añadido aún. Si en un paso de alguno de los dos algoritmos vemos que tal arista no existe, el siguiente vértice añadido por el algoritmo pasa a ser la raíz de un nuevo árbol en el bosque generador. Esto se ilustra en el siguiente ejemplo.

EJEMPLO: ¿Cuál es la salida del algoritmo de búsqueda en profundidad para el grafo dado en la Figura 14(a)?

Solución: Comenzamos con la búsqueda en profundidad en el vértice a y añadimos los vértices b, c , y g junto con las correspondientes aristas hasta que quedamos bloqueados. Retrocedemos a c , pero todavía seguimos bloqueados; así que retrocedemos hasta b , donde añadimos los vértices f y e y las correspondientes aristas. La vuelta atrás nos hace regresar al vértice a . Así, comenzamos un nuevo árbol en d y añadimos los vértices h, l, k y j y las correspondientes aristas. Retrocedemos hasta k , luego hasta j , luego hasta i y volvemos a d . Finalmente, comenzamos un nuevo árbol en e , completando la búsqueda en profundidad. El resultado final se muestra en la Figura 14(b).

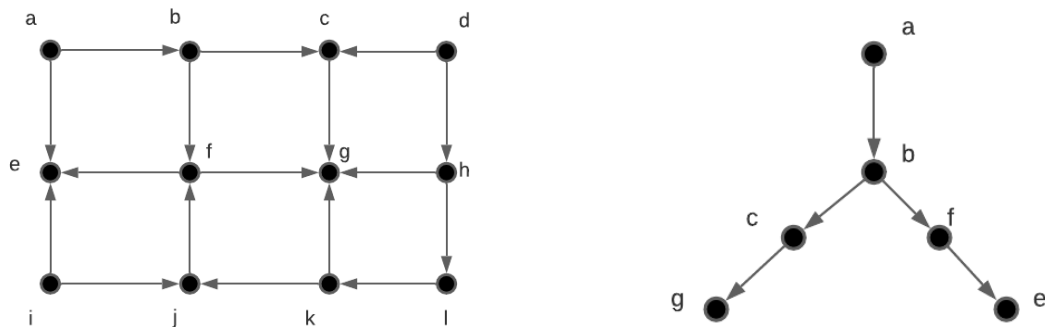




Figura 14: Una búsqueda en profundidad en un grafo dirigido

ÁRBOL GENERADOR MÍNIMO

Una compañía planea construir una red de comunicaciones para conectar sus cinco centros informáticos. Cualquier pareja de estos centros debe estar enlazada mediante una línea telefónica alquilada. ¿Qué enlaces deberían realizarse para asegurar que hay un camino entre cualquier par de centros de modo que el coste total de la red sea el menor posible? Podemos modelar este problema usando el grafo ponderado de la Figura 1, donde los vértices representan los centros informáticos, las aristas representan posibles líneas de comunicaciones y los pesos de las aristas son los costes mensuales del alquiler de las líneas asociadas a las aristas. Podemos resolver este problema encontrando un árbol generador de manera que la suma de pesos de las aristas del árbol sea mínima. Este árbol generador se llama árbol generador mínimo.

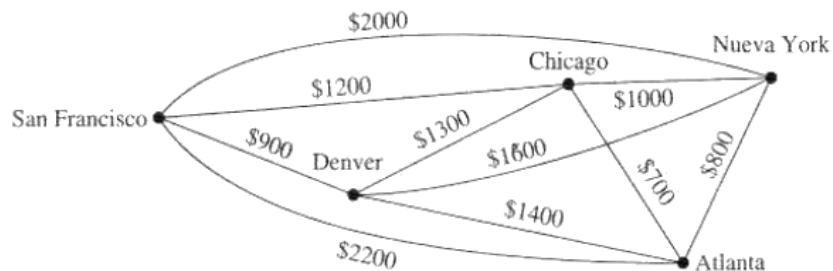


Figura 1. Un grafo ponderado que muestra el coste mensual del alquiler de las líneas de una red de ordenadores.

ALGORITMOS PARA ÁRBOLES GENERADORES MÍNIMOS

Definición: Un árbol generador mínimo de un grafo ponderado es un árbol generador tal que la suma de los pesos de sus aristas es la mínima posible de entre todos los árboles generadores.

Algoritmo de Prim.

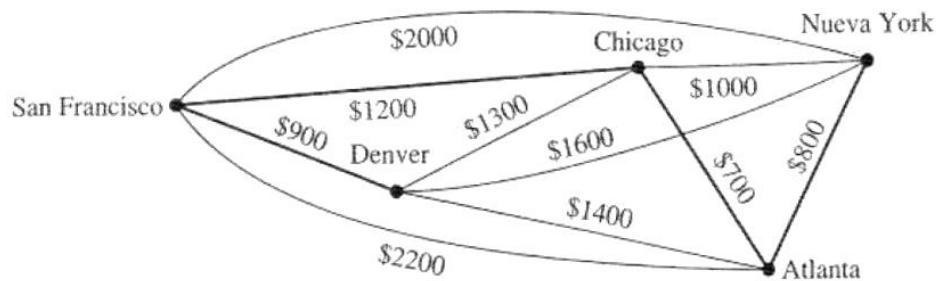
El primer algoritmo que discutiremos fue propuesto por Robert Prim en 1957, aunque las ideas básicas son anteriores. Para ejecutar el algoritmo de Prim, comenzamos eligiendo cualquier arista de peso mínimo y la seleccionamos para el árbol. Añadimos sucesivamente aristas al árbol de entre las de peso mínimo que sean incidentes con un vértice que ya está en el árbol y que no formen un ciclo con otras aristas del árbol. Paramos cuando hayamos añadido $n-1$ aristas.

ALGORITMO 1 Algoritmo de Prim

```
procedure Prim( $G$ : grafo ponderado, conexo y no dirigido de  $n$  vértices)
 $T :=$  una arista de peso mínimo
for  $i := 1$  to  $n - 2$ 
begin
     $e :=$  una arista de peso mínimo incidente con un vértice de  $T$  que no forme un
        ciclo en  $T$  cuando se añada
     $T := T$  con  $e$  añadida
end { $T$  es un árbol generador mínimo de  $G$ }
```

Ejemplo: usa el algoritmo de Prim para diseñar una red de comunicaciones de coste mínimo que conecte todos los ordenadores en el grafo de la Figura 1

Solución:



Elección	Arista	Coste
1	{Chicago, Atlanta}	\$ 700
2	{Atlanta, Nueva York}	\$ 800
3	{Chicago, San Francisco}	\$1200
4	{San Francisco, Denver}	\$ 900
Total:		\$3600

Algoritmo de Kruskal

El segundo algoritmo que discutiremos fue descubierto por Joseph Kruskal en 1956, aunque las ideas básicas que utiliza fueron descritas con antelación. Para ejecutar el algoritmo de Kruskal, se elige una arista del grafo de entre las de menor peso.

Se añaden paulatinamente las aristas con menor peso siempre que éstas no formen un ciclo con las otras ya incorporadas. El proceso termina cuando se han seleccionado $n-1$ aristas.

ALGORITMO 2 Algoritmo de Kruskal

```

procedure Kruskal( $G$ : grafo ponderado, conexo y no dirigido de  $n$  vértices)
 $T$  := grafo vacío
for  $i$  := 1 to  $n - 1$ 
begin
     $e$  := una arista de peso mínimo de  $G$  que no forme un
        ciclo cuando se añada a  $T$ 
     $T$  :=  $T$  con  $e$  añadida
end { $T$  es un árbol generador mínimo de  $G$ }

```


Ejemplo: Utiliza el algoritmo de Kruskal para obtener un árbol generador mínimo en el grafo ponderado de la Figura 2.

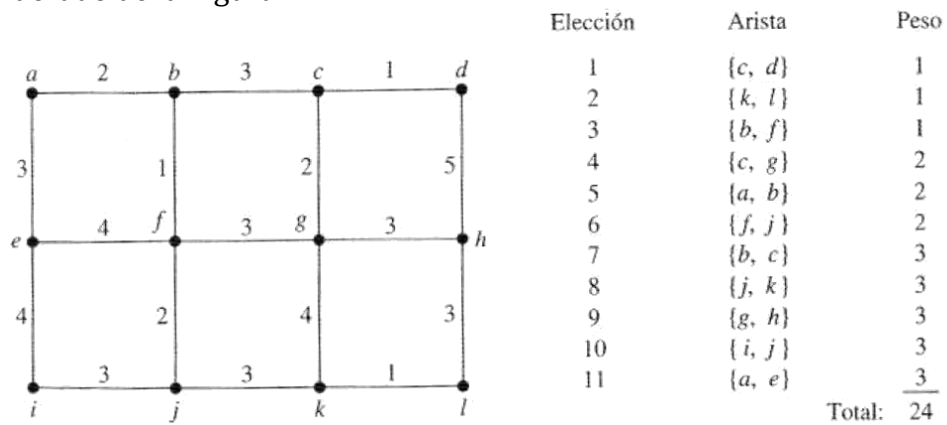


Figura 2.

PREGUNTAS DEL LIBRO

1. ¿Qué es la ingeniería del software? Describa la diferencia entre los ingenieros clásicos y los ingenieros de software.

R// La ingeniería de software es una disciplina que se encarga del diseño, desarrollo y mantenimiento de sistemas de software mediante la aplicación de principios de ingeniería.

La cual sigue metodologías para determinar requisitos, diseño, desarrollo, implementación y prueba del software, centrándose en las necesidades del cliente.

Diferencias entre el ingeniero clásico y uno de software.

Ingeniero clásico	Ingeniero de software
<ul style="list-style-type: none">• Se centra en producir un diseño del producto.• Analiza el diseño para determinar su corrección.• utilizan las matemáticas en su análisis para confirmar que las especificaciones del diseño se cumplen.	<ul style="list-style-type: none">• Se centra en establecer con precisión los requisitos que el producto de software debe cumplir y de esta manera producir los diseños y posteriormente la implementación que los satisfagan.• valida la corrección de su diseño y verifica la implementación del software a través de pruebas para asegurar que cumpla con los requisitos establecidos.• También emplean matemáticas para la resolución de problemas, sin embargo, el uso de estas dependerá del tipo de proyecto, su aplicación y los cálculos asociados necesarios.

2. **Describa la "crisis del software" de finales de la década de 1960 que dio lugar a la primera conferencia sobre ingeniería del software en 1968.**

R// Esta crisis de software principalmente se dio por los excesos de presupuestos y calendarios de los proyectos, además de los problemas con la calidad y fiabilidad del software suministrado.

Por ende, a raíz de esto, la conferencia dio lugar al nacimiento de la ingeniería de software como una disciplina independiente.

3. **Discute sobre el Stadish Research Report y el nivel de éxito de proyectos de Tecnologías de Información (IT) en el día de hoy. ¿Desde tu perspectiva, existe una crisis en la Ingeniería de Software en la actualidad? Establece las razones de tu respuesta.**

Respuesta: En la actualidad, la agilidad para entregar software de calidad es primordial, debido a que los estándares con respecto a otros tiempos han subido, debido al avance tecnológico con el que contamos. Sin embargo, hay escenarios donde no se logra un estándar mínimo para la entrega a este: Un ejemplo son las inteligencias artificiales, que, con el incremento tan drástico del uso de estas, los desarrolladores tenían la necesidad de competir por quién perfeccionaba la eficiencia de estas, lo cual conlleva a inseguridades interactuando con ellas, tanto así que las investigaciones para las AIs, fueron detenidas por un lapso de tiempo.

4. **Discute qué rol deberían de tener las matemáticas en la Ingeniería de Software actual.**

Respuesta: Se concluye que las matemáticas cumplen un rol clave en la Ingeniería de Software, ya que con estas se pueden resolver problemas que surjan a la hora del planteamiento de los requerimientos.

Con el reciente "Boom" de las inteligencias artificiales también es muy necesario el manejo de las matemáticas para poder desarrollar buenas redes neuronales, también son importantes en el ámbito de la seguridad del software al proporcionar un enfoque riguroso, permitir la criptografía y el modelado formal, facilitar el análisis de riesgos y la evaluación de medidas de seguridad, y establecer mecanismos de autenticación y autorización confiables.

5. Describe los ciclos de vida de Cascada y espiral. ¿Cuáles son sus diferencias y similitudes?

Respuesta: Cascada: El ciclo de vida de software en cascada realmente es muy intuitivo, pues se tiene en cuenta de que al ser; como su nombre lo indica “cascada”, no es posible ir en sentido contrario, es decir, que no se continua a la siguiente fase hasta que la fase actual este completamente terminada y con resultados satisfactorios.

La metodología cascada según el libro se podrías dividir en dos sectores que se conocen como “V” donde la parte o brazo izquierdo de la v, es decir, “” donde el paso a paso seguir es: requerimientos, especificaciones, diseño, código. El lado derecho “/” el paso a paso es: unificación para testeo, integración de testeo, sistema de testeo, aceptación del testeo.

Lo anteriormente mencionado no es fijo, puede variar dependiendo del proyecto

Espiral: el ciclo de vida de software espiral permite una flexibilidad mayor a la de cascada pues permite trabajar sobre proyectos donde los requerimientos no están realmente definidos.

Cada espiral es una mejora de la anterior, cabe resaltar que cada espiral no se desecha, estas sirven y son reutilizables, donde en cada espiral el cliente da retroalimentación sobre lo que se ha avanzado, este proceso de repite hasta haber terminado por completo.

Diferencias: se pueden diferenciar principalmente por la flexibilidad de trabajo que disponen, es decir, la metodología cascada es muy estricta con los requerimientos, pues si no están claros no se puede trabajar y ni siquiera se puede avanzar a las siguientes fases.

La metodología espiral es cíclica y/o evolutiva con entregas incrementales mientras que la metodología cascada es totalmente lineal.

6. Discute los aportes de Floyd y Hoare.

Robert Floyd	C.A.R Hoare
<ul style="list-style-type: none">• Contribuciones para “Theory of parsing”• Demostró que las matemáticas pueden ser usadas para la verificación de programas.• Introduce el concepto de “assertions” que proveían una forma de verificar la exactitud de los programas.• Su técnica demostró que un programa de computador es	<ul style="list-style-type: none">• Desarrolló el algoritmo de ordenación Quicksort.• Creador del lenguaje de programación CSP (Communicating Sequential Processes)• Lógica de Hoare: Notación formal que especifica el comportamiento de un programa• Verificación formal: Trabajó en

<p>una secuencia de afirmaciones lógicas.</p> <ul style="list-style-type: none"> • Publicó el “Assigning Meaning to Programs” el cuál influyó el trabajo de Hoare en precondiciones y postcondiciones. • 	<p>el desarrollo de lenguajes específicos y herramientas para ayudar a la verificación de propiedades de programas (corrección funcional y ausencia de errores)</p>
--	---

7. Explica la diferencia entre *partial correctness* y *total correctness*:

Respuesta: La corrección parcial se enfoca en garantizar que un programa cumpla con las precondiciones y postcondiciones especificadas, sin abordar la finalización del programa. Mientras tanto, la corrección total va más allá y exige que el programa cumpla con las precondiciones y postcondiciones, así como que termine su ejecución en un tiempo finito.

8. ¿Qué son los métodos formales?

Los métodos formales son un conjunto de técnicas y herramientas utilizadas en la ingeniería de software para la especificación, diseño y verificación de sistemas de software. Estos métodos se basan en fundamentos matemáticos y lógicos para garantizar la corrección y confiabilidad de los sistemas.

Los métodos formales constan de los siguientes elementos principales:

Especificación formal: Implica describir el comportamiento deseado del sistema de manera precisa y no ambigua utilizando lenguajes formales. Estos lenguajes, como la lógica de predicados o el cálculo de predicados, permiten especificar propiedades, restricciones y requisitos del sistema.

Modelado: Consiste en representar el sistema de software utilizando técnicas formales como el álgebra de procesos, autómatas o lenguajes de especificación. El modelado formal permite capturar la estructura y el comportamiento del sistema de manera rigurosa.

Verificación formal: Implica el uso de herramientas y técnicas para demostrar matemáticamente que un sistema cumple con ciertas propiedades o requisitos especificados. Esto se logra mediante la aplicación de reglas de inferencia y técnicas de prueba formales. La verificación formal ayuda a identificar errores, inconsistencias o problemas de diseño en el sistema antes de su implementación.

Refinamiento y desarrollo incremental: Los métodos formales promueven un enfoque iterativo en el desarrollo de sistemas, donde se realiza un refinamiento gradual del diseño y la especificación. Esto implica comenzar con una especificación abstracta y, mediante pasos sucesivos de refinamiento, llegar a una especificación y diseño más detallados y concretos.

Pruebas formales: Además de la verificación formal, los métodos formales también involucran la realización de pruebas formales. Estas pruebas se basan en razonamientos lógicos y matemáticos para demostrar la corrección de componentes o propiedades específicas del sistema.

En general, los métodos formales buscan aplicar rigor matemático y lógico en todas las etapas del desarrollo de software para garantizar la calidad y la corrección del sistema. Estas técnicas son especialmente útiles en áreas críticas como la aviación, la industria nuclear y los sistemas de control, donde se requiere una alta confiabilidad y seguridad.

9. Explica la diferencia entre partial correctness y total correctness:

La corrección parcial se enfoca en garantizar que un programa cumpla con las precondiciones y postcondiciones especificadas, sin abordar la finalización del programa. Mientras tanto, la corrección total va más allá y exige que el programa cumpla con las precondiciones y postcondiciones, así como que termine su ejecución en un tiempo finito.

10. Discute como las inspecciones y las pruebas de software asisten a la entrega de software de alta calidad.

Las inspecciones y las pruebas de software son dos actividades importantes en el proceso de desarrollo de software que ayudan a garantizar la entrega de software de alta calidad. A continuación, se explican cómo estas actividades asisten en ese objetivo:

1. **Inspecciones de software:** Las inspecciones son revisiones sistemáticas y estructuradas del software, realizadas por un equipo de personas, con el propósito de identificar y corregir errores, problemas de diseño, deficiencias en la documentación y otras fallas. Estas inspecciones se centran en revisar tanto el código fuente como la documentación asociada al software.

- **Identificación temprana de problemas:** Las inspecciones permiten encontrar errores y problemas en etapas tempranas del proceso de desarrollo, lo que facilita su corrección antes de que se propaguen y se conviertan en problemas más costosos y difíciles de solucionar.

- Mejora de la calidad del software: Al detectar y corregir errores, deficiencias y problemas de diseño, las inspecciones contribuyen a mejorar la calidad del software, aumentando la confiabilidad y la eficiencia del sistema.

- Transferencia de conocimiento: Durante las inspecciones, los miembros del equipo pueden compartir conocimientos y buenas prácticas, lo que promueve la colaboración y el aprendizaje entre los miembros del equipo y mejora la calidad del trabajo realizado.

2. Pruebas de software: Las pruebas de software implican la ejecución de programas o sistemas para identificar errores, verificar su funcionamiento y evaluar si cumplen con los requisitos y especificaciones establecidos. Las pruebas ayudan a descubrir defectos y a evaluar la calidad general del software.

- Detección de errores y fallas: Las pruebas permiten identificar errores y defectos en el software al verificar su comportamiento bajo diferentes condiciones y escenarios de uso. Esto ayuda a garantizar que el software funcione correctamente y cumpla con los requisitos establecidos.

- Mejora de la estabilidad y confiabilidad: Las pruebas ayudan a mejorar la estabilidad y la confiabilidad del software al identificar y corregir problemas de rendimiento, errores de programación y situaciones límite que puedan causar fallas o mal funcionamiento del sistema.

- Validación de requisitos: Las pruebas permiten verificar si el software cumple con los requisitos establecidos, tanto funcionales como no funcionales. Esto asegura que el software entregado satisfaga las expectativas y necesidades de los usuarios.

En conjunto, las inspecciones y las pruebas de software desempeñan un papel crucial en la entrega de software de alta calidad, al ayudar a identificar y corregir errores, mejorar el diseño y garantizar que el software cumpla con los requisitos y expectativas establecidos. Estas actividades contribuyen a reducir los riesgos y mejorar la confiabilidad y el rendimiento del software en producción.

BIBLIOGRAFÍA:

- MATEMATICA DISCRETA Y SUS APLICACIONES (5a ED.). (2004, June 9). Casadellibro. <https://www.casadellibro.com.co/libro-matematica-discreta-y-sus-aplicaciones-5-ed/9788448140731/957996>
- AULA DE MATEMÁTICA. (2020, August 23). GRAFOS (tipos de grafos y terminología básica) [Video]. YouTube. <https://www.youtube.com/watch?v=ER4H26id3PE>
- Math Plus. (2016, October 5). Grafos Caminos e isomorfismo y caminos entre un par de vertices [Video]. YouTube. <https://www.youtube.com/watch?v=KDsuvkSqfLM>
- Carlos Delgado. (2021, February 2). Matemáticas discretas II Clase 8 4 Isomorfismos en grafos [Video]. YouTube. <https://www.youtube.com/watch?v=ApT-hYuEh7k>
- Math Plus. (2016a, September 27). Grafos Teorema del apretón de manos [Video]. YouTube. https://www.youtube.com/watch?v=RCcPFGqL_jg
- AULA DE MATEMÁTICA. (2020b, August 23). GRAFOS (tipos de grafos y terminología básica) [Video]. YouTube. <https://www.youtube.com/watch?v=ER4H26id3PE>