



ALUMNO: Pablo Naelson Martinez Ramirez

NÚMEROS DE CONTROL: 24130865

PROFESOR: M.C. SERGIO VICTOR GUARDADO GUZMÁN

CARRERA: ING. EN SISTEMAS COMPUTACIONALES

MATERIA: MATEMÁTICAS DISCRETAS

INSTITUTO: TECNM CAMPUS LAGUNA

Contenido

1 SISTEMAS NUMÉRICOS	4
1.1 Sistemas numéricos (Binario, Octal, Decimal, Hexadecimal)	4
1.2 Conversiones entre sistemas numéricos	6
1.3 Operaciones básicas (Suma, Resta, Multiplicación y División)	10
1.4 Aplicación de los sistemas numéricos en la computación	16
2 CONJUNTOS Y RELACIONES	20
2.2 Operaciones con conjuntos	21
2.3 Propiedades y aplicaciones de los conjuntos	23
2.4 Conceptos básicos: producto cartesiano y relación binaria	24
2.5 Representación de las relaciones Para representar las relaciones binarias podemos utilizar dos tipos de gráficos:	25
2.6 Propiedades de las relaciones	25
2.7 Relaciones de equivalencia	27
2.8 Funciones	28
2.9 Aplicaciones de las relaciones y las funciones en la computación.	31
3. LÓGICA MATEMÁTICA	35
3.1 Lógica proposicional	35
3.1.1 Proposiciones simples y compuestas	36
3.1.2 Tablas de verdad	40
3.1.3 Tautologías, contradicción y contingencia	42
3.1.4 Equivalencias lógicas	43
3.1.5 Reglas de inferencia	46
3.1.6 Argumentos válidos y no validos	47
3.1.7 Demostración formal	49
3.2 Lógica de predicados	50
3.2.1 Cuantificadores	51
3.2.2 Representación y evaluación de predicados	52
3.3 Algebra declarativa	53
3.4 Inducción matemática	54

3.5 Aplicaciones de la lógica matemática en la computación	55
4. ÁLGEBRA BOOLEANA	56
4.1 - Teoremas y postulados.	56
4.2 Optimización de expresiones booleanas.	59
4.3 Aplicación del algebra booleana (Compuertas lógicas).....	62
4.3.1 Mini y maxi términos.	67
4.3.2 Representación de expresiones booleanas con circuitos lógicos.	72
5. TEORÍA DE GRAFOS	76
5.1 Elementos, características y componentes de los grafos.	76
5.1.1 TIPOS DE GRAFOS.....	77
5.2 Representación de los grafos.	80
5.2.1 Matemática.....	81
5.2.2 Computacional	83
5.3 Algoritmos de recorrido y búsqueda.	84
5.3.1 El camino más corto.....	86
5.3.3 En profundidad	88
6. ÁRBOLES Y REDES	89
6.1 Árboles.	89
6.1.1 Componentes y propiedades	90
6.1.2 Clasificación por altura y número de nodos	93
6.2. Árboles con peso	94
6.2.1 Recorrido de un árbol	96
6.3 Redes.....	97
6.3.1 Teorema de flujo máximo	98
6.3.2 Teorema de flujo mínimo	98
6.3.3 Pareos y redes de Petri	99

1 SISTEMAS NUMÉRICOS

Un sistema numérico tiene como objetivo el permitir el conteo de los elementos de un conjunto. El sistema se conforma por n unidades en orden sucesivo que aumentan de n en n . De acuerdo a n se define el número de unidades que se necesitan para pasar de un orden a otro.

Una de las condiciones para utilizar algún sistema numérico es el que permita realizar operaciones básicas sobre el conjunto N de una forma sencilla.

Otra condición es que por cada elemento $n \in N$ debe corresponderle un símbolo escrito.

Cabe mencionar, que un elemento en el conjunto N siempre podrá tener dos clases de valores:

El valor individual, Ej. En 58, el 8 representa 8 unidades en el sistema decimal.

El valor que le corresponde de acuerdo a su posición, Ej. En 80, el 8 representa 8 decenas.

1.1 Sistemas numéricos (Binario, Octal, Decimal, Hexadecimal)

Sistema Numérico de Base 10

Los sistemas numéricos están compuestos por símbolos y por las normas utilizadas para interpretar estos símbolos. El sistema numérico que se usa más a menudo es el sistema numérico decimal, o de Base 10. El sistema numérico de Base 10 usa diez símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9. Estos símbolos se pueden combinar para representar todos los valores numéricos posibles.

Ejemplo:

$$2134 = 2134$$

Hay un 4 en la posición correspondiente a las unidades, un 3 en la posición de las decenas, un 1 en la posición de las centenas y un 2 en la posición de los miles. Este ejemplo parece obvio cuando se usa el sistema numérico decimal. Es importante saber exactamente cómo funciona el sistema decimal, ya que este conocimiento permite entender los otros dos sistemas numéricos, el sistema numérico de Base 2 y el sistema numérico hexadecimal de Base 16. Estos sistemas usan los mismos métodos que el sistema decimal.

Sistema Numérico de Base 2

Los computadores reconocen y procesan datos utilizando el sistema numérico binario, o de Base 2. El sistema numérico binario usa sólo dos símbolos, 0 y 1 (ENCENDIDO/APAGADO), en lugar de los diez símbolos que se utilizan en el sistema numérico decimal.

Ejemplo:

$$101102 = 22$$

Sistema Numérico de Base 8

El inconveniente de la codificación binaria es que la representación de algunos números resulta muy larga. Por este motivo se utilizan otros sistemas de numeración que resulten más cómodos de escribir: el sistema octal y el sistema hexadecimal. Afortunadamente, resulta muy fácil convertir un número binario a octal o a hexadecimal.

En el sistema octal, usa ocho dígitos diferentes: 0, 1, 2, 3, 4, 5, 6 y 7.

Ejemplo:

$$\text{El número octal } 2738 = 149610$$

Sistema Numérico de Base 16 (Hexadecimal)

El sistema hexadecimal usa dieciséis símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. Se utilizan los caracteres A, B, C, D, E y F representando las cantidades

decimales 10, 11, 12, 13, 14 y 15 respectivamente, porque no hay dígitos mayores que 9 en el sistema decimal.

Ejemplo:

El número hexadecimal $1A3F_{16} = 6719_{10}$

1.2 Conversiones entre sistemas numéricos

Conversión de Decimal a Binario

Para la conversión de decimal a binario se emplean dos métodos.

Método 1 por divisiones sucesivas, el cual consiste en:

Se va dividiendo la cantidad decimal por 2, apuntando los residuos, hasta obtener un cociente cero. El último residuo obtenido es el bit más significativo (MSB) y el primero es el bit menos significativo (LSB).

Ejemplo

Convertir el número 15310 a binario.

El resultado en binario de 15310 es 10011001

Método 2:

Otra forma de obtener el número decimal a binario es realizar lo siguiente:

Convertir un número decimal al sistema binario es muy sencillo: basta con realizar divisiones sucesivas por 2 y escribir los restos obtenidos en cada división en orden inverso al que han sido obtenidos.

Por ejemplo, para convertir al sistema binario el número decimal 77 haremos una serie de divisiones que arrojarán los restos siguientes:

$$77 / 2 = 38 \text{ Resto: } 1$$

$$38 / 2 = 19 \text{ Resto: } 0$$

$$19 / 2 = 9 \text{ Resto: } 1$$

$$9 / 2 = 4 \text{ Resto: } 1$$

$$4 / 2 = 2 \text{ Resto: } 0$$

$$2 / 2 = 1 \text{ Resto: } 0$$

$$1 / 2 = 0 \text{ Resto: } 1$$

y, tomando los restos en orden inverso obtenemos la cifra binaria:

Decimal 77 = Binario 1001101

Conversión de un número decimal a octal

La conversión de un número decimal a octal se hace con la misma técnica que ya hemos utilizado en la conversión a binario, mediante divisiones sucesivas por 8 y colocando los restos obtenidos en orden inverso. Por ejemplo, para escribir en octal el número decimal 12210 tendremos que hacer las siguientes divisiones:

$$122 / 8 = 15 \quad \text{Resto: } 2$$

$$15 / 8 = 1 \quad \text{Resto: } 7$$

$$1 / 8 = 0 \quad \text{Resto: } 1$$

Tomando los restos obtenidos en orden inverso tendremos la cifra octal:

Decimal 122 = Octal 172

Conversión de un número decimal a hexadecimal

Utilizando la técnica habitual de divisiones sucesivas, la conversión de un número decimal a hexadecimal. Por ejemplo, para convertir a hexadecimal del número decimal 1735 será necesario hacer las siguientes divisiones:

$$1735 / 16 = 108 \quad \text{Resto: } 7$$

$$108 / 16 = 6 \quad \text{Resto: } C, \text{ es decir, } 12 \text{ en decimal}$$

$$6 / 16 = 0 \quad \text{Resto: } 6$$

De ahí que, tomando los restos en orden inverso, resolvemos el número en hexadecimal:

decimal 1735 = hexadecimal 6C7

Conversión de Binario a Octal

Observa la tabla siguiente, con los siete primeros números expresados en los sistemas decimal, binario y octal:

Decimal	Binario	Octal
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7

Cada dígito de un número octal se representa con tres dígitos en el sistema binario. Por tanto, el modo de convertir un número entre estos sistemas de numeración equivale a "expandir" cada dígito octal a tres dígitos binarios, o en "contraer" grupos de tres caracteres binarios a su correspondiente dígito octal.

Por ejemplo, para convertir el número binario 101001011 a octal tomaremos grupos de tres bits y los sustituiremos por su equivalente octal:

101 = 5 octal

001 = 1 octal

011 = 3 octal

y, de ese modo el número binario 101001011 = octal 513

La conversión de números octales a binarios se hace, siguiendo el mismo método, reemplazando cada dígito octal por los tres bits equivalentes. Por ejemplo, para convertir el número octal 750 a binario, tomaremos el equivalente binario de cada uno de sus dígitos:

7 octal = 111

5 octal = 101

0 octal = 000

y, por tanto, el número octal 750 = 111101000 binario

Conversión de números binarios a hexadecimales y viceversa

Del mismo modo que hallamos la correspondencia entre números octales y binarios, podemos establecer una equivalencia directa entre cada dígito hexadecimal y cuatro dígitos binarios, como se ve en la siguiente tabla:

Decimal	Binario	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E

15	1111	F
----	------	---

La conversión entre números hexadecimales y binarios se realiza "expandiendo" o "con-trayendo" cada dígito hexadecimal a cuatro dígitos binarios. Por ejemplo, para expresar en hexadecimal el número binario 101001110011 bastará con tomar grupos de cuatro bits, empezando por la derecha, y reemplazarlos por su equivalente hexadecimal:

1010 = A

0111 = 7

0011 = 3

y, por tanto, el número binario 101001110011 = al hexadecimal A73

En caso de que los dígitos binarios no formen grupos completos de cuatro dígitos, se deben añadir ceros a la izquierda hasta completar el último grupo. Por ejemplo:

101110 = 00101110 = 2E en hexadecimal

La conversión de números hexadecimales a binarios se hace del mismo modo, reemplazando cada dígito hexadecimal por los cuatro bits equivalentes de la tabla. Para convertir a binario, por ejemplo, el número hexadecimal 1F6 hallaremos en la tabla las siguientes equivalencias:

1 = 0001

F = 1111

6 = 0110

y, por lo tanto, el número hexadecimal 1F6 = al binario 000111110110

1.3 Operaciones básicas (Suma, Resta, Multiplicación y División)

Suma de números binarios

La tabla de sumar para números binarios es la siguiente:

+	0	1
0	0	1
1	1	10

Las posibles combinaciones al sumar dos bits son:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 10$$

Note que al sumar $1 + 1$ es 102, es decir, llevamos 1 a la siguiente posición de la izquierda (acarreo). Esto es equivalente, en el sistema decimal a sumar $9 + 1$, que da 10: cero en la posición que estamos sumando y un 1 de acarreo a la siguiente posición.

Ejemplo:

Acarreo			1					
	1	0	0	1	1	0	0	0
+	0	0	0	1	0	1	0	1
Resultado	1	0	1	0	1	1	0	1

Se puede convertir la operación binaria en una operación decimal, resolver la decimal, y después transformar el resultado en un (número) binario. Operamos como en el sistema decimal: comenzamos a sumar desde la derecha, en nuestro ejemplo, $1 + 1 = 10$, entonces escribimos 0 en la fila del resultado y llevamos 1 (este "1" se llama acarreo o arrastre). A continuación, se suma el acarreo a la siguiente columna: $1 + 0 + 0 = 1$, y seguimos hasta terminar todas las columnas (exactamente como en decimal).

Resta de números binarios

El algoritmo de la resta en sistema binario es el mismo que en el sistema decimal. Pero conviene repasar la operación de restar en decimal para comprender la operación binaria, que es más sencilla. Los términos que intervienen en la resta se llaman minuendo, sustraendo y diferencia.

Las restas básicas $0 - 0$, $1 - 0$ y $1 - 1$ son evidentes:

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 - 1 = 1 \text{ (se transforma en } 10 - 1 = 1 \text{) (en sistema decimal equivale a } 2 - 1 = 1 \text{)}$$

La resta $0 - 1$ se resuelve, igual que en el sistema decimal, tomando una unidad prestada de la posición siguiente: $0 - 1 = 1$ y me llevo 1, lo que equivale a decir en el sistema decimal, $2 - 1 = 1$.

En decimal, por ejemplo, tienes $100 - 19$, obviamente a 0 no le puedes quitar 9, así que debemos tomar prestado 1 para volverlo un 10 (en decimal la base es 10), y así si $10 - 9 = 1$.

En binarios pasa lo mismo, no le puedes quitar 1 a 0, debes de tomar 1 prestado al de un lado, pero cuidado aquí viene lo complicado tu número no se va a volver 10, recuerda que en binario la base es 2 y por lo tanto se volverá 2 en binario, y ahora sí a 2 le quitas 1, $2 - 1 = 1$, y continúas restando, pero recuerda que llevas 1, porque pediste prestado.

Ejemplo para que le entiendas mejor, vamos a restar $201 - 67$, ya sabemos que es 134, vamos a hacerlo en binario:

$$\begin{array}{r} 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1 \dots\dots\dots 201 \\ -\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1 \dots\dots\dots 67 \\ \hline \end{array}$$

Tomamos los dos últimos números, 1-1 es igual a 0, y no llevamos nada (no pedimos prestado)

$$\begin{array}{r}
 11001001 \\
 -01000011 \\
 \hline
 0
 \end{array}$$

Ahora la siguiente columna 0-1, ya dijimos que no se puede, así que va a tomar 1 prestado al de la columna del lado izquierdo, sé que vas a decir "es un cero, no nos puede prestar 1", lo que pasa es que ese cero le pide a su vez al de lado, y así hasta que encuentres un 1, pero no te fijas en eso, vamos a seguir restando y no nos vamos a preocupar por eso ahora, entonces ahora nos prestaron 1 (no importa quién) y tenemos un 1 0 (este número es 2 en binario no 10 en decimal, no te vayas a confundir), entonces en binario tienes 10-1, que en decimal es 2-1=1, y llevamos 1 (porque pedimos 1 prestado)

$$\begin{array}{r}
 11001001 \text{ arriba} \\
 -01000011 \text{ abajo} \\
 \hline
 10
 \end{array}$$

Para la siguiente columna tenemos 0 - 0, pero recuerda que tomamos 1 prestado así que en realidad tenemos 0 - 1 (le sumamos el 1 al de abajo), de nuevo tenemos que pedir prestado y entonces tenemos en binaria 10 -1 que en decimal es 2-1=1, y de nuevo llevamos 1

$$\begin{array}{r}
 11001001 \\
 -01000011 \\
 \hline
 110
 \end{array}$$

Continuamos con 1 - 0, pero como llevamos 1 tenemos ahora 1 - 1, esto si lo podemos resolver $1 - 1 = 0$ (en binario y decimal).

$$\begin{array}{r} 11001001 \\ -01000011 \\ \hline 0110 \end{array}$$

Lo demás es muy fácil:

$$0 - 0 = 0$$

$$0 - 0 = 0$$

$$1 - 1 = 0$$

$$1 - 0 = 1$$

$$\begin{array}{r} 11001001 \\ -01000011 \\ \hline \end{array}$$

$$10000110 \quad \text{que en decimal es 134.}$$

Es lo mismo que la resta en decimal, pides prestado y llevas, nada más debes de ser cuidadoso y recordar que tu base es 2.

"En este mundo solo existen 10 tipos de personas, las que saben binario y las que no" =)

MULTIPLICACIÓN

La tabla de multiplicar para números binarios es la siguiente:

.	0	1
0	0	0
1	0	1

El algoritmo del producto en binario es igual que en números decimales; aunque se lleva a cabo con más sencillez, ya que el 0 multiplicado por cualquier número da 0, y el 1 es el elemento neutro del producto.

Por ejemplo, multipliquemos 10110 por 1001:

$$\begin{array}{r}
 \underline{10110 \times 1001} \\
 10110 \\
 00000 \\
 00000 \\
 \underline{10110} \\
 11000110
 \end{array}$$

División de números binarios

La división en binario es similar al decimal; la única diferencia es que, a la hora de hacer las restas, dentro de la división, éstas deben ser realizadas en binario.

Ejemplo

Dividir 100010010 (274) entre 1101 (13):

$$\begin{array}{r}
 100010010 \overline{)1101} \\
 \underline{-0000} \qquad 010101 \\
 10001 \\
 \underline{-1101} \\
 01000 \\
 \underline{-0000} \\
 10000
 \end{array}$$

$$\begin{array}{r}
 \underline{- 1101} \\
 00011 \\
 \underline{- 0000} \\
 01110 \\
 \underline{- 1101} \\
 00001
 \end{array}$$

1.4 Aplicación de los sistemas numéricos en la computación

Existe una cantidad infinita de sistemas numéricos, sin embargo, para una computadora, únicamente existen 4, que son el Binario (con base 2), el octal (con base 8), el decimal (base 10) y hexadecimal (base 16). Detallaremos el uso de cada uno de ellos por la computadora.

Sistema Binario

El Sistema Binario, por ser el sistema base de la computación y el único entendido de manera nativa por una computadora, es el sistema en el que está escrita toda instrucción, dato, etc. Está compuesto por dos únicos dígitos que 1 y 0 o como en realidad trabaja la computadora, “apagado” y “encendido” y se es como representa todos los datos con los que trabaja la computadora, desde su más bajo nivel: el hardware. Estos dígitos son llamados bits

00
 01
 10 (El uno se movió una posición a la izquierda)
 11

Para números enteros (a la izquierda del punto decimal):

Trigésimo Segundo (32) = 25

Décimo Sexto (16) = 24

Octavo (8) = 21

Cuarto (4) = 22

Segundo (2) = 21

Primero (1) = 20

Para números decimales (a la derecha del punto):

Un Medio = 2^{-1}

Un Cuarto = 2^{-2}

Un Octavo = 2^{-3}

Sistema Octal

Para trabajar la computadora agrupa a los bits en grupos de ocho, a los cuales se denomina byte y es esta la razón por la que es tan importante el sistema octal, sin embargo, una computadora no puede trabajar con el sistema octal como tal, sino que utiliza su conversión en sistema binario, usando tres bits para cada dígito octal

Este sistema es muy usado en trabajos digitales, por su fácil conversión de y hacia el sistema binario. Tiene su base igual a ocho, lo que genera la necesidad de ocho símbolos para representar valores en este sistema y para esta finalidad se seleccionaron los primeros ocho símbolos del sistema decimal: 0, 1, 2, 3, 4, 5, 6 y 7.

A continuación del 7 y para seguir contando hacia adelante, hay que agregar una nueva columna a la izquierda la cual tendrá como valor inicial un 1. De esta forma es posible obtener otras ocho nuevas combinaciones tal como sucedía en los otros sistemas comentados anteriormente. Estos son algunos de los valores para cada símbolo.

Septuagésimo Cuarto (64) = 8^2

Octavo (8) = 8^1

Unidad (1) = 8^0

Un Octavo = 8^{-1}

Un Sesenta y Cuatroavos = 8^{-2}

Los números octales son parecidos a los números decimales excepto por los símbolos 8 y 9, que no son usados.

Sistema Hexadecimal

El sistema hexadecimal es empleado al indexar la memoria o al representar un byte debido a que al contener más dígitos es posible usar menos números para representar números más grandes, haciendo posible que un byte, conformado por 8 bits o términos binarios, se represente con solo dos términos hexadecimales, lo que es un ahorro de información. Sin embargo, la computadora tampoco reconoce el sistema hexadecimal como tal y, al igual que el sistema octal, lo representa con términos binarios, empleando conjuntos de cuatro bits, para cada término

hexadecimal. Sin embargo, al presentar información al usuario es más factible presentar A9 que 10101001.

Este sistema requiere el uso de 16 símbolos, siendo formado por los mismos empleados en el sistema decimal y seis letras del alfabeto arábico comprendidas entre A y F. Dado que las computadoras usualmente agrupan conjuntos de bits en múltiplos de cuatro este sistema permite representar a cada grupo con un simple símbolo. Por ello es que es tan usado en estos días. En la tabla de abajo se muestra la relación entre los sistemas.

Binario	Octal	Hexa	
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Al igual que en los otros sistemas en Hexadecimal, cuando se llega a la F y se requiere seguir contando hacia adelante se torna necesario agregar una nueva columna a la izquierda de la actual la cual inicialmente deberá estar en 1. Esto permite generar otros 16 símbolos nuevos diferentes a los anteriores.

Sistema Decimal

Por último, el sistema decimal únicamente se utiliza al interactuar con el usuario, debido a que un usuario común no está acostumbrado a tratar con diferentes sistemas numéricos.

Este sistema está formado por diez símbolos, llamados números arábigos. También es llamado sistema de base 10. Usando los diez símbolos separadamente 0, 1, 2, 3, ..., 9 nos permite representar el valor de los números en unidades individuales, pero para representar más de nueve números es necesario combinarlos. Cuando usamos símbolos en combinación, el valor de cada uno de

ellos depende de su posición con respecto al punto decimal, designando así un símbolo para las unidades, otro para las decenas, otro para las centenas, otro para los millares (de miles, no de millón), en adelante.

El símbolo correspondiente a las unidades asume la posición más izquierda antes del punto decimal. Esta designación de posición determina que la potencia del número se corresponde con la distancia en que está del punto decimal, y es por ello que la primera posición se llama UNIDAD ($10^0 = 1$). Matemáticamente esto puede ser representado como:

$$\text{unidad} = 10^0 \qquad \text{decena} = 10^1 \qquad \text{centena} = 10^2$$

*Por ejemplo: El valor en combinación de los símbolos **234** es determinado por la suma de los valores correspondientes a cada posición:*

$$2 \times 10^2 \qquad + \qquad 3 \times 10^1 \qquad + \qquad 4 \times 10^0$$

Que equivale a:

$$2 \times 100 \qquad + \qquad 3 \times 10 \qquad + \qquad 4 \times 1$$

Efectuando las multiplicaciones esto da:

$$200 \qquad + \qquad 30 \qquad + \qquad 4$$

Cuya suma da como resultado: 234

La posición derecha del punto decimal es representada por número enteros pero negativos comenzando desde -1 para la primera posición. Matemáticamente las tres primeras posiciones a la derecha del punto decimal se expresan como:

$$\text{décimas } 10^{-1} \qquad \text{centésimas } 10^{-2} \qquad \text{milésimas } 10^{-3}$$

En un ejemplo como el anterior, pero más elaborado podemos ver que el valor 18.947 equivale a:

$$1 \times 10^1 + 8 \times 10^0 + 9 \times 10^{-1} + 4 \times 10^{-2} + 7 \times 10^{-3}$$

=

$$1 \times 10 + 8 \times 1 + 9 \times 0.1 + 4 \times 0.01 + 7 \times 0.001$$

=

$$10 + 8 + 0.9 + 0.04 + 0.007$$

Para representar un número base diez es posible colocar su valor seguido de la base en sub-índice (18.974_{10}) o bien seguido de la letra d entre paréntesis: 645(d).

2 CONJUNTOS Y RELACIONES

Conjunto: Cualquier colección de objetos o individuos. Se denota con mayúsculas.

Elemento: Cierta individuo x que es parte del conjunto A . Se identifican con minúsculas.

Ejemplos:

$$A = \{0, 2, 4, 6, \dots\}$$

2.1 Características de los conjuntos y subconjuntos

Todo conjunto A es subconjunto de sí mismo. Así, dados dos conjuntos $A \subseteq B$, cabe la posibilidad de que sean iguales, $A = B$. Sea A un subconjunto de B tal que $A \neq B$. Entonces se dice que A es un subconjunto propio de B , y se denota por $A \subsetneq B$.

¿Qué son los conjuntos y subconjuntos?

Un subconjunto A de un conjunto B , es un conjunto que contiene algunos de los elementos de B (o quizá todos): Un conjunto A es un subconjunto del conjunto B si cada elemento de A es a su vez un elemento de B . Cuando A es un subconjunto de B , se denota como $A \subseteq B$ y se dice que « A está contenido en B ».

¿Cuáles son las características de un conjunto?

La característica esencial de un conjunto es la de estar bien definido, es decir que dado un objeto particular, determinar si este pertenece o no al conjunto. Por ejemplo, si se considera el conjunto de los números dígitos, sabemos que el 3 pertenece al conjunto, pero el 19 no.

¿Qué es conjunto y subconjunto ejemplos?

Conjuntos subconjuntos

Se da cuando todos los elementos de un conjunto pertenecen al otro. El conjunto de los seres vivos es muy grande: tiene muchos subconjuntos, por ejemplo: Las plantas son un subconjunto de los seres vivos. Los seres humanos son un subconjunto de los animales.

¿Cuáles son los subconjuntos propios de un conjunto?

En Teoría de Conjuntos se define un subconjunto propio A dentro de un conjunto B como aquel subconjunto en el que todos los elementos de A son elementos de B y además existe al menos un elemento de B que no pertenece a A .

¿Qué es un subconjunto para niños?

Un conjunto es subconjunto de otro o está incluido en otro, cuando todos los elementos del primer conjunto son también elementos del segundo conjunto. Si queremos representar gráficamente que un conjunto es subconjunto de otro, el diagrama del primer conjunto debe dibujarse en el interior del diagrama del segundo.

¿Qué significa la B en conjuntos?

Un conjunto es una colección de objetos, conocidos como los elementos del conjunto. $x \in A$ significa que x es un elemento del conjunto A . $B = A$ significa que A y B tienen los mismos elementos. $B \subset A$ significa que B es un subconjunto de A ; Cada elemento de B es también un elemento de A .

¿Qué es un diagrama de Venn y un ejemplo?

Un diagrama de Venn es una representación gráfica que utiliza círculos solapados para ilustrar la relación lógica entre dos o más grupos de elementos. Por ejemplo, si quisiéramos saber qué tienen en común las ranas, las focas y los pájaros, podríamos utilizar un diagrama de Venn.

¿Qué es un conjunto para niños de primer grado?

Cuando clasificamos personas, animales o cosas según características en común, formamos conjuntos.

¿Qué significa \mathbb{N} en los conjuntos?

El conjunto de los números naturales está formado por el 0, 1, 2, 3, 4.

El conjunto de los números naturales se representa mediante la letra \mathbb{N} .

El conjunto de los números naturales es ilimitado, ya que a cada número le sigue siempre otro número.

¿Qué significa por lo menos en conjuntos?

Asimismo, los estudiantes deben entender que las palabras “por lo menos” significan que como mínimo la respuesta será la cantidad designada.

¿Qué significa n minúscula en conjuntos?

El cálculo del número de elementos de un conjunto consiste en contar los elementos del conjunto; por lo tanto, se considerarán conjuntos finitos. Se denominará $n(A)$ al número cardinal de elementos de A o clase de A .

2.2 Operaciones con conjuntos

Operaciones entre conjuntos unión

Para cada par de conjuntos A y B existe un conjunto unión de los dos, que se denota

como $A \cup B$, el cual contiene todos los elementos de A y de B.

Se define así: $A \cup B = \{x \in U / x \in A \vee x \in B\}$

Ejemplo:

Si $A = \{a, b, c\}$ y $B = \{b, c, d, e\}$ entonces,

$A \cup B = \{a, b, c, d, e\}$

Operaciones entre conjuntos INTERSECCION

Los elementos comunes a A y B forman un conjunto denominado intersección de A y B,

representado por $A \cap B$.

Se define así: $A \cap B = \{x \in U / x \in A \wedge x \in B\}$

Ejemplo:

Si $A = \{a, b, c\}$ y $B = \{b, c, d, e\}$ entonces,

$A \cap B = \{b, c\}$

Operaciones entre conjuntos diferencia

Los elementos de un conjunto A que no se encuentran en otro conjunto B, forman otro

conjunto llamado diferencia de A y B, representado por $A - B$.

Se define así: $A - B = \{x \in A / x \notin B\}$

$A - B$

Ejemplo:

Si $A = \{1, 2, 3, 4, 5, 6\}$ y $B = \{2, 4, 8, 9\}$ entonces,

$A - B = \{1, 3, 5, 6\}$

Operaciones entre conjuntos complemento

El complemento de un conjunto A es el conjunto de todos los elementos que no pertenecen a A.

$cA = cUA ; \{x \in U / x \notin A\}$

Ejemplo:

Si $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ y $A = \{2, 4, 6\}$ entonces,

$cA = \{1, 3, 5, 7, 8, 9\}$

Operaciones entre conjuntos Diferencia simétrica

La diferencia simétrica de dos conjuntos A y B viene dada por los elementos que pertenecen a uno y sólo uno de los dos.

Se define así: $A \Delta B = (A-B) \cup (B-A)$

Ejemplo:

Si $A = \{1, 2, 3, 4, 5, 6\}$ y $B = \{4, 5, 6, 7, 8, 9\}$ entonces

$A-B = \{1, 2, 3\}$; $B-A = \{7, 8, 9\}$

$A \Delta B = \{1, 2, 3, 7, 8, 9\}$

2.3 Propiedades y aplicaciones de los conjuntos

Inclusión

Dados dos conjuntos A y B, se dice que A es un subconjunto de B, y se expresa $A \subseteq B$, cuando todos los elementos de A son también elementos de B.

Complementación

Dado un conjunto U y un subconjunto $A \subseteq U$ se llama complementario del conjunto A, y se denota A^c , al subconjunto de U formado por todos los elementos que no pertenecen a A.

Unión

Dados dos conjuntos A y B se llama unión de A y B, y se representa $A \cup B$, al conjunto formado por los elementos que pertenecen a A o a B.

Intersección

Dados dos conjuntos A y B se llama intersección de A y B, y se representa $A \cap B$, al conjunto formado por los elementos que pertenecen a A y a B.

Diferencia

Dados dos conjuntos A y B se llama diferencia entre A y B, y se representa $A \setminus B$ o $A - B$, al conjunto formado por los elementos de A que no pertenecen a B.

Producto Cartesiano

Dados dos conjuntos A y B, se llama producto cartesiano de A por B, y se denota $A \times B$, al conjunto constituido por pares ordenados de elementos, el primero perteneciente al conjunto A y el segundo al B.

Aplicaciones

El concepto de aplicación (función) es de gran importancia en Informática. Una función es el modo más natural de implementar la correspondencia entre los datos y el resultado de un proceso de cálculo en un ordenador. Los llamados lenguajes funcionales como OCAML, HASKELL, etc., se fundamentan en este concepto y suelen identificar programa y función.

2.4 Conceptos básicos: producto cartesiano y relación binaria

Producto cartesiano:

Para entender la idea de producto cartesiano debemos saber que se trata de una operación entre dos conjuntos, de tal modo que se forma otro conjunto con todos los pares ordenados posibles (el ejemplo utilizado en los conceptos es un producto cartesiano).

Por ejemplo, dados los conjuntos $A = \{1, 2, 3, 4\}$ y $B = \{a, b\}$, su producto cartesiano es: $A \times B = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b), (4, a), (4, b)\}$

Los elementos de $A \times B$ son pares ordenados. Cada par que se forma con un elemento del conjunto A y uno del conjunto B, en ese orden, recibe el nombre de par ordenado. Sus elementos se colocan entre paréntesis, separados por coma.

Relación binaria:

Llamamos relación binaria a la relación R existente entre dos elementos a y b, de dos conjuntos A y B respectivamente. Indicando que el elemento a está relacionado con b. Esta relación se puede denotar de diversas formas:

1- Como pares ordenados (a, b). 2- Indicando que aRb . 3- Como una mezcla entre los dos anteriores $R(a, b)$.

Al conjunto de todos los elementos relacionados mediante la relación R en un conjunto lo denotamos como $R(M)$

Ejemplo: Sea el conjunto $A = \{\text{el conjunto de los números naturales}\}$, una relación binaria del conjunto de A sobre sí mismo puede ser, $R = \text{ser múltiplo de...}$

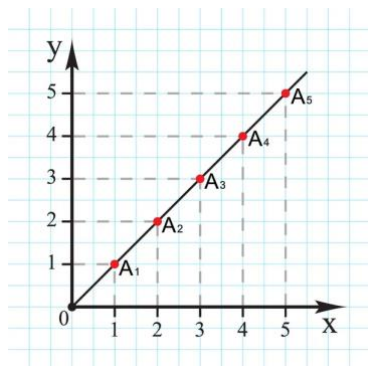
De tal forma que, por ejemplo 4 está relacionado con 2 (es decir, 4 es un múltiplo de 2), por tanto, escribimos $4R2$ o $(4,2)$.

En el caso de no estar relacionados escribiremos a no está relacionado con b tachando la R. Un ejemplo de dos elementos que no están relacionados con esta relación son 3 y 5.

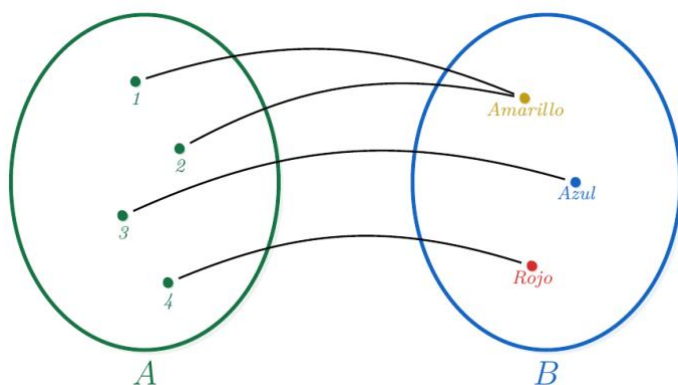
2.5 Representación de las relaciones

Para representar las relaciones binarias podemos utilizar dos tipos de gráficos:

a) El diagrama cartesiano: donde representaremos los ejes cartesianos, y en cada eje los elementos de cada conjunto. Representaremos las relaciones por medio de puntos (si el eje es similar al eje de coordenadas) o por medio de cruces si lo representamos mediante cuadrículas



b) Diagrama sagital o flechas (mediante diagramas de Venn): representaremos los elementos del conjunto dentro del círculo y representaremos las relaciones mediante flechas.



2.6 Propiedades de las relaciones

Producto Punto

Es aquella dentro del conjunto de las relaciones en A y B sean pertenecientes a U, el producto cruz se define como un nuevo conjunto con elementos de diferentes tamaños “a lo” de orden 2 y elementos de “u”.

Las relaciones y sus propiedades se definen en 4 términos:

- 1.-Uno a Uno
- 2.-Uno a muchos
- 3.-Muchos a uno
- 4.-Muchos a muchos

Uno A Uno

En esta propiedad se dice que la reacción debe ser exclusiva ejemplo:

Un hombre nacido en México solo puede tener una esposa y una mujer nacida en México solo puede tener un esposo y su relación es 1-1.

UNO A UNO

$$C=(a_1,b_2,c_3)$$

$$C=B \rightarrow A = B * A$$

$$A*B (1^a,2b,3c)$$

$$Ax=xa$$

Uno A Muchos

¡La relación de esta propiedad se dice que es exclusiva! Ejemplo:

Un alumno puede tener uno y solo un grupo y un grupo puede tener muchos alumnos.

Muchos A Uno A*B M-1

$$C=(1a,1b,1c,2^a,2b,2c,3^a,3b,3c)$$

MUCHOS A UNO: Esta propiedad se dice que la relación es inclusiva e inversa ejemplo:

En una caja de lápices puede tener muchos lápices y un lápiz solo puede estar en una caja M-1

Muchos A Muchos

Se define como la propiedad de muchas exclusiones.

Ejemplo:

Un producto puede estar en una o muchas ventas y una venta puede tener uno o muchos productos.

2.7 Relaciones de equivalencia

Suponga que se tiene un conjunto X de 10 pelotas, cada una de las cuales es roja, azul o verde. Si se dividen las pelotas en los conjuntos R , A y V de acuerdo con el color, la familia $\{R, A, V\}$ es una partición de X . Una partición es útil para definir una relación. Si S es una partición de X , se puede definir $x R y$ de modo que signifique que para algún conjunto $S \in S$, tanto x como y pertenecen a S . La relación obtenida se describe como “es del mismo color que...”. El siguiente teorema muestra que este tipo de relación siempre es reflexiva, simétrica y transitiva.

Sea S una partición de un conjunto X . Defina $x R y$ de modo que signifique que para algún conjunto S en S , tanto x como y pertenecen a S . Entonces R es reflexiva, simétrica y transitiva.

Demostración: Sea $x \in X$. Por definición de partición, x pertenece a algún conjunto $S \in S$. Entonces, $x R x$ y R es reflexiva. Suponga que $x R y$. Entonces ambas x y y pertenecen a algún conjunto $S \in S$. Como ambos y y x pertenecen a S , $y R x$ y R son simétricas. Por último, suponga que $x R y$ y $y R z$. Entonces ambos x y y pertenecen a algún conjunto $S \in S$ y ambos y y z pertenecen a algún conjunto $T \in S$. Como y pertenece exactamente a un miembro de S , debemos tener $S = T$. Por lo tanto, ambas x y z están en S y $x R z$. Se ha demostrado que R es transitiva.

Para que esto último se entienda de una mejor forma, se da un ejemplo:

Considere la relación

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

En $\{1, 2, 3, 4, 5\}$ la relación es reflexiva porque $(1, 1), (2, 2), (3, 3), (4, 4), (5, 5) \in R$. La relación es simétrica porque siempre que (x, y) está en R , (y, x) también está en R . Por último, la relación es transitiva porque siempre que (x, y) y (y, z) están en R , (x, z) también está en R . Como R es reflexiva, simétrica y transitiva, R es una relación de equivalencia en $\{1, 2, 3, 4, 5\}$.

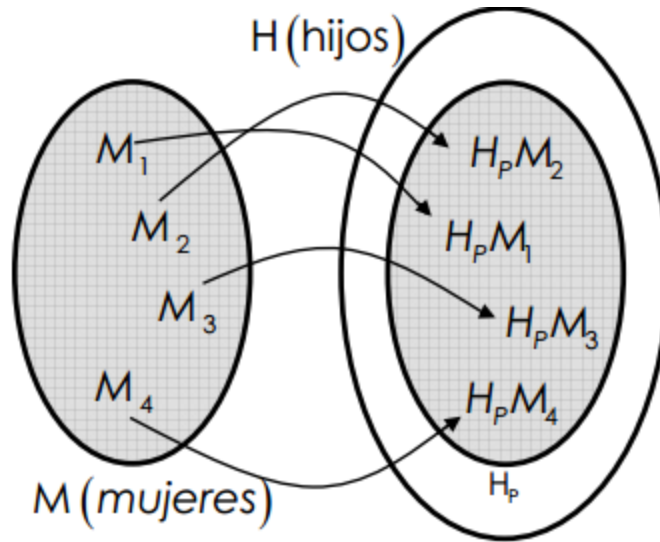
2.8 Funciones

Esta clasificación obedece a la forma en que están relacionados los elementos del dominio con los del recorrido. Conviene utilizar la notación: $f: D_f \rightarrow C_f$ "función que mapea al dominio D_f en el recorrido C_f ".

Función inyectiva (uno a uno): es inyectiva o uno a uno y se denota como 1-1, si a diferentes elementos del dominio le corresponden diferentes elementos del codominio. En esta función, para dos valores cualesquiera x_1 y x_2 de su dominio se cumple que: x_1 es diferente de x_2 , entonces $f(x_1)$ es diferente que $f(x_2)$.

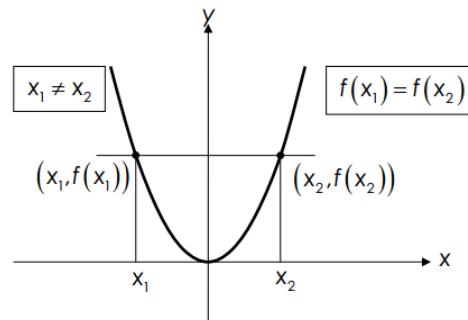
Ejemplo: La función $f(x) = 3x + 1$ es 1-1 ya que si se define como $f: \mathbb{R} \rightarrow \mathbb{R}$ entonces se tendrá que a diferentes elementos del dominio les corresponden diferentes elementos del codominio.

Ejemplo: Sea M el conjunto de mujeres con hijos, H el conjunto de los hijos y f la función que asocia a cada mujer con su hijo primogénito. Es una función 1-1 o inyectiva.



Para comprobar gráficamente que una función es 1-1 basta con comprobar que toda recta paralela al eje "x" corta a la gráfica de la función en un solo punto a la vez.

Ejemplo. Sea la función $f: \mathbb{R} \rightarrow \mathbb{R}$ dada por $f(x) = x^2$.



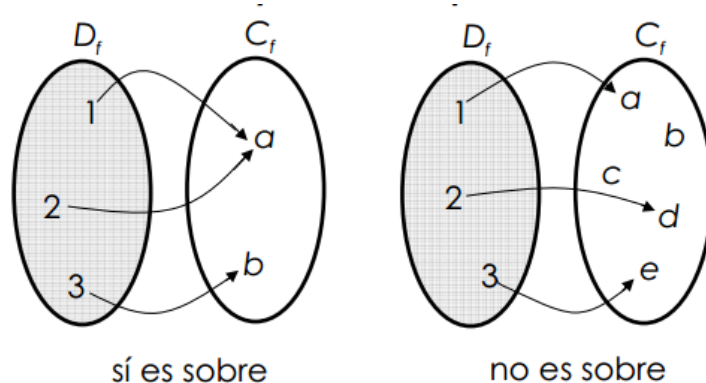
Función suprayectiva (sobre): Una función es suprayectiva o sobre si todo elemento de su Codominio es imagen de por lo menos un elemento de su Dominio.

Si para toda $b \in Cf$, existe $a \in Df$ tal que $f(a) = b$, entonces f es sobreyectiva.

Otra forma de expresar que una función es sobre es decir que debe cumplir con que su Codominio y su Recorrido sean iguales.

Ejemplo: Sea la función $f(x) = 3x + 1$ definida como $f: \mathbb{R} \rightarrow \mathbb{R}$. En este caso se ve que todo número real es imagen de algún otro número real bajo la función f .

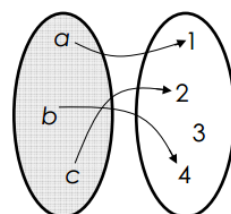
Esto significa que el recorrido es igual al codominio y por lo tanto la función dada es suprayectiva o sobre.



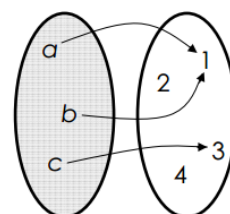
Función biyectiva (1-1 y sobre): Una función es biyectiva si al mismo tiempo es inyectiva y suprayectiva, y la relación entre los elementos del dominio y los del codominio es biunívoca.

Una función puede ser:

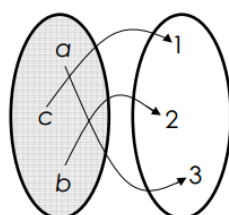
- I) 1-1 y sobre (biyectiva).
- II) 1-1, pero no sobre.
- III) No 1-1, pero sí sobre.
- IV) Ni 1-1 ni sobre.



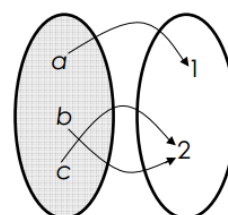
1-1 sí y sobre no



1-1 no y sobre no



Biyectiva
1-1 y sobre



1-1 no y sobre sí

2.9 Aplicaciones de las relaciones y las funciones en la computación.

Uno de los conceptos más importantes en Matemáticas es el de función, ya que se puede aplicar en numerosas situaciones de la vida cotidiana, y determinar las relaciones que existen entre magnitudes tanto en Matemáticas, Físicas, Economía, etc., y poder calcular el valor de una de ellas en función de otras de las que depende.

Ya desde hace años, se observaron fenómenos que estaban relacionados con otros, así el volumen de un gas a temperatura constante, está relacionado con la presión, la fuerza de atracción entre dos cuerpos se vio que estaba relacionada con la masa de esos cuerpos y la distancia que les separa, y el capital final de una inversión está determinado por el capital invertido y el tiempo que dure esa inversión, etc.

APLICACIONES DE LAS FUNCIONES A DISTINTAS ÁREAS:

En cualquier área de las ciencias, existen leyes en las que se relacionan distintas magnitudes, temperatura-presión, masa-velocidad, intensidad del sonido-distancia,

etc. Es decir, a partir de los valores de algunas magnitudes se obtienen los valores de otras de forma directa a través de fórmulas ya demostradas.

Un punto de origen del concepto de función nace precisamente de las relaciones que mantienen diferentes magnitudes, así pues la función se puede representar algebraicamente o de forma gráfica en la que se relacionan varias magnitudes entre sí.

Mediante la representación gráfica de estas relaciones entre diferentes magnitudes, se pudo dar de forma visual esa relación e interpretarla de forma rápida y sencilla. Una forma de representación es la que se hace mediante ejes cartesianos, en la que se la función se representa de forma general por la relación numérica de magnitudes en una gráfica.

Así pues, la función la podremos representar tanto gráficamente como mediante una expresión algebraica o fórmula.

Euler fue el primero en emplear la expresión $f(x)$ para representar una función f asociada a un valor x . Es decir, con esta representación que es empleada hoy, se comienza la utilización del concepto de función tal y como hoy se entiende.

- Función en Cinemática:

El problema consiste en expresar la relación entre el espacio recorrido y el tiempo invertido en ello. Si queremos la función que representa el espacio recorrido por un móvil, con velocidad uniforme que parte del reposo $e(t) = v \cdot t$ que es una función del tipo $f(x) = m \cdot x$ cuya gráfica es una recta dependiente de m y que pasa por el origen de coordenadas.

Otro problema muy común y que su uso es muy estudiado es el lanzamiento de proyectiles. Las funciones son de tipo cuadrática de la forma $y = ax^2 + bx + c$. Por ejemplo, si queremos calcular la distancia que alcanza un objeto que es lanzado hacia arriba con una inclinación determinada α y a una velocidad inicial de lanzamiento v_0 , en función del tiempo se puede representar de forma gráfica y algebraica:

$$x = x_0 + v_{0x}t$$

$$1 \quad 2$$

$$y = y_0 + v_{0y}t - \frac{1}{2}gt^2$$

$$2$$

$$v_x = v_0 \cos \alpha$$

$$v_y = v_0 \sin \alpha - gt$$

Según las magnitudes que se quieran relacionar las expresiones tanto gráficas como algebraicas serán las adecuadas:

Función en Dinámica:

Cuando una partícula tiene una trayectoria curvilínea, está sometida a una aceleración perpendicular a la trayectoria y dirigida hacia el centro de la curva, llamada aceleración centrípeta y cuya expresión es, esta aceleración es producida por una fuerza cuya expresión es $F = M \cdot a$ = expresión que es una función cuadrática.

También en dinámica se emplean funciones que describen fenómenos cotidianos. Las funciones se pueden obtener de forma experimental o por medio de fórmulas.

Representar por ejemplo la longitud que puede alcanzar un muelle desde el que se cuelga un peso viene dada por una función de tipo lineal del tipo $y = ax + b$ que se representa por una recta.

También por la ley de Hooke $F = K \cdot x$ se puede determinar por medio de una tabla de valores o por una gráfica la fuerza o peso que se debe aplicar para que el muelle se desplace una cierta distancia.

Por ejemplo, si se tiene un muelle con una constante de elasticidad $k = 0,5$, podemos ir representando la relación entre las magnitudes fuerza-distancia

- Función en Energía:

La energía cinética viene expresada por E_c = de tipo cuadrático.

- Función de crecimiento ilimitado:

Responde a la forma $f(x) = a \cdot b^x$ con $a, c > 0$ y $b > 1$.

Es por ejemplo el crecimiento de la población $P_t = (1 + r') \cdot P_0$, donde P_t es el crecimiento de la población al cabo de t años, r' es el crecimiento anual de la población de forma constante expresado en tanto por 1, P_0 es la población actual.

- Función de decrecimiento limitado:

Su ecuación viene dada por $f(x) = a \cdot b^x$ con $a > 0$, $b > 1$ y $c > 0$. Es por ejemplo la desintegración radiactiva cuya fórmula $N_t = N_0 e^{-\lambda t}$, donde N_t es el número de átomos en el momento t , N_0 es el número de átomos radiactivos iniciales, λ es la constante de desintegración, t es el tiempo.

- Función de crecimiento limitado:

Su ecuación es de la forma $f(x) = a(1 - e^{-bx})$ con $a > 0$. Es por ejemplo las pruebas de memoria cuya fórmula viene dada por $n = n(1 - e^{-0,2x})$ donde n es el número de objetos que se pueden recordar y x es el número de minutos que se les muestran.

- Función del sonido:

La intensidad del sonido que podemos percibir desde un punto sonoro llamado foco dependerá de la distancia a la que se encuentre el receptor desde el punto emisor del sonido.

Así pues, esta intensidad que recibe el receptor vendrá dada por la fórmula: $I = \frac{P}{4\pi d^2}$ en la que I es la intensidad del sonido medida en decibelios y d es la distancia medida en metros a la que se encuentra el receptor del emisor.

La función que representa las magnitudes intensidad del sonido-distancia es de la forma:

- Función de Economía:

Para el estudio de la función de costes de una empresa, cuando una empresa produce ciertos bienes, genera ciertos gastos llamados costes. Para tener una producción eficiente, la función de costes debe ser mínima.

La función de costes depende de la relación:

$$C_t(Q) = C_v(Q) + C_f$$

Donde Q es la cantidad de producto producido, C_t es el coste total, C_v son los costes variables en función de la cantidad de producto producido y C_f son los costes fijos de producción.

- Función en Termodinámica:

La Ley de Boyle, nos dice que para un gas a temperatura constante, se verifica la siguiente relación

Entre la presión y el volumen:

$$P \cdot V = K \Rightarrow P = \frac{K}{V}$$

Su representación gráfica es una hipérbola equilátera cuyas asíntotas coinciden con los ejes de coordenadas. En este caso se representa mediante la rama positiva de la hipérbola, pues no tiene sentido hablar de presiones o volúmenes negativos.

Otras funciones importantes:

- Función en la Ley de la gravitación universal de Newton y Ley de Coulomb.
- Ley de la medida de la intensidad de una onda
- Escala de Richter $M = \log_{10} P$
- Las funciones circulares: relacionadas con las vibraciones, propagación de ondas y movimiento pendular.

Unidad 3

2. Lógica matemática

3.1 Lógica proposicional

La lógica proposicional trata sobre la verdad o la falsedad de las proposiciones y de cómo la verdad se transmite de unas proposiciones (premisas) a otras (conclusión). Una proposición es la unidad mínima de significado susceptible de ser verdadera o falsa.

Una palabra aislada, por sí misma, no nos dice nada. La palabra "perro" tiene una referencia, pero no nos da ninguna información si no es en el contexto de una proposición como "El perro está haciendo cosas raras". Por ello una palabra, a menos que constituya una proposición, no es verdadera o falsa. Sólo tienen valor de verdad las proposiciones.

Debemos distinguir dos tipos de proposiciones: las proposiciones atómicas y las proposiciones moleculares. Las proposiciones atómicas son aquellas que no se componen de otras proposiciones. La proposición

Todos los hombres son mortales

Es una proposición atómica porque ninguno de sus elementos componentes es una proposición. Como podemos observar, una proposición atómica es verdadera o falsa, y su verdad o falsedad no depende de otras proposiciones, sino de cómo es la realidad. Si hubiera algún hombre inmortal, la proposición del ejemplo sería falsa.

Las proposiciones moleculares son aquellas que están compuestas por proposiciones atómicas. Un ejemplo de proposición molecular sería:

Voy a comprar pan y a tomar un café

La proposición del ejemplo es molecular porque se compone de dos proposiciones atómicas:

Voy a comprar pan

Voy a tomar un café

Estas dos proposiciones atómicas están conectadas mediante la partícula "y". Una proposición molecular será verdadera o falsa, pero a diferencia de lo que ocurre con las proposiciones atómicas, su verdad o falsedad no depende directamente de la realidad, sino que depende o es función de la verdad o falsedad de las proposiciones atómicas que la componen. Esto significa que si quiero saber si es verdadero o falso que voy a comprar pan y a tomar un café, es necesario que conozca la verdad o falsedad de "voy a comprar pan" y de "voy a tomar un café" por separado.

3.1.1 Propositiones simples y compuestas

Una proposición compuesta es una frase que consta de uno o varios sujetos y de un predicado que afirma algo en torno a dichos sujetos. Los sujetos de una proposición simple deben ser todos términos singulares. El predicado debe contener un verbo que exprese la acción sobre los sujetos. En matemáticas se usan ciertos símbolos para representar predicados de uso frecuente como: el símbolo “ $=$ ”, como representante del predicado “es igual a”, y el símbolo “ $<$ ” como sustituto de “es menor que”.

Las proposiciones compuestas son las siguientes:

1.-Disyunción

La conectiva ‘o’ disyunción tiene dos sentidos y a ellos se alude ya en el lenguaje ordinario cuando se distingue entre ‘o’ y ‘o’’. Cada uno de dichos sentidos es expresado en la lógica sentencia mediante un signo propio. La conectiva ‘o’ corresponde a la llamada disyunción inclusiva es simbolizada por el signo ‘ \vee ’ insertado entre dos fórmulas. La disyunción de las proposiciones simples $p \vee q$ que se lee: “p o q” es falsa si ambas proposiciones son falsas. El operador lógico disyunción también se denomina OR y representa la suma lógica. Esta se puede describir mediante una tabla de verdad, una tabla de verdad de una proposición P formada por las proposiciones P_1, \dots, P_n donde V indica verdadero y F falso, de modo que para cada una de estas combinaciones se indica el valor de verdad de P.

$p \vee q$

Se lee: ‘p o q’. La conectiva ‘o...o’ corresponde a la llamada disyunción exclusiva y es simbolizada por el signo ‘ \neq ’ insertado entre dos fórmulas.

$P \neq q$

Se lee ‘o o q’.

Ejemplo:

Antonio se dedica a la natación o al alpinismo

Se entiende de las dos siguientes maneras:

1.- Antonio se dedica a la natación o al alpinismo (o a ambos) (ejemplo de disyunción inclusiva)

2.- Antonio se dedica a la natación al alpinismo (pero no a ambos) (ejemplo de disyunción exclusiva)

Toda disyunción afirma que es cierto lo que afirma por lo menos una de sus componentes esto se define en cuanto a su interpretación

2.-Conjunción

La conjunción de dos proposiciones simples $P \wedge q$ (que se lee:" p y q") es verdadera si ambas proposiciones son verdaderas. La conjunción (\wedge), es una conectiva lógica que denomina el operador lógico AND y representa el producto lógico, y es una proposición de la forma P y Q donde estos son proposiciones cuales quiera.

La conectiva 'y' o conjunción es simbolizada por el signo ' \wedge ' insertado entre dos fórmulas. Así,

$p \wedge q$

Se lee ' p y q'

Te mostramos unos ejemplos sencillos para comprender el estudio de la conjunción.

Antonio es mentiroso y Juan es mentiroso,

Se formula de modo más idiomático escribiendo:

Antonio y Juan son mentirosos

Si p es la proposición: «1 es un número impar» y q es la proposición: «3 es un número primo», entonces $p \wedge q$ será la proposición: «1 es un número impar y 3 es un número primo». En donde se observa que $p \wedge q$ su valor de verdad es verdadero, pues tanto p : «1 es un número impar», como q : «3 es un número primo», ambos son verdaderos.

Si p es la proposición: «París está en Francia» y q es la proposición: «2 es un número impar», entonces la proposición: $p \wedge q$ será «París está en Francia y 2 es un número impar», donde su valor de verdad es: falso, pues el valor de verdad de p : «París está en Francia», es verdadero, pero el valor de q : «2 es un número impar» es falso.

3.-Negación

La negación de p , denotada por $\neg p$, es la proposición

No p

Las proposiciones pueden ser simples o compuestas. Para designarlas se emplean letras latinas minúsculas: p , q , r , s , etc. Para negar una proposición simple se emplea el símbolo \neg de tal forma que $\neg p$ (se lee "no p "), y es tal, que si p es verdadera (1), $\neg p$ sería falsa (0), y viceversa. El operador negación (\neg) también se

denomina NOT por razones obvias ' _ ' prefijado a la letra setencial o al enunciado. Así,

-p

Se lee 'no p'

Ejemplo:

No (Nehru es francés) Aquí se dice que la conectiva designa 'partícula conectiva'. Se usa el adjetivo sustantivo.

O bien seria:

No es el caso que Nehru sea francés Se observa que en el lenguaje lógico 'no' antecede al enunciado, en el lenguaje ordinario (en español) sigue al sujeto

En una expresión más idiomática a los dos ejemplos anteriores es:

Nehru no es francés.

La conectiva 'no' o negación es la única conectiva singular de las mentadas.

4.-Condicional

Si p y q son proposicionales, la proposición compuesta de la siguiente forma:

Si p entonces q

Es una proposición condicional y se denota

$p \rightarrow q$

Donde las componentes P, Q son proposiciones cuales quiera. La primera componente de una condicional es La proposición p es la hipótesis (o antecedente) y la segunda proposición q es la conclusión (o consecuente).

EJEMPLO:

Si X es múltiplo de A, entonces X es par

Nota: Procura poner el verbo en subjuntivo cuando una componente vaya precedida de "que". En (Q cada vez P), se hace salvedad de esta regla.

Cuando la hipótesis de una condicional es una conjunción, suele repetirse la palabra "si" tantas veces como componentes tiene la hipótesis.

Ejemplo, en lugar de:

Si P y Q entonces R:

Se escribe:

Si P y si Q entonces R.

Ejemplo de lo anterior es:

Si la condicional es:

Si P entonces Q.

Recíproca es:

Si Q entonces P.

Contrapuesta es:

Si $\sim Q$ entonces $\sim P$.

A cada condicional le asocian otras dos que juegan un papel importante: la recíproca y la contrapuesta de la condicional dada.

5.-Bicondicional

Una bicondicional es la conjunción de una condicional y su recíproca.

Sus siete maneras diferentes de enunciar una misma condicional es:

1.- Si P entonces Q y, recíprocamente, Si Q entonces P

2.- Si P entonces Q, y recíprocamente.

3.- Si p, y solo entonces, Q.

4.- P si Q, y solo entonces.

5.- P si Q, y solo si Q.

6.- P si, y solo si, Q.

7.- A fin de que P es necesario y suficiente que Q.

Si p y q son proposiciones, la proposición compuesta

p si y solo si q

Es una proposición bicondicional y se denota

$p \leftrightarrow q$.

3.1.2 Tablas de verdad

Tablas de verdad de las conectivas lógicas

Formalizar una proposición es sólo el primer paso. Ahora tenemos que analizar las fórmulas obtenidas en relación con su verdad o la falsedad. El valor de verdad de las proposiciones moleculares depende del valor de verdad de las proposiciones atómicas que la componen y de las conectivas lógicas. Una proposición atómica puede ser verdadera o falsa. Nosotros adoptaremos la convención de referirnos al valor de verdad "Verdadero" con el símbolo "1" y al valor de verdad "Falso" con el símbolo "0". Podemos expresar los posibles valores de verdad de una proposición atómica mediante la siguiente tabla:

p
1
0

Esta tabla significa que la proposición atómica "p" (que puede ser cualquier proposición atómica) puede ser verdadera (1) o falsa (2). En realidad no sabemos si es verdadera o falsa, porque eso depende de su significado, que desconocemos. Pero lo que sabemos con toda seguridad es que debe tener uno de esos valores de verdad.

La cosa se complica cuando pretendemos averiguar los posibles valores de verdad de una proposición molecular. En efecto, la proposición molecular.

$p \wedge q$

Puede ser verdadera o falsa, pero su verdad o falsedad depende de la verdad o falsedad de p y de q . Así pues, si p es verdadera pero q es falsa, $(p \wedge q)$ será falsa, por ejemplo. A cada combinación de valores de verdad de p y de q , le corresponde un valor de verdad a la proposición compleja. Podemos expresar esto con la siguiente tabla de verdad de la conjunción:

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

Como vemos en la tabla, la fórmula $(p \wedge q)$ sólo es verdadera cuando p es verdadera y q es verdadera, siendo falsa en todos los demás casos. Podemos confeccionar una tabla semejante para todas las conectivas lógicas:

Tabla de verdad de la disyunción

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

Como vemos, la disyunción sólo es falsa en caso de que sus dos términos lo sean, y es verdadera en todos los demás supuestos.

Tabla de verdad del condicional

p	q	$p \supset q$
1	1	1
1	0	0
0	1	1
0	0	1

3.1.3 Tautologías, contradicción y contingencia

•TAUTOLOGÍA: Una proposición compuesta es una tautología si es verdadera para todas las asignaciones de valores de verdad para sus proposiciones componentes. Dicho de otra forma, su valor V no depende de los valores de verdad de las proposiciones que la forman, sino de la forma en que están establecidas las relaciones $A \vee \neg A$

sintácticas de unas con otras. Sea el caso: A

A	$\neg A$	$A \vee \neg A$
V	F	V
F	V	V

lor \neg

•CONTRADICCIÓN: Se entiende por proposición contradictoria, o contradicción, aquella proposición que en todos los casos posibles de su tabla de verdad su valor siempre es F. Dicho de otra forma, su valor F no depende de los valores de verdad de las proposiciones que la forman, sino de la forma en que están establecidas las relaciones sintácticas de unas con otras. Sea el caso: $A \wedge \neg A$

A	$\neg A$	$A \wedge \neg A$
V	F	F
F	V	F

•CONTINGENCIA: Se entiende por verdad contingente, o verdad de hecho, aquella proposición que puede ser verdadera o falsa, (combinación entre tautología y contradicción) según los valores de las proposiciones que la integran. Sea el caso: $A \wedge (B \vee C)$

$$A \wedge (B \vee C)$$

A	B	C	$B \vee C$	$A \wedge (B \vee C)$
V	V	V	V	V
V	V	F	V	V
V	F	V	V	V
V	F	F	F	F
F	V	V	V	F
F	V	F	V	F
F	F	V	V	F
F	F	F	F	F

3.1.4 Equivalencias lógicas

Dos fórmulas lógicas son equivalentes si tienen los mismos valores de verdad para todos los posibles valores de verdad de sus componentes atómicos.

Diremos que dos proposiciones P y Q son lógicamente equivalentes si es una tautología, es decir, si las tablas de verdad de P y Q son iguales.

Equivalencia lógica en la ley asociativa de la conjunción

A modo ilustrativo demostraremos, a continuación, que, en virtud de la ley asociativa de la conjunción, la fórmula $p(qr)$ es lógicamente equivalente a $(pq)r$.

Para ello no hay más que hacer la tabla de verdad de cada una de esas expresiones y comprobar si, en efecto, todas sus interpretaciones son iguales para la conectiva dominante.

Equivalencia lógica en la ley asociativa de la disyunción

Te proponemos que rellenes la siguiente tabla con “Vs” y “Fs” donde proceda para comprobar que, en virtud de la ley asociativa de la disyunción, la fórmula $p \vee (q \vee r)$ es equivalente a $(p \vee q) \vee r$.

Si dos fórmulas lógicas son equivalentes entonces la fórmula que se obtiene al operarlas con la bicondiconal es una tautología.

EJEMPLO

Equivalencia lógica en la ley asociativa de la conjunción

A modo ilustrativo demostraremos, a continuación, que, en virtud de la ley asociativa de la conjunción, la fórmula $p \wedge (q \wedge r)$ es lógicamente equivalente a $(p \wedge q) \wedge r$.

Para ello no hay más que hacer la tabla de verdad de cada una de esas expresiones y comprobar si, en efecto, todas sus interpretaciones son iguales para la conectiva dominante.

Equivalencia lógica en la ley asociativa de la disyunción

Te proponemos que rellenes la siguiente tabla con “Vs” y “Fs” donde proceda para comprobar que, en virtud de la ley asociativa de la disyunción, la fórmula $p \vee (q \vee r)$ es equivalente a $(p \vee q) \vee r$.

Si dos fórmulas lógicas son equivalentes entonces la fórmula que se obtiene al operarlas con la bicondiconal es una tautología.

$$(p \rightarrow \neg q) \vee (\neg p \vee r) \leftrightarrow \neg p \vee \neg q \vee r$$

p	q	r	$\neg q$	$\neg p$	$p \rightarrow \neg q$	$\neg p \vee r$	$(p \rightarrow \neg q) \vee (\neg p \vee r)$	$\neg p \vee \neg q$	$\neg p \vee \neg q \vee r$
V	V	V	F	F	F	V	V	F	V
V	V	F	F	F	F	F	F	F	F
V	F	V	V	F	V	V	V	V	V
V	F	F	V	F	V	F	V	V	V
F	V	V	F	V	V	V	V	V	V
F	V	F	F	V	V	V	V	V	V
F	F	V	V	V	V	V	V	V	V
F	F	F	V	V	V	V	V	V	V

Donde se puede observar que la última y la antepenúltima columnas son iguales.

3.1.5 Reglas de inferencia

Las reglas de inferencia son también llamadas reglas de transformación y su principal característica es que nos permiten dar conclusiones muy bien formadas y validas a partir de otras premisas.

Las reglas de inferencia se clasifican en: Atómicas (Simples) y Moleculares (Compuestas)

Dentro de la inferencia encontramos sus reglas en donde es muy fácil aprender su uso. Se debe utilizar las preposiciones o formas lógicas nombres que se le dará a las preposiciones.

Una premisa verdadera conducirá a una conclusión verdadera.

Conjunción	\wedge - + - ^
Disyunción	\vee - \sqcup
Negación	No
Condional	Si... Entonces

La regla de inferencia son argumentos válidos breves que se utilizan dentro de un argumento más largos como una demostración.

DEMOSTRACIONES:

REGLA DE INFERENCIA	NOMBRE	REGLA DE INFERENCIA	NOMBRE
$\frac{A \rightarrow B \quad A}{B}$	MODUS PONENS	$\frac{p \rightarrow q \quad q \rightarrow r}{p \rightarrow r}$	SILOGISMO HIPOTETICO
$\frac{A \rightarrow B \quad \neg B}{\neg A}$	MODUS TOLLENS	$\frac{A \vee B \quad \neg A}{B}$	SILOGISMO DISYUNTIVO

3.1.6 Argumentos válidos y no válidos

Un argumento es correcto – del punto de vista lógico, si siempre que las premisas son verdaderas su conclusión lo es por razones formales. O, dicho de otro modo, si es imposible por razones formales que las premisas sean verdaderas y la conclusión sea falsa. En este caso se dice que la conclusión es consecuencia lógica de las premisas o que las premisas implican la conclusión

Si probamos con todas las alternativas, resulta que o y no son las únicas expresiones que no pueden intercambiarse por otras.

De esto es evidente que la validez de (1) depende solo del hecho de que una de las premisas consiste de dos enunciados conectados por la conjunción o, que la otra premisa es la negación del primer enunciado de la primera premisa y que la conclusión es el segundo enunciado de la primera premisa. Y (1) no es el único argumento cuya validez depende de este hecho. Lo mismo ocurre con el ejemplo (4) y (5), por ejemplo. Decimos que (1), (4) y (5) tienen una misma forma en común, y es esta forma la que es responsable de su validez. Esta forma común puede representarse esquemáticamente así:

(6) A o B, No A, B

Estas representaciones esquemáticas de los argumentos se llaman esquemas argumentales. Las letras A y B representan enunciados arbitrarios. Al sustituir estas letras por enunciados reales, obtenemos un argumento real. Cualquier sustitución de este tipo que hagamos en el esquema (6) resultará en un argumento deductivo, por eso decimos que (6) es un esquema argumental deductivo o válido.

Argumento: Conjunto de fórmulas para el razonamiento lógico.

Argumento Válido: Un argumento es válido si se cumple:

Un argumento puede ser válido con premisas y conclusión verdaderas.

Pero también puede ser válido con premisas falsas y conclusión verdadera, o incluso con premisas y conclusión falsas.

Lo que NUNCA será es válido con premisas verdaderas y conclusión falsa.

Ejemplo 1:

$p \rightarrow (q \vee \neg r), \neg q, p \models \neg r$

P	2	3	C	5	P	P
p	q	r	$\neg r$	$q \vee \neg r$	$p \rightarrow (q \wedge \neg r)$	$\neg q$
V	V	V	F	V	V	F
V	V	F	V	V	V	F
V	F	V	F	F	F	V
V	F	F	V	V	V	V
F	V	V	F	V	V	F
F	V	F	V	V	V	F
F	F	V	F	F	V	V
F	F	F	V	V	V	V

3.1.7 Demostración formal

Demostración Formal (Directa, Por contradicción)

Contradicción:

Contradicción es aquella proposición que siempre es falsa para todos los valores de verdad, una de las más usadas y más sencilla es $p \wedge p'$.

Como lo muestra su correspondiente tabla de verdad.

p	p'	$p \wedge p'$
0	0	1
1	0	0

Ejemplo:

Si se tiene p : “El coche es verde”, la proposición $p \wedge p'$ equivale a decir que

“El coche es verde y el coche no es verde”. Por lo tanto se está contradiciendo, es decir, es una falacia.

3.2 Lógica de predicados

La lógica matemática es una parte de la lógica y las matemáticas, que consiste en el estudio matemático de la lógica y en la aplicación de este estudio a otras áreas de las matemáticas. La lógica matemática tiene estrechas conexiones con las ciencias de la computación y la lógica filosófica.

La lógica matemática estudia los sistemas formales en relación con el modo en el que codifican nociones intuitivas de objetos matemáticos como conjuntos, números, demostraciones y computación.

La lógica de predicados estudia las frases declarativas con mayor grado de detalle, considerando la estructura interna de las proposiciones. Se tomara como elemento básico los objetos y las relaciones entre dichos objetos. Es decir, se distingue:

Que se afirma (predicado o relación)

De quien se afirma (objeto)

Definimos a continuación las reglas sintácticas para construir fórmulas:

Definición 1: El alfabeto de la lógica de predicados estará formado por los siguientes conjuntos simbólicos:

- Conjunto de Símbolos de Variables (VAR): Es un conjunto de las últimas letras del alfabeto en minúsculas. Se utilizan subíndices, por ejemplo:
- Conjunto de símbolos de Constantes (CONS): Este conjunto lo forman las primeras letras del alfabeto en minúsculas, también utilizaremos subíndices:
- Conjunto de letras de función (FUNC): Representaremos a este conjunto por las letras f, g, h, L. Incluimos subíndices para poder diferenciar las funciones:
- Conjunto de letras de Predicado (PRED): Se representan mediante letras mayúsculas,

Símbolos de conectivas:

\neg = Negación

\vee = Conectiva "o"

\wedge = Conectiva "y"

\rightarrow = implicación

\leftrightarrow = Doble implicación o equivalencia

Cuantificadores:

\exists =existencial

\forall =Universal

3.2.1 Cuantificadores

El cuantificador universal: indica que algo es cierto para todos los individuos.

- Sea A una expresión y sea x una variable. Si deseamos indicar que A es verdadero para todos los posibles valores de x, escribiremos $(\forall x) A$.
- $(\forall x)$ es cuantificador universal.
- A es el ámbito (alcance) del cuantificador.
- El símbolo \forall se lee "para todo".

Ejemplo:

- Todo el mundo tiene buena suerte de vez en cuando.

$B \equiv$ «tener buena suerte de vez en cuando»

$B(x) \equiv$ «x tiene buena suerte de vez en cuando»

$\forall x B(x)$ en el conjunto de los seres humanos.

Cuantificador Existencial: Sea A una expresión y x una variable. Si deseamos indicar que A es verdadero para al menos un valor de la variable x, escribiremos $\exists x A$.

\exists se denomina cuantificador existencial, y A es el ámbito o alcance del cuantificador existencial.

Ejemplo:

- Hay una persona que ha irrumpido en el aula con malos modales.

$B \equiv$ «irrumpir en el aula con malos modales»

$B(x) \equiv$ «x irrumpe en el aula con malos modales»

$\exists x B(x)$ en el conjunto de los seres humanos.

3.2.2 Representación y evaluación de predicados

La principal debilidad de la lógica proposicional es su limitada habilidad para expresar conocimiento. Existen varias sentencias complejas que pierden mucho de su significado cuando se las representa en lógica proposicional. Por esto se desarrolló una forma lógica más general, capaz de representar todos los detalles expresados en las sentencias, esta es la ***lógica de predicados***.

Al igual que las proposiciones, los predicados tienen un valor de veracidad, pero a diferencia de las proposiciones, su valor de veracidad, depende de sus términos. Es decir, un predicado puede ser verdadero para un conjunto de términos, pero falso para otro.

La lógica de predicados, se ocupa únicamente de métodos de argumentación sólidos. Tales argumentaciones se denominan ***Reglas de Inferencia***. Si se da un conjunto de axiomas que son aceptados como verdaderos, las reglas de inferencia garantizan que sólo serán derivadas consecuencias verdaderas.

- El cuantificador universal; « indica que la fórmula bien formada, dentro de su alcance, es verdadera para todos los valores posibles de la variable que es cuantificada. Por ejemplo:

« X. . . .

Establece que «para todo X, es verdad que. . . «

- El cuantificador existencial; \$, indica que la fórmula bien formada, dentro de su alcance, es verdadera para algún valor o valores dentro del dominio. Por ejemplo:

\$ X. . . .

Establece que «existe un X, tal que. . . «

Ejemplos de predicados cuantificados:

« X, [*niño* (X) => *le_gusta* (X, *helados*)].

« $Y, [\text{mamífero}(Y) \Rightarrow \text{nace}(Y, \text{vivo})]$.

\$ $Z, [\text{cartero}(Z) \wedge \text{mordió}(\text{boby}, Z)]$.

3.3 Álgebra declarativa

Lo que algunos llaman álgebra declarativa no es otra cosa que el álgebra proposicional, o sea, la estructura algebraica que se forma con expresiones utilizando los conectivos lógicos.

Empezaremos por definir formalmente cómo se construye una fórmula en lógica. Una expresión sintácticamente correcta se le llama fórmula bien formada (fbf) o simplemente fórmula y su definición es:

Una fórmula en lógica de proposiciones se obtiene al aplicar una ó más veces las siguientes reglas:

(B) si p es una proposición lógica, es una fbf.

(R) si F es una fórmula bien formada (fbf) también lo es $(\neg F)$.

(R) si p, q son fbf entonces también lo es $(p * q)$ donde $*$ es uno de los operadores binarios, $\wedge \vee \rightarrow \leftrightarrow$.

3.4 Inducción matemática

El método deductivo, muy usado en matemática, obedece a la siguiente idea: “A partir de un cierto conjunto de axiomas aceptados sin demostración y de reglas lógicas no contradictorias, se deducen otros enunciados llamados teoremas combinando los axiomas y respetando en cada etapa las reglas lógicas”.

Otro método para demostrar resultados generales que dependen en algún sentido de los números naturales es conocido con el nombre de Inducción Matemática.

Esta dependencia de los números naturales significa: se sabe que una determinada afirmación es verdadera para algunos casos particulares y surge la pregunta. ¿Dicha afirmación sigue siendo verdadera para los infinitos números naturales restante?

Existen muchas afirmaciones que solo son válidas para un número finito de casos y en consecuencia son falsas para un número infinito de situaciones. Sin embargo, podemos encontrar proposiciones (afirmaciones) que son verdaderas solo a partir de un cierto número natural n_0 , de ser así, la técnica que se desarrollaremos se llama Inducción Incompleta. Para demostrar que una proposición $p(n)$, $\forall n \in M \subseteq \mathbb{N}$, Es verdadera es necesario comprobar la validez de ella para todos los elementos del conjunto M . En el caso en que $M = \mathbb{N}$, diremos que es una Inducción Completa.

Si se requiere demostrar la falsedad de una cierta proposición $p(n)$, $\forall n \in M \subseteq \mathbb{N}$, Es suficiente indicar un elemento particular $m \in M$ de manera que $p(m)$ sea falsa. (Construcción de un contra ejemplo).

Ejemplo 1. $\forall n \in \mathbb{N}, n^2 - 3n - 1 < 0$

Es fácil probar que esta desigualdad es verdadera para $n = 1, 2, 3$. Sin embargo, para $n = 4$ no se cumple ya que 4

$2 - 3 \cdot 4 - 1 = 3 > 0$. Nótese que este ejemplo sencillo muestra que una proposición puede ser verdadera para los primeros números naturales, sin embargo, es falsa, para números naturales más grandes.

3.5 Aplicaciones de la lógica matemática en la computación

La lógica computacional es la misma lógica matemática aplicada al contexto de las ciencias de la computación. Su uso es fundamental a varios niveles: en los circuitos computacionales, en la programación lógica y en el análisis y optimización (de recursos temporales y espaciales) de algoritmos.

Circuitos computacionales

El nivel menos abstracto dentro de una computadora está constituido por circuitos electrónicos que responden a diferentes señales eléctricas, siguiendo los patrones de la lógica booleana; esto es, compuertas lógicas que devuelven un valor dependiendo de las entradas que se le dan al sistema. Existen ocho compuertas lógicas básicas con las cuales se pueden formar sistemas muy complejos: AND, OR, Inverter, Buffer, NAND, NOR, XOR y XNOR.

Algoritmos

En matemáticas, ciencias de la computación y disciplinas relacionadas, un algoritmo (del griego y latín, dixit algorithmus y éste a su vez del matemático persa Al Juarismi) es un conjunto preescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.

Programación lógica

La programación lógica consiste en la aplicación del corpus de conocimiento sobre lógica para el diseño de lenguajes de programación; La programación lógica es un tipo de paradigmas de programación dentro del paradigma de programación declarativa. El resto de los subparadigmas de programación dentro de la programación declarativa son: programación funcional, programación basada en restricciones, programas DSL (de dominio específico) e híbridos.

Unidad 4

ÁLGEBRA BOOLEANA

4.1 - Teoremas y postulados.

El álgebra booleana es un álgebra, que trata con números binarios y variables binarias. Por lo tanto, también se le llama Álgebra binaria o Álgebra lógica. Un matemático, llamado George Boole, había desarrollado esta álgebra en 1854. Las variables utilizadas en esta álgebra también se denominan variables booleanas.

El rango de voltajes correspondiente a la lógica 'Alta' se representa con '1' y el rango de voltajes correspondiente a la lógica 'Baja' se representa con '0'.

Postulados y leyes básicas del álgebra booleana

En esta sección, analicemos sobre los postulados booleanos y las leyes básicas que se utilizan en el álgebra booleana. Estos son útiles para minimizar las funciones booleanas.

Postulados Booleanos

Considere los números binarios 0 y 1, la variable booleana (x) y su complemento (x'). La variable booleana o su complemento se conocen como literal. Las cuatro operaciones lógicas O posibles entre estos literales y números binarios se muestran a continuación.

$$X + 0 = x$$

$$X + 1 = 1$$

$$x + x = x$$

$$x + x' = 1$$

Del mismo modo, a continuación se muestran las cuatro operaciones lógicas Y posibles entre esos literales y números binarios.

$$x.1 = x$$

$$x.0 = 0$$

$$Xx = x$$

$$x.x' = 0$$

Estos son los simples postulados booleanos. Podemos verificar estos postulados fácilmente, sustituyendo la variable booleana por '0' o '1'.

Nota: el complemento de complemento de cualquier variable booleana es igual a la variable en sí. Es decir, $(x')' = x$.

Leyes básicas del álgebra booleana

Las siguientes son las tres leyes básicas del álgebra booleana.

- Ley conmutativa
- Ley asociativa
- Ley distributiva

Ley conmutativa

Si cualquier operación lógica de dos variables booleanas da el mismo resultado independientemente del orden de esas dos variables, entonces esa operación lógica se dice que es conmutativa. Las operaciones lógicas OR y lógicas AND de dos variables booleanas x & y se muestran a continuación

$$x + y = y + x$$

$$xy = yx$$

El símbolo '+' indica una operación OR lógica. Del mismo modo, el símbolo '.' indica una operación AND lógica y es opcional para representar. La ley conmutativa obedece a las operaciones lógicas x & lógicas y .

Ley asociativa

Si primero se realiza una operación lógica de cualquiera de las dos variables booleanas y luego se realiza la misma operación con la variable restante da el mismo resultado, entonces se dice que la operación lógica es asociativa. Las

operaciones lógicas OR y lógicas AND de tres variables booleanas x, y & z se muestran a continuación.

$$x + (y + z) = (x + y) + z$$

$$x \cdot (yz) = (xy) \cdot z$$

La ley asociativa obedece a las operaciones lógicas y lógicas y operaciones AND.

Ley distributiva

Si cualquier operación lógica se puede distribuir a todos los términos presentes en la función booleana, entonces se dice que esa operación lógica es Distributiva. La distribución de las operaciones lógicas OR y lógicas AND de tres variables booleanas x, y & z se muestra a continuación.

$$x \cdot (y + z) = xy + xz$$

$$x + (yz) = (x + y) \cdot (x + z)$$

La ley distributiva obedece a operaciones lógicas y lógicas y operaciones AND.

Estas son las leyes básicas del álgebra booleana. Podemos verificar estas leyes fácilmente, sustituyendo las variables booleanas con '0' o '1'.

Teoremas del álgebra booleana

Los siguientes dos teoremas se utilizan en el álgebra de Boole.

- Teorema de dualidad
- Teorema de De Morgan

Teorema de dualidad

Este teorema establece que el doble de la función booleana se obtiene al intercambiar el operador lógico AND con el operador lógico OR y los ceros con los unos. Para cada función booleana, habrá una función dual correspondiente.

Hagamos las ecuaciones booleanas (relaciones) que discutimos en la sección de postulados booleanos y las leyes básicas en dos grupos. La siguiente tabla muestra estos dos grupos.

Grupo 1	Grupo 2
$x + 0 = x$	$x \cdot 1 = x$
$x + 1 = 1$	$x \cdot 0 = 0$

$x + x = x$	$xx = x$
$x + x' = 1$	$x.x' = 0$
$x + y = y + x$	$xy = yx$
$x + (y + z) = (x + y) + z$	$x.(yz) = (xy).z$
$x.(y + z) = xy + xz$	$x + (yz) = (x + y).(x + z)$

En cada fila, hay dos ecuaciones booleanas y son duales entre sí. Podemos verificar todas estas ecuaciones booleanas de Grupo 1 y Grupo 2 usando el teorema de dualidad.

Teorema de De Morgan

Este teorema es útil para encontrar el complemento de la función booleana. Indica que el complemento de OR lógico de al menos dos variables booleanas es igual al AND lógico de cada variable complementada.

El teorema de De Morgan con 2 variables booleanas x e y puede representarse como

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

El doble de la función booleana anterior es

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

Por lo tanto, el complemento de AND lógico de dos variables booleanas es igual al OR lógico de cada variable complementada. De manera similar, podemos aplicar el teorema de De Morgan para más de 2 variables booleanas también.

4.2 Optimización de expresiones booleanas.

Utilizando expresiones booleanas, vamos a definir Funciones booleanas, que son exactamente iguales a las funciones matemáticas a las que estamos acostumbrados pero con la particularidad de que las variables son booleanas y que los valores devueltos por la función también son booleanos, es decir, una función booleana sólo puede tomar los valores '0' o '1'

Como hemos hecho antes, vamos a ver un ejemplo utilizando una función matemática de las que todos conocemos. Por ejemplo esta:

$$F(x) = x^2 + 1$$

Se trata de una función Real que tiene una variable Real (x). Para cada valor de x, obtenemos el valor de la función. Así por ejemplo podemos calcular el siguiente:

- $f(0) = 1$
- $f(1) = 2$
- $f(2) = 5$
- $f(3) = 10$

A

Como es una función Real, obtenemos como valores de la función Números Reales. También podemos definir funciones reales de 2 o más variables, como por ejemplo:

- $f(x, y) = x*y + 3$ Función de 2 variables
- $g(x, y, z) = x*y + z$ Función de 3 variables

Como estamos acostumbrados a trabajar con este tipo de funciones, nos resultan sencillas. Ahora vamos a definir **funciones booleanas**. Para ello hay que tener en mente que trabajaremos con variables booleanas y que por tanto usaremos las operaciones + y * del **Algebra de Boole**, y que como ya sabemos, nada tienen que ver con las operaciones suma y producto a las que estamos habituados. Por ejemplo, sea la siguiente función booleana de una variable:

$$F(A) = \bar{A}$$

El valor devuelto por la función es el negado del que se le pasa por la variable. Como la variable A es booleana, sólo puede tomar los valores "0" y "1". Los que la función F toma son:

$$F(0) = \bar{0} = 1$$

$$F(1) = \bar{1} = 0$$

Vamos a definir una función un poco más compleja, usando dos variables booleanas, A y B

$$F(A, B) = (A + B) * \bar{B}$$

¿Cuándo vale F(0,0)? sólo hay que sustituir en la función los valores de A y B por "0", obteniéndose:

$$F(0, 0) = (0 + 0) * \bar{0} = 0 * 1 = 0$$

Calcularemos el valor de F para el resto de valores de entrada de A y B:

$$F(0, 0) = (0 + 0) * \bar{0} = 0 * 1 = 0$$

Calcularemos el valor de F para el resto de valores de entrada de A y B:

$$F(0, 1) = (0 + 1) * \bar{1} = 1 * 0 = 0$$

$$F(1, 0) = (1 + 0) * \bar{0} = 1 * 1 = 1$$

$F(1, 1) = (0 + 0) * \bar{0} = 0 * 1 = 0$ Se deja como ejercicio para practicar
(La solución es 0).

Fijándonos en esta función tan sencilla, podemos darnos cuenta de varias cosas:
1. Puesto que las variables de entrada A y B, sólo pueden tomar los valores "0" y "1", hay 4 casos distintos:

$$a) A = 0, B = 0 \Rightarrow F(0, 0) = 0$$

$$b) A = 0, B = 1 \Rightarrow F(0, 1) = 0$$

$$c) A = 1, B = 0 \Rightarrow F(1, 0) = 1$$

$$d) A = 1, B = 1 \Rightarrow F(1, 1) = 0$$

2. Antes de calcular los valores que toma la función, según lo que valgan A y B, se pueden aplicar algunas propiedades para obtener una función más simplificada.

$$F(A, B) = (A + B) * \bar{B} = \{\text{Aplicando la propiedad distributiva}\} = A * \bar{B} + B * \bar{B} = A * \bar{B}$$

Es más sencillo trabajar con esta función simplificada: $F(A, B) = A * \bar{B}$

Las funciones booleanas pueden ser de muchas más variables, como en los siguientes ejemplos:

$$F(x, y, z) = x * y + z. \text{ Función booleana de 3 variables}$$

$$F(A, B, C, D) = \bar{A} * B + C * \bar{D} \text{ Función booleana de 4 variables}$$

$$F(E_4, E_3, E_2, E_1, E_0) = \bar{E}_4 * \bar{E}_3 * E_2 * E_1 * \bar{E}_0 \text{ Función booleana de 5 variables}$$

Por cuestiones de comodidad, muchas veces no escribimos entre paréntesis las variables de la función, así por ejemplo podemos definir una función de 3 variables de la siguiente manera:

$$F = \bar{A} * B + \bar{C}$$

4.3 Aplicación del algebra booleana (Compuertas lógicas)

El álgebra booleana como cálculo de dos valores es fundamental para los circuitos de computadora, la programación de computadoras y la lógica matemática, y también se usa en otras áreas de las matemáticas, como la teoría de conjuntos y las estadísticas.

Computadoras: A principios del siglo XX, varios ingenieros eléctricos reconocieron intuitivamente que el álgebra de Boole era análogo al comportamiento de ciertos tipos de circuitos eléctricos. Claude Shannon demostró formalmente que tal comportamiento era lógicamente equivalente al álgebra booleana en su tesis de maestría de 1937, Un análisis simbólico de relés y circuitos de conmutación.

Hoy en día, todas las computadoras modernas de propósito general realizan sus funciones utilizando lógica booleana de dos valores; es decir, sus circuitos eléctricos son una manifestación física de la lógica booleana de dos valores. Lo logran de varias maneras: como voltajes en cables en circuitos de alta velocidad y dispositivos de almacenamiento capacitivos, como orientaciones de un dominio magnético en dispositivos de almacenamiento ferromagnéticos, como agujeros en tarjetas perforadas o cinta de papel, y así sucesivamente. (Algunas computadoras antiguas utilizaban circuitos o mecanismos decimales en lugar de circuitos lógicos de dos valores).

Por supuesto, es posible codificar más de dos símbolos en cualquier medio dado. Por ejemplo, uno podría usar respectivamente 0, 1, 2 y 3 voltios para codificar un alfabeto de cuatro símbolos en un cable, o agujeros de diferentes tamaños en una tarjeta perforada. En la práctica, las restricciones estrictas de alta velocidad, pequeño tamaño y baja potencia se combinan para hacer del ruido un factor importante. Esto hace que sea difícil distinguir entre símbolos cuando hay varios símbolos posibles que podrían aparecer en un solo sitio. En lugar de intentar distinguir entre cuatro voltajes en un cable, los diseñadores digitales se han establecido en dos voltajes por cable, alto y bajo.

Las computadoras usan circuitos booleanos de dos valores por las razones anteriores. Las arquitecturas de computadoras más comunes usan secuencias ordenadas de valores booleanos, llamados bits, de 32 o 64 valores, por ejemplo, 011010001101011101010101010101011. Al programar en código de máquina, lenguaje ensamblador y algunos otros lenguajes de programación, los programadores trabajan con la estructura digital de bajo nivel del registro de datos. Estos registros funcionan con voltajes, donde cero voltios representan 0 booleano, y un voltaje de referencia (a menudo + 5V, + 3.3V, + 1.8V) representa booleano 1. Estos lenguajes admiten operaciones numéricas y operaciones lógicas. En este contexto, "numérico" significa que la computadora trata las secuencias de bits como números binarios (base dos números) y ejecuta operaciones aritméticas como sumar, restar, multiplicar o dividir. "Lógico" se refiere a las operaciones lógicas booleanas de disyunción, conjunción y negación entre dos secuencias de bits, en las que cada bit en una secuencia se compara simplemente con su contraparte en

la otra secuencia. Por lo tanto, los programadores tienen la opción de trabajar y aplicar las reglas de álgebra numérica o álgebra booleana según sea necesario. Una característica diferenciadora principal entre estas familias de operaciones es la existencia de la operación de acarreo en la primera, pero no en la segunda.

Lógica de dos valores

Otras áreas donde dos valores son una buena opción son la ley y las matemáticas. En la conversación relajada de todos los días, se aceptan respuestas complejas o con matices como "quizás" o "solo los fines de semana". En situaciones más específicas, como un tribunal de justicia o matemáticas basadas en teoremas, se considera ventajoso formular preguntas para admitir una respuesta simple de sí o no: si el acusado es culpable o no culpable, si la proposición es verdadera o falsa. —Y no permitir ninguna otra respuesta. Por muy importante que sea la camisa de fuerza que esto pueda demostrar en la práctica para el encuestado, el principio de la simple pregunta de sí o no se ha convertido en una característica central de la lógica tanto matemática como judicial, haciendo que la lógica de dos valores merezca organización y estudio por derecho propio.

Un concepto central de la teoría de conjuntos es la pertenencia. Ahora, una organización puede permitir múltiples grados de membresía, como novatos, asociados y completos. Sin embargo, con conjuntos, un elemento está dentro o fuera. Los candidatos para ser miembros de un conjunto funcionan igual que los cables de una computadora digital: cada candidato es miembro o no miembro, así como cada uno de los cables es alto o bajo.

Al ser el álgebra una herramienta fundamental en cualquier área susceptible de tratamiento matemático, estas consideraciones se combinan para hacer que el álgebra de dos valores sea de fundamental importancia para el hardware de computadora, la lógica matemática y la teoría de conjuntos.

La lógica de dos valores se puede extender a la lógica de valores múltiples, en particular reemplazando el dominio booleano $\{0, 1\}$ con el intervalo de la unidad $[0, 1]$, en cuyo caso, en lugar de solo tomar los valores 0 o 1, cualquier valor entre e incluyendo 0 y 1 pueden ser asumidos. Algebraicamente, la negación (NO) se reemplaza con $1 - x$, la conjunción (AND) se reemplaza con la multiplicación (xy), y la disyunción (OR) se define a través de la ley de De Morgan. La interpretación de estos valores como valores de verdad lógica produce una lógica de múltiples valores, que forma la base de la lógica difusa y la lógica probabilística. En estas interpretaciones, un valor se interpreta como el "grado" de verdad, en qué medida una proposición es verdadera o la probabilidad de que la proposición sea verdadera.

Operaciones booleanas

La aplicación original para operaciones booleanas fue la lógica matemática, donde combina los valores de verdad, verdaderos o falsos, de fórmulas individuales.

Los lenguajes naturales como el inglés tienen palabras para varias operaciones booleanas, en particular conjunción (y), disyunción (o), negación (no) e implicación (implica). Pero no es sinónimo de y no. Cuando se usan para combinar afirmaciones situacionales como "el bloque está sobre la mesa" y "los gatos beben leche", que ingenuamente son verdaderos o falsos, los significados de estos conectivos lógicos menudos tienen el significado de sus contrapartes lógicas. Sin embargo, con descripciones de comportamientos como "Jim caminó por la puerta", uno comienza a notar diferencias como la falta de conmutación, por ejemplo, la conjunción de "Jim abrió la puerta" con "Jim entró por la puerta" en ese orden es no es equivalente a su conjunción en la otra orden, desde y generalmente significa y luego en tales casos. Las preguntas pueden ser similares: el orden "¿Es el cielo azul y por qué el cielo es azul?" Tiene más sentido que el orden inverso. Los comandos conjuntivos sobre el comportamiento son como afirmaciones de comportamiento, como vestirse e ir a la escuela. Comandos disyuntivos como amarme o dejarme o pescar o cortar cebo tienden a ser asimétricas por la implicación de que una alternativa es menos preferible. Los sustantivos unidos como el té y la leche generalmente describen la agregación como con la unión establecida, mientras que el té o la leche es una opción. Sin embargo, el contexto puede revertir estos sentidos, ya que entre sus opciones están el café y el té, lo que generalmente significa lo mismo que sus opciones son el café o el té (alternativas). La doble negación, como en "No me gusta la leche", rara vez significa literalmente "Me gusta la leche", sino que conlleva algún tipo de cobertura, como para dar a entender que existe una tercera posibilidad. "No no P" se puede interpretar libremente como "seguramente P", y aunque P necesariamente implica "no P" "Lógica intuicionista. En vista del uso altamente idiosincrásico de las conjunciones en lenguajes naturales, el álgebra de Boole no puede considerarse un marco confiable para interpretarlos.

Las operaciones booleanas se utilizan en lógica digital para combinar los bits transportados en cables individuales, interpretándolos así en $\{0,1\}$. Cuando se utiliza un vector de n compuertas binarias idénticas para combinar dos vectores de bit cada uno de n bit, las operaciones de bit individuales pueden entenderse colectivamente como una sola operación en valores de un álgebra booleana con 2^n elementos.

La teoría de conjuntos ingenua interpreta las operaciones booleanas como acciones en subconjuntos de un conjunto X determinado. Como vimos anteriormente, este comportamiento es exactamente paralelo a las combinaciones de coordenadas de los vectores de bits, con la unión de dos conjuntos correspondientes a la disyunción de dos vectores de bits y así sucesivamente.

El álgebra booleana libre de 256 elementos en tres generadores se implementa en pantallas de computadora basadas en gráficos de trama, que utilizan bit blit para manipular regiones enteras que consisten en píxeles, y dependen de las

operaciones booleanas para especificar cómo se debe combinar la región de origen con el destino, normalmente Con la ayuda de una tercera región llamada la máscara. Modernas tarjetas de vídeo ofrecen todo $2^2 \cdot 2^3 = 256$ operaciones ternarias para este propósito, con la opción de operación como un parámetro de un byte (8 bits). Las constantes SRC = 0xaa o 10101010, DST = 0xcc o 11001100, y MSK = 0xf0 o 11110000 permiten operaciones booleanas como (SRC ^ DST) y MSK (que significa XOR el origen y el destino y luego Y el resultado con la máscara) para ser escrito directamente como una constante que denota un byte calculado en tiempo de compilación, 0x60 en el ejemplo (SRC ^ DST) y MSK, 0x66 si solo SRC ^ DST, etc. En el tiempo de ejecución, la tarjeta de video interpreta el byte como la operación raster indicada por la expresión original de una manera uniforme que requiere un hardware notablemente pequeño y que lleva tiempo completamente independiente de la complejidad de la expresión.

Los sistemas de modelado sólido para diseño asistido por computadora ofrecen una variedad de métodos para construir objetos a partir de otros objetos, uno de ellos es la combinación de operaciones booleanas. En este método, el espacio en el que existen los objetos se entiende como un conjunto S de voxels (el análogo tridimensional de los píxeles en gráficos bidimensionales) y las formas se definen como subconjuntos de S, permitiendo que los objetos se combinen como conjuntos a través de unión, intersección, etc. Un uso obvio es en la construcción de una forma compleja a partir de formas simples simplemente como la unión de esta última. Otro uso es en la escultura entendida como remoción de material: cualquier operación de rectificado, fresado, fresado o taladrado que se puede realizar con maquinaria física en materiales físicos puede simularse en la computadora con la operación booleana $x \wedge \neg y$ o $x - y$, que en la teoría de conjuntos es la diferencia de conjuntos, elimina los elementos de y de los de x. Por lo tanto, dadas las dos formas, una para ser mecanizada y la otra para el material a remover, el resultado de mecanizar la primera para remover la última se describe simplemente como su diferencia establecida.

4.3.1 Mini y maxi términos.

Minitérminos

Para una función booleana de n variables x_1, \dots, x_n , un producto booleano en el que cada una de las n variables aparece una sola vez (negada o sin negar) es llamado minterms. Es decir, un minterm es una expresión lógica de n variables consistente únicamente en el operador conjunción lógica (AND) y el operador complemento o negación (NOT).

Por ejemplo, abc , $ab'c$ y abc' son ejemplos de minterms para una función booleana con las tres variables a , b y c .

En general, uno asigna a cada minterm (escribiendo las variables que lo componen en el mismo orden), un índice basado en el valor binario del minterm. Un término negado, como a' es considerado como el número binario 0 y el término no negado a es considerado como un 1. Por ejemplo, se asociaría el número 6 con $a b c'$ (1102), y nombraríamos la expresión con el nombre m_6 . Entonces m_0 de tres variables es $a'b'c'$ (0002) y m_7 debería ser abc (1112).

Función equivalente

Se puede observar que cada minterm solo devuelve 'verdadero' con una sola entrada de las posibles. Por ejemplo, el minterm 5, $a b' c$, es verdadero solo cuando a y c son ciertos y b es falso - la entrada $a = 1, b = 0, c = 1$ da resultado 1.

Si tenemos una tabla de verdad de una función lógica, es posible escribir la función como "suma de productos". Por ejemplo, dada la tabla de verdad

A	b	f(a, b)
0	0	1
0	1	0
1	0	1
1	1	0

Observamos que las filas con resultado 1 son la primera y la tercera, entonces podremos escribir f como la suma de los minterms m_0 y m_2 .

Si queremos verificar esto:

$$F(a, b) = m_0 + m_2 = (a'b') + (ab')$$

Tendremos que la tabla de verdad de la función, calculándola directamente, será la misma.

Maxitérminos

Un maxterm es una expresión lógica de n variable que consiste únicamente en la disyunción lógica y el operador complemento o negación. Los maxterms son una expresión dual de los minterms. En vez de usar operaciones AND utilizamos operaciones OR y procedemos de forma similar.

Por ejemplo, los siguientes son maxterms:

$$a+b'+c$$

$$a'+b+c$$

El complemento de un minterm es su respectivo maxterm. Esto puede ser fácilmente verificado usando la Ley de Morgan. Por ejemplo:

$$m_1' = M_1$$

$$(a'b)' = a+b'$$

Para indexar maxterms lo haremos justo de la forma contraria a la que seguimos con los minterms. Se asigna a cada maxterm un índice basado en el complemento del número binario que representa (otra vez asegurándonos que las variables se escriben en el mismo orden, usualmente alfabético). Por ejemplo, podemos

asignar M6 (Maxterm 6) al maxterm $a'+b'+c$. De forma similar M0 de tres variables debería ser $a+b+c$ y M7 es $a'+b'+c'$.

Función equivalente

Se puede ver fácilmente que un maxterm sólo da como resultado un cero para una única entrada de la función lógica. Por ejemplo, el maxterm 5, $a'+b+c'$, es falso solo cuando a y c son ciertos y b es falso - la entrada $a = 1, b = 0, c = 1$ da como resultado un cero.

Si tenemos una tabla de verdad de una función lógica, es posible escribir la función como "producto de sumas". Por ejemplo, dada la tabla de verdad

a	b	f(a, b)
0	0	1
0	1	0
1	0	1
1	1	0

Observamos que las filas que tiene como salida un 0 son la segunda y la cuarta, entonces podemos escribir f como un producto de maxterms M1 y M3.

Si queremos verificar esto:

$$F(a, b) = M1 M3 = (a+b') (a'+b')$$

Tendremos que la tabla de verdad de la función, calculándola directamente, será la misma.

Mapa de Karnaugh

Otra manera de simplificar funciones es representándolas en mapas de Karnaugh. Esto es equivalente a resolver las simplificaciones por teoremas. Sin embargo, mucha gente considera que resulta más fácil visualizar las simplificaciones si se presentan gráficamente.

Los mapas de Karnaugh pueden aplicarse a dos, tres, cuatro y cinco variables. Para más variables, la simplificación resulta tan complicada que conviene en ese caso utilizar teoremas mejor. Para efectos de clase, veremos las simplificaciones de dos, tres y cuatro variables.

Método de reducción de mapas de karnaugh

El Álgebra de Boole, resuelve problemas que dependiendo del número de términos que tenía la función canónica, siendo el número de compuertas lógicas utilizadas igual al número de términos obtenidos MÁS UNO; por lo tanto, los

circuitos obtenidos son de dos niveles de conmutación con un tiempo mínimo de retardo, pero que de ninguna manera es el más sencillo ni el más económico.

Los mapas de Karnaugh es uno de los métodos más prácticos. Se puede decir que es el más poderoso, cuando el número de variables de entrada es menor o igual a seis; más allá, ya no es tan práctico. En general, el mapa de Karnaugh se considera como la forma gráfica de una tabla de verdad o como una extensión del diagrama de Venn.

Antes de explicar cómo se utiliza el mapa de Karnaugh en la minimización de funciones, veremos cómo se obtiene el mapa. Esto nace de la representación geométrica de los números binarios. Un número binario de n bits, puede representarse por lo que se denomina un punto en un espacio N . Para entender lo que se quiere decir con esto, considérese el conjunto de los números binarios de un bit, es decir 0 o 1. Este conjunto puede representarse por dos puntos en un espacio 1; esto es, por dos puntos unidos por una línea.

Introducción de método de Quine-McCluskey

En matemáticas las expresiones booleanas se simplifican por numerosas razones:

- Una expresión más simple es más fácil de entender y tiene menos posibilidades de error a la hora de su interpretación.
- Una expresión simplificada suelen ser más eficiente y efectiva cuando se implementan en la práctica, como en el caso de circuitos eléctricos o en determinados algoritmos.

El método de Quine-McCluskey es particularmente útil cuando se tienen funciones con un gran número de variables, no es el caso del método de Karnaugh, que se hace impracticable con más de cinco variables. En nuestro caso, como el máximo número de variables será cuatro podremos utilizar conjuntamente ambos métodos.

Una expresión booleana se compone de variables y términos. Para este método las variables sólo podrán tener un valor numérico de cero (el correspondiente al valor de verdad false) o uno (el correspondiente al valor de verdad true) y se designarán mediante una letra.

Como notación se designará x si la variable contiene el valor uno, x' en caso de que contenga el valor cero.

Por otra parte, las variables se relacionarán entre sí únicamente mediante operaciones lógicas and para formar términos y mediante or para relacionarse con otros términos constituyendo una suma de productos. Ésta debe de ser canónica, es decir:

- Cada variable se usa una vez en cada término. A dichos términos se les llama términos canónicos.

Ejemplo $f(x, y, z) = x'y z + x y'z$

$X'y z$ se representa con 011, donde $x = 0, y = 1, z = 1$

$X y'z$ se representa con 101, donde $x = 1, y = 0, z = 1$

Circuito Combinacional

Un circuito combinacional, como su nombre lo sugiere es un circuito cuya salida depende solamente de la combinación de sus entradas en el momento que se está realizando la medida en la salida.

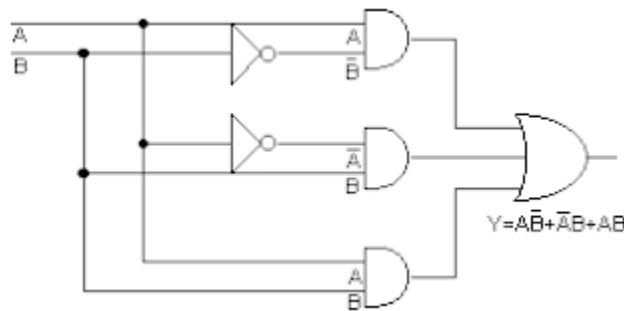
Analizando el circuito, con compuertas digitales, que se muestra a continuación, se puede ver que la salida de cada una de las compuertas que se muestra depende únicamente de sus entradas.

La salida F variará si alguna de las entradas A o B o las dos a la vez cambian.

4.3.2 Representación de expresiones booleanas con circuitos lógicos.

El álgebra de Boole permite la comprensión y facilita el manejo de diferentes dispositivos que manipulan señales eléctricas, tales como las compuertas y los circuitos lógicos.

Un bloque lógico es una representación simbólica gráfica de una o más variables de entrada a un operador lógico para obtener una señal de salida. En electrónica, estos bloques lógicos son las compuertas.



Las compuertas lógicas pueden recibir una o más señales de entrada.



En la compuerta anterior, A y B son señales que entran a la compuerta y pueden tener un valor de 1 ó 0 dependiendo de si existe o no la señal. Estos dos generan una sola salida, que también es 1 ó 0 dependiendo de la compuerta que se trate y de los valores de entrada.

Compuerta AND

Esta compuerta es representada por una multiplicación en el Algebra de Boole. Indica que es necesario que en todas sus entradas se tenga un estado binario 1 para que la salida otorgue un 1 binario. En caso contrario de que falte alguna de sus entradas con este estado o no tenga si quiera una accionada, la salida no podrá cambiar de estado y permanecerá en 0. Esta puede ser simbolizada por dos o más interruptores en serie de los cuales todos deben estar activos para que esta permita el flujo de la corriente.

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



$$Q = A * B$$

Compuerta OR

En el Algebra de Boole esta es una suma. Esta compuerta permite que con cualquiera de sus entradas que este en estado binario 1, su salida pasara a un estado 1 también. No es necesario que todas sus entradas estén accionadas para conseguir un estado 1 a la salida pero tampoco causa algún inconveniente. Para lograr un estado 0 a la salida, todas sus entradas deben estar en el mismo valor de 0. Se puede interpretar como dos interruptores en paralelo, que sin importar cual se accione, será posible el paso de la corriente.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



$$Q = A + B$$

Compuerta NOT

En este caso esta compuerta solo tiene una entrada y una salida y esta actúa como un inversor. Para esta situación en la entrada se colocará un 1 y en la salida otorgara un 0 y en el caso contrario esta recibirá un 0 y mostrara un 1. Por lo cual todo lo que llegue a su entrada, será inverso en su salida.

Q	Q'
0	1
1	0



$$Q = \bar{Q}$$

Compuerta NAND

También denominada como AND negada, esta compuerta trabaja al contrario de una AND ya que al no tener entradas en 1 o solamente alguna de ellas, esta concede un 1 en su salida, pero si esta tiene todas sus entradas en 1 la salida se presenta con un 0.

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0



$$Q = \overline{A * B}$$

Compuerta NOR

Así como vimos anteriormente, la compuerta OR también tiene su versión inversa. Esta compuerta cuando tiene sus entradas en estado 0 su salida estará en 1, pero si alguna de sus entradas pasa a un estado 1 sin importar en qué posición, su salida será un estado 0.

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0



$$Q = \overline{A + B}$$

Compuerta XOR

También llamada OR exclusiva, esta actúa como una suma binaria de un dígito cada uno y el resultado de la suma sería la salida. Otra manera de verlo es que con valores de entrada igual el estado de salida es 0 y con valores de entrada diferente, la salida será 1.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



$$Q = A * \bar{B} + \bar{A} * B$$

Compuerta XNOR

Esta es todo lo contrario a la compuerta XOR, ya que cuando las entradas sean iguales se presentará una salida en estado 1 y si son diferentes la salida será un estado 0.

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1



$$Q = A * B + \overline{A} * \overline{B}$$

Compuerta IF

Esta compuerta no es una muy utilizada o reconocida ya que su funcionamiento en estados lógicos es parecido a si solo hubiera un cable conectado porque exactamente lo que se le coloque en la entrada, se encontrara en la salida. Pero también es conocido como un buffer, en la práctica se utiliza como amplificador de corriente o como seguidor de tensión para adaptar impedancias.

Q	Q'
0	0
1	1



$$Q = Q$$

Unidad 5

5. Teoría de grafos

5.1 Elementos, características y componentes de los grafos.

ELEMENTOS Y CARACTERÍSTICAS.

Un grafo, G es un par ordenado de V y A , donde V es el conjunto de vértices o nodos del grafo y A es un conjunto de pares de vértices, a estos también se les llama arcos o ejes del grafo. Un vértice puede tener 0 o más aristas, pero toda arista debe unir exactamente a dos vértices. Los grafos representan conjuntos de objetos que no tienen restricción de relación entre ellos. Un grafo puede representar varias cosas de la realidad cotidiana, tales como mapas de carreteras, vías férreas, circuitos eléctricos, etc. La notación $G = A(V, A)$ se utiliza comúnmente para identificar un grafo. Los grafos se constituyen principalmente de dos partes: las aristas, vértices y los caminos que pueda contener el mismo grafo.

COMPONENTES.

Se compone principalmente de:

Aristas.

Son las líneas con las que se unen las aristas de un grafo y con la que se construyen también caminos. Si la arista carece de dirección se denota indistintamente $\{a, b\}$ o $\{b, a\}$, siendo a y b los vértices que une. Si $\{a, b\}$ es una arista, a los vértices a y b se les llama sus extremos.

Estas a su vez pueden ser:

Aristas Adyacentes: Se dice que dos aristas son adyacentes si convergen en el mismo vértice.

Aristas Paralelas: Se dice que dos aristas son paralelas si vértice inicial y el final son el mismo.

Aristas Cíclicas: Arista que parte de un vértice para entrar en el mismo.

Cruce: Son dos aristas que cruzan en un punto.

Vértices.

Son los puntos o nodos con los que está conformado un grafo. Llamaremos grado de un vértice al número de aristas de las que es extremo. Se dice que un vértice es 'par' o 'impar' según lo sea su grado.

Estos a su vez pueden ser:

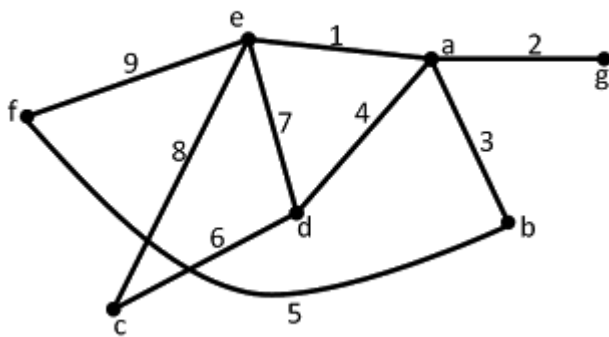
Vértices Adyacentes: Si tenemos un par de vértices de un grafo (U, V) y si tenemos una arista que los une, entonces U y V son vértices adyacentes y se dice que U es el vértice inicial y V el vértice adyacente.

Vértice Aislado: Es un vértice de grado cero. **Vértice Terminal:** Es un vértice de grado 1.

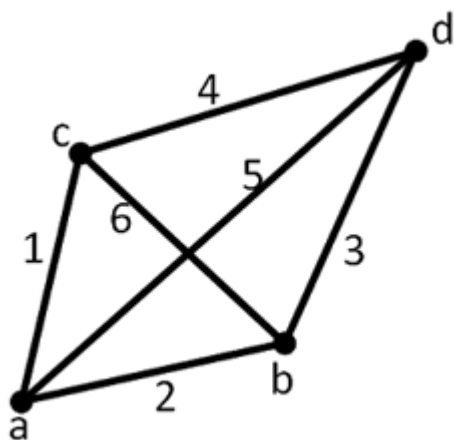
5.1.1 TIPOS DE GRAFOS.

Hay 6 tipos principales de grafos:

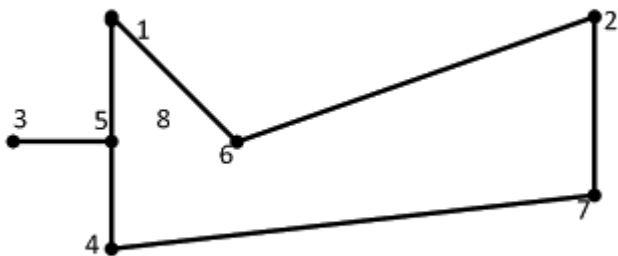
Grafo simple: Se dice que el grafo $G = (V, E)$ es un grafo simple de grado n si todos sus vértices tienen grado n .



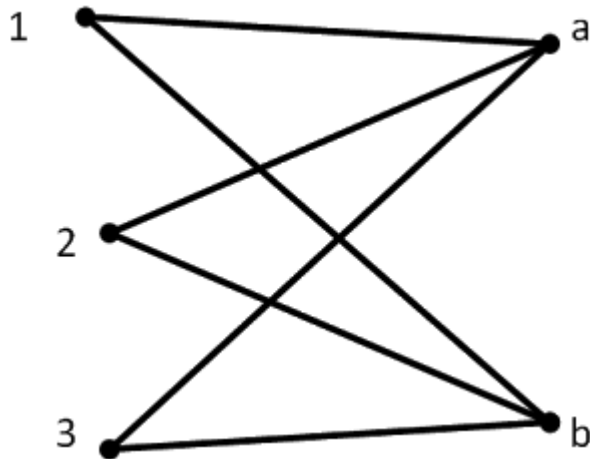
Grafo completo: Un grafo es completo si cada par de vértices está unido por una arista. Se denota por K_n al grafo completo de n vértices.



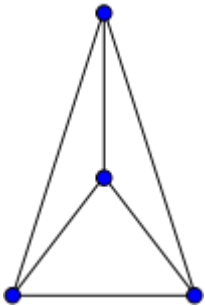
Grafo bipartido: Un grafo es bipartido si $V=V_1 \cup V_2$ y cada arista de E une un vértice de V_1 y otro de V_2 .



Grafo bipartido completo: Un grafo es bipartido completo si $V=V_1 \cup V_2$ y dos vértices de V están unidos por una arista de E si y solo si un vértice está en V_1 y el otro en V_2 . Se denota por $K_{r,s}$, donde V_1 tiene r vértices y V_2 tiene s vértices.

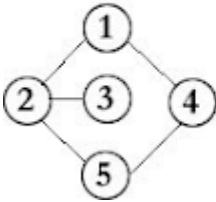


Grafos planos: Un grafo plano es aquel que puede ser dibujado en el plano sin que ninguna arista se intersecta.

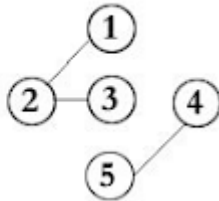


Grafos conexos: Un grafo es conexo si cada par de vértices está conectado por un camino; es decir, si para cualquier par de vértices (a , b), existe al menos un camino posible desde " a " hacia " b ".

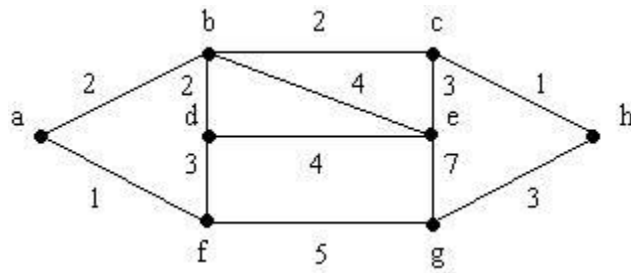
Grafo conexo



Grafo no conexo



Grafo ponderado: Un grafo es ponderado si presenta los pesos de cada arista y se puede determinar la longitud de una ruta, la cual es la suma de todos los pesos de las aristas.



5.2 Representación de los grafos.

Representación de grafos

Existen diferentes formas de representar un grafo, y hay muchos métodos para almacenarlos en una computadora. La estructura de datos usada dependerá de las características del grafo, y el algoritmo usado para manipularlo. Entre las más comunes esta las listas y matrices, con frecuencia se usa una combinación de ambas.

Estructura de lista

Lista de incidencia: El grafo está representado por una matriz de A (aristas) por V (vértices), donde [arista, vértice] contiene la información de la arista (conectado o no conectado).

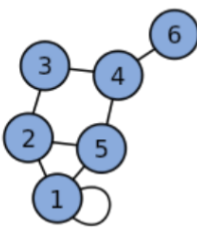
Lista de adyacencia: El grafo está representado por un arreglo de listas de adyacencia. Para un vértice i , la lista de adyacencia está formada por todos los vértices adyacentes a i . Puede construirse en tiempo lineal, y las inserciones pueden hacerse al principio de cada lista, con lo que se asegura tiempo constante.

Lista de grados: También llamada secuencia de grados o sucesión gráfica de una secuencia de números, que corresponde a los grados de los vértices del grafo.

Estructuras matriciales

Matriz de adyacencia: El grafo está representado por una matriz cuadrada M de tamaño n^2 , donde n es el número de vértices. Si hay una arista entre un vértice x y un vértice y , entonces el elemento $m_{\{x, y\}}$ es 1, de lo contrario, es 0.

Matriz de incidencia: El grafo está representado por una matriz de A (aristas) por V (vértices), donde [vértice, arista] contiene la información de la arista (1 - conectado, 0 - no conectado)

Grafo $G(V,A)$	Conjuntos	Matriz de adyacencia	Matriz de incidencia	Secuencia de grados	Lista de Adyacencia
	$V = \{1, 2, 3, 4, 5, 6\}$ $A = \{\{1,1\}, \{1,2\}, \{1,5\}, \{2,3\}, \{2,5\}, \{3,4\}, \{4,5\}, \{4,6\}\}$	$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	(4,3,3,3,2,1)	$\{\{1,2,5\}, \{3,5\}, \{4\}, \{5,6\}\}$

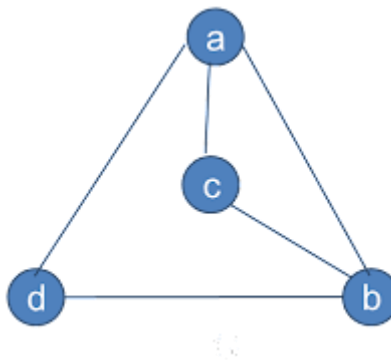
5.2.1 Matemática

Por medio de la teoría de los grafos podemos resolver diversos problemas, como la síntesis para circuitos secuenciales, contadores, o sistemas de apertura. Se utiliza en diferentes áreas por ejemplo, en las áreas de Sistemas y Computación, en áreas de ingeniería. También por medio de ellas podemos responder preguntas tales como, ¿Qué tarea debo hacer primero?, ¿Qué tiempo es más corto?, ¿Cuál es el

más barato?, y así podemos obtener caminos óptimos para las soluciones aplicando diversos algoritmos como puede ser el algoritmo de Floyd.

Un grafo **G** es un par **(V, E)** donde:

- o **V** = {**v**₁, ..., **v**_n} es un conjunto de vértices
 - o **E** = {**e**₁, ..., **e**_m} es un conjunto de aristas,
 - o Con cada **e**_k $\hat{=}$ {**v**_i, **v**_j}, con **v**_i, **v**_j $\hat{\in}$ **V**, **v**_i \neq **v**_j
- Los vértices se representan como puntos y las aristas como líneas entre vértices
 - Ejemplo:
 - o **G** = (**V**, **E**)
 - o **V** = {**a**, **b**, **c**, **d**}
 - o **E** = {{**a**, **b**}, {**b**, **c**}, {**a**, **c**}, {**a**, **d**}, {**d**, **b**}}
 - Proponer otro recorrido:



5.2.2 Computacional

Representación mediante matrices: La forma más fácil de guardar datos en nodos es mediante la utilización de un vector que indique los nodos, de manera que aristas entre los nodos se puedan ver como relaciones entre los índices.

Sintaxis:

```
Tipo_de_variable [ ] [ ]... [ ] Nombre_del_array = new Tipo_de_variable  
[dimensión1] [dimensión2]... [Dimensión];
```

Arreglos Unidimensionales: Es un arreglo que solo posee una dimensión, está formado por un conjunto de elementos del mismo tipo de datos que almacenan bajo un nombre y se diferencia por la posición de cada uno en el arreglo que inicia desde el 0. Estos pueden ser de 1 hasta n veces, donde n es un número de elementos del arreglo.

Sintaxis:

```
TipoDato nombre []=new TipoDato [Total de elementos];
```

5.3 Algoritmos de recorrido y búsqueda.

Existen algunas maneras útiles en las cuales se pueden ordenar sistemáticamente los nodos de un árbol. Los más importantes son: pre orden, post-orden y en-orden.

Estos tienen tres tipos de actividades comunes:

1. Visitar el nodo raíz
2. Recorrer el subárbol izquierdo
3. Recorrer el subárbol derecho

Estas tres actividades llevadas a cabo en distinto orden, crean los distintos recorridos del árbol.

Pre orden: (raíz, izquierdo, derecho). Para recorrer un árbol binario no vacío en pre orden, hay que realizar las siguientes operaciones recursivamente en cada nodo, comenzando con el nodo de raíz:

1. Visite la raíz
2. Atraviese el sub-árbol izquierdo
3. Atraviese el sub-árbol derecho

Inorden: (izquierdo, raíz, derecho). Para recorrer un árbol binario no vacío en inorden (simétrico), hay que realizar las siguientes operaciones recursivamente en cada nodo:

1. Atraviese el sub-árbol izquierdo
2. Visite la raíz
3. Atraviese el sub-árbol derecho

Postorden: (izquierdo, derecho, raíz). Para recorrer un árbol binario no vacío en postorden, hay que realizar las siguientes operaciones recursivamente en cada nodo:

1. Atraviese el sub-árbol izquierdo
2. Atraviese el sub-árbol derecho
3. Visite la raíz

Se llama recorrido de un árbol al proceso que permite acceder una vez a cada uno de los elementos de un árbol para examinar el conjunto completo. Primero se ven

el algoritmo para construir el árbol, para la expresión dada en sufijo, prefijo o posfijo y también se presentan algoritmos para reconocer si una expresión está correcta cuando está dada en prefijo o posfijo.

Los ordenamientos más importantes son llamados: prefijo, sufijo y posfijo.

Los algoritmos de recorrido de un árbol presentan tres tipos actividades:

1. Visitar el nodo raíz
2. Recorrer el subárbol izquierdo
3. Recorrer el subárbol derecho

Estas tres acciones llevadas a cabo en distinto orden proporcionan los distintos recorridos del árbol.


- Recorrido en PREFIJO:
 1. Visitar la raíz
 2. Recorrer el subárbol izquierdo en prefijo
 3. Recorrer el subárbol derecho en prefijo
- Recorrido SUFIJO:
 1. Recorrer el subárbol izquierdo en sufijo
 2. Visitar la raíz
 3. Recorrer el subárbol derecho en sufijo
- Recorrido en POSFIJO:
 1. Recorrer el subárbol izquierdo en postfijo
 2. Recorrer el subárbol derecho en postfijo
 3. Visitar la raíz

5.3.1 El camino más corto

También conocido como los problemas de los caminos cortos entre dos nodos.

Algoritmo de Floyd-Warshall

En informática, el algoritmo de Floyd-Warshall, descrito en 1959 por Bernard Roy, es un algoritmo de análisis sobre grafos para encontrar el camino mínimo entre grafos. El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución.

1. Permite calcular la distancia mínima entre 2 puntos de 1 grafo.
2. Cada nodo se representa por: 
3. Pasos:
 1. Asignar el valor 0 al nodo origen
 2. Mediante un proceso iterativo se le asignará a cada nodo X_i un valor n igual a la longitud del camino más corto que exista desde el nodo origen al nodo X_j .

Algoritmo Dijkstra:

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.

1.- Dado un V_0 , Dijkstra busca un conjunto D con

- Las menores distancias de V_0 al resto de vértices

2.- Al inicio, solo conocemos

- Las distancias de los adyacentes
- **D** es inicializada a
- Factor de peso para los adyacentes, Infinito ∞ para los no adyacentes

3.- D va ser mejorado sucesivamente

- Escogiendo el vértice V_k no elegido antes
- Que tenga la distancia más corta V_0, V_k
- Probamos si pasando por V_k
- Se puede obtener distancias más cortas de las que tenemos
- Para cada Vértice restante del grafo

5.3.2 A lo ancho

La búsqueda en anchura es otro procedimiento para visitar sistemáticamente todos los vértices de un grafo. Es adecuado especialmente para resolver problemas de optimización, en los que se deba elegir la mejor solución entre varias posibles.

Búsqueda en anchura: Es equivalente a recorrer un árbol por niveles. Dado un nodo v , se visitan primero todos los nodos adyacentes a v , luego todos los que están a distancia 2 (y no visitados), a distancia 3, y así sucesivamente hasta recorrer todos los nodos.

El recorrido en anchura, generalización del recorrido por niveles de un árbol. Explora sistemáticamente las aristas del grafo de forma que primero se visitan los vértices más “ceranos” al que estamos explorando.

Consta de tres elementos:

- Contador (n).
- Vector de naturales (R) para “marcar” los vértices ya visitados y almacenar el orden de recorrido.
- Cola (Q) para gestionar los vértices no visitados.

5.3.3 En profundidad

El algoritmo de recorrido en profundidad o DFS, explora sistemáticamente las ramas o aristas del grafo de manera que primero se visitan los nodos o vértices adyacentes a los visitados más recientemente. De esta forma se va "profundizando" en el grafo, es decir, alejándose progresivamente del nodo inicial.

Búsqueda en Profundidad

1. Se comienza en el vértice inicial (vértice con índice 1) que se marca como vértice activo.
2. Hasta que todos los vértices hayan sido visitados, en cada paso se avanza al vecino con el menor índice siempre que se pueda, pasando este a ser el vértice activo.
3. Cuando todos los vecinos al vértice activo hayan sido visitados, se retrocede al vértice X desde el que se alcanzó el vértice activo y se prosigue siendo ahora X el vértice activo.

Es equivalente a un recorrido en pre orden de un árbol. Se elige un nodo v de partida. Se marca como visitado y se recorren los nodos no visitados adyacentes a v, usando recursivamente la búsqueda primero en profundidad. El recorrido puede ser para grafos dirigidos o no dirigidos.

Unidad 6

6. Árboles y redes

6.1 Árboles.

Un árbol es una estructura no lineal de datos ampliamente usada que imita la forma de un árbol (un conjunto de nodos conectados). En ciencias de la computación, un árbol binario es una estructura de datos en la cual cada nodo puede tener un hijo izquierdo y un hijo derecho. No pueden tener más de dos hijos (de ahí el nombre "binario"). Si algún hijo tiene como referencia a null, es decir que no almacena ningún dato, entonces este es llamado un nodo externo. En el caso contrario el hijo es llamado un nodo interno. Un árbol binario es un grafo conexo, acíclico y no dirigido tal que el grado de cada vértice no es mayor a 3. De esta forma solo existe un camino entre un par de nodos. Usos comunes de los árboles binarios son los árboles binarios de búsqueda, los montículos binarios y Codificación de Huffman. Los árboles surgen en problemas teóricos como el tiempo óptimo para ordenar.

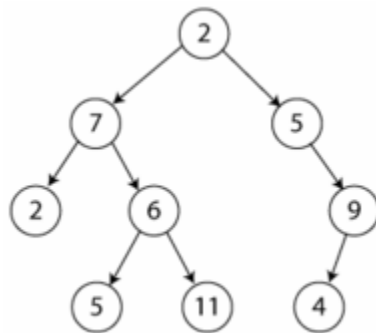
Los árboles forman una subclase de gráficas que más se utilizan. La ciencia de la computación hace uso de los árboles ampliamente, especialmente para organizar y relacionar datos en una base de datos.

Los árboles forman una de las subclases de gráficas que más se utilizan. La ciencia de la computación hace uso de los árboles ampliamente, especialmente para organizar y relacionar datos en una base de datos. Los árboles surgen en problemas teóricos como el tiempo óptimo para ordenar.

Formalmente se define un árbol de tipo T como una estructura homogénea que es la concatenación de un elemento de tipo T junto con un número finito de árboles disjuntos, llamados subárboles.

Una forma particular de árbol puede ser la estructura vacía. Un árbol es un grafo simple en el cual existe un único camino entre cada par de vértices.

Los árboles pueden ser contruidos con estructuras estáticas y dinámicas. Las estáticas son arreglos, registros y conjuntos, mientras que las dinámicas están representadas por listas. Sea $G = (V, A)$ un grafo no dirigido. G se denomina ARBOL, si es conexo y no contiene ciclos.



6.1.1 Componentes y propiedades

Altura: Es el máximo número de niveles de todos los nodos del árbol. Equivale al nivel más alto de los nodos más 1. También podemos hablar de altura de ramas, el máximo número de nodos que hay que recorrer para llegar de la raíz a una de las hojas.

Ancestros: los padres y los abuelos de un nodo hijo. **Descendientes:** Hijos de los hijos.

Grado del Árbol: Es el máximo grado de todos los nodos del árbol.

Grado: El número de hijos que tiene el elemento con más hijos dentro del árbol. En el árbol del ejemplo, el grado es tres, ya que tanto A como D tienen tres hijos, y no existen elementos con más de tres hijos. También es el número de descendientes directos de un determinado nodo.

Hermano: Dos nodos serán hermanos si son descendientes directos de un mismo nodo. En cuanto a la posición dentro del árbol.

Longitud de Camino: Es el número de arcos que deben ser recorridos para llegar desde la raíz al nodo X. Por definición la raíz tiene longitud de camino 1, y sus descendientes directos longitud de camino 2 y así sucesivamente.

Nivel: Es el número de arcos que deben ser recorridos para llegar a un determinado nodo. Por definición la raíz tiene nivel 1. Se define para cada elemento del árbol como la distancia a la raíz, medida en nodos

Nodo Hermano: Dos nodos serán hermanos si son descendientes directos de un mismo nodo.

Nodo Hijo: Cualquiera de lo nodo apuntado por uno de los nodo del árbol. Un nodo puede tener varios hijos. X es hijo de Y, sí y solo sí el nodo X es apuntado por Y. También se dice que X es descendiente directo de Y.

Nodo Hoja: Nodo que no tiene hijos. Se llama hoja o terminal a aquellos nodos que no tienen ramificaciones (hijos).

Nodo Interior: Es un nodo que no es raíz ni hoja.

Nodo Padre: X es padre de Y sí y solo sí el nodo X apunta a Y. También se dice que X es antecesor de Y.

Nodo Raíz: Es el único nodo del árbol que no tiene padre es decir no es hijo de ningún elemento. Este es el nodo que usaremos para referirnos al árbol.

Nodo: Son los Vértices o elementos del Árbol.

Orden: Es el número potencial de hijos que puede tener cada elemento de árbol. De este modo, diremos que un árbol en el que cada nodo puede apuntar a otros dos es de orden dos, si puede apuntar a tres será de orden tres, etc. Podríamos decir que nuestro árbol de ejemplo es de orden tres.

Peso: Es el número de nodos del árbol sin contar la raíz.

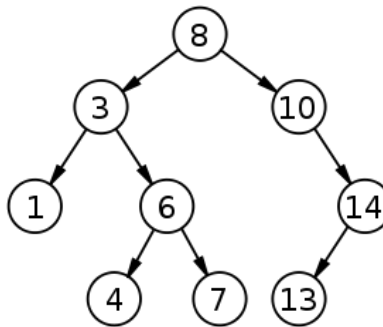
Rama: Es el camino desde el nodo raíz a una hoja.

Las siguientes son las características y propiedades más importantes de los árboles en general:

1. Todo árbol que no es vacío, tiene un único nodo raíz.
2. Un nodo X es descendiente directo de un nodo Y, si el nodo X es apuntado por el nodo Y. en este caso es común utilizar la expresión X es hijo de Y.
3. Un nodo X es antecesor directo de un nodo Y, si el nodo X apunta al nodo Y. en este caso es común utilizar la expresión X es padre de Y.
4. Se dice que todos los nodos que son descendientes directos (hijos) de un mismo nodo (padre), son hermanos.

5. Todo nodo que no tiene ramificaciones (hijos), se conoce con el nombre de terminal u hoja.
6. Todo nodo que no es raíz, ni terminal u hoja se conoce con el nombre de interior.
7. Grado es el número de descendientes directos de un determinado nodo. Grado del árbol es el máximo grado de todos los nodos del árbol, es decir, el grado más alto entre todos los nodos.
8. Nivel es el número de arcos que deben ser recorridos para llegar a un determinado nodo. Por definición la raíz tiene nivel 1.
9. Altura del árbol es el máximo número de niveles de todos los nodos del árbol.

A continuación se presenta un ejemplo para clarificar estos conceptos:



1.- 8 es la raíz del árbol.

2.- 3 es hijo de 8.

10 es hijo de 8.

1 es hijo de 3.

14 es hijo de 10.

13 es hijo de 14.

3.-8 es padre de 3.

3 es padre de 6.

6 es padre de 7.

10 es padre de 14.

14 es padre de 13.

4.- 3 y 10 son hermanos.

1 y 6 son hermanos.

4 y 7 son hermanos.

5.- 1, 4, 7, 13 son nodos terminales u hojas.

6.- 6, 14, 10, 3 son nodos interiores.

7.- El grado del nodo 8 es 2.

El grado del nodo 3 es 2.

El grado del nodo 6 es 2.

El grado del nodo 14 es 1.

El grado del nodo 1 es 0.

El grado del árbol es 3.

8.- El nivel del nodo 8 es 1.

El nivel del nodo 3 es 2.

El nivel del nodo 6 es 3.

El nivel del nodo 10 es 2.

El nivel del nodo 13 es 4.

6.1.2 Clasificación por altura y número de nodos

Altura = el nivel más grande

Raíz = que no tiene padre (inicial)

Padre = que tiene hijo(s)

Hoja = no tiene hijo(s), tiene padre

Conjunto de árboles = Bosque.

Árbol ordenado: tiene nivel, los hijos de izquierda a derecha.

n-árbol: cuando cada padre tiene a lo más n hijos

Árbol binario: cada padre tiene a lo más 2 hijos.

Altura de un nodo: Es la longitud del camino más largo desde el nodo hasta una hoja que sea descendiente de este nodo.

Altura de un árbol = altura del nodo raíz.

Para poder realizar búsquedas eficientes en árboles se manejan dos características: Los árboles pueden estar balanceados por altura o por peso.

Árbol balanceado por altura: en dónde todos los hijos o nodos hoja se intentan mantener a la misma distancia de la raíz.

Árbol balanceado por peso: en dónde los nodos más visitados o utilizados se mantienen a poca distancia de la raíz.

6.2. Árboles con peso

Dado un grafo conexo, un árbol recubierto mínimo de ese grafo es un subgrafo que tiene que ser un árbol y contener todos los vértices del grafo inicial. Cada arista tiene asignado un peso proporcional entre ellos, que es un número representativo

de algún objeto, distancia, etc... , y se usa para asignar un peso total al árbol recubierto mínimo computando la suma de todos los pesos de las aristas del árbol en cuestión. Un árbol recubridor mínimo o un árbol expandido mínimo es un árbol recubridor que pesa menos o igual que otros árboles recubridores.

Todo grafo tiene un bosque recubridor mínimo. En el caso de un empate, porque podría haber más de un árbol recubridor mínimo; en particular, si todos los pesos son iguales, todo árbol recubridor será mínimo. De todas formas, si cada arista tiene un peso distinto existirá sólo un árbol recubridor mínimo.

Árbol abarcador de menor peso. Vamos a considerar de nuevo el problema de la red de conducción, pero ahora añadiremos un ingrediente nuevo. El estudio previo de ingeniería nos informa de qué tramos es posible construir pero, además, disponemos de la información sobre el coste de cada uno de esos tramos. Queremos, claro, elegir una red que conecte todas las ciudades con el menor coste posible. La información de los costes se traduce en que cada arista lleva asociado un número. Lo que buscamos es un árbol abarcador del grafo, pero justo aquel (o aquellos) para el que la suma de los costes de las aristas elegidas sea mínimo.

Para modelar esta situación, necesitamos una generalización del concepto de grafo. Un grafo con pesos (o grafo ponderado) será un grafo G en el que, además, cada arista a tenga asociado lo que llamaremos su peso, $p(a)$, un número real no negativo. La matriz de vecindades de un grafo con pesos será simétrica, con ceros en la diagonal, y sus entradas serán los pesos de las aristas (o 0 si no hay tales aristas).

6.2.1 Recorrido de un árbol

- **Pre orden:** (raíz, izquierdo, derecho).

Para recorrer un árbol binario no vacío en pre orden, hay que realizar las siguientes operaciones recursivamente en cada nodo, comenzando con el nodo de raíz:

1. Visite la raíz
2. Atraviese el sub-árbol izquierdo
3. Atraviese el sub-árbol derecho

- **Inorden:** (izquierdo, raíz, derecho).

Para recorrer un árbol binario no vacío en inorden (simétrico), hay que realizar las siguientes operaciones recursivamente en cada nodo:

1. Atraviese el sub-árbol izquierdo
2. Visite la raíz
3. Atraviese el sub-árbol derecho

- **Postor den:** (izquierdo, derecho, raíz).

Para recorrer un árbol binario no vacío en postor den, hay que realizar las siguientes operaciones recursivamente en cada nodo:

1. Atraviese el sub-árbol izquierdo
2. Atraviese el sub-árbol derecho
3. Visite la raíz

En general, la diferencia entre pre orden, inorden y postor den es cuándo se recorre la raíz. En los tres, se recorre primero el sub-árbol izquierdo y luego el derecho.

- En pre orden, la raíz se recorre antes que los recorridos de los subárboles izquierdo y derecho
- En inorden, la raíz se recorre entre los recorridos de los árboles izquierdo y derecho, y

- En postorden, la raíz se recorre después de los recorridos por el subárbol izquierdo y el derecho

6.3 Redes.

Otro de los grandes problemas de optimización se da en la búsqueda del flujo máximo en las redes de distribución. Definición 33.- Llamaremos red a un dígrafo simple con dos vértices especiales, un minimal (la fuente) s y un maximal (el pozo) t , y que tiene asignada una capacidad no negativa c_{ij} a cada arco (v_i, v_j) del dígrafo. Un flujo f , es una asignación de un valor f_{ij} a cada arco (v_i, v_j) del dígrafo. Escribiremos $f^+(v)$ para denotar el flujo total saliente de un vértice v ; y $f^-(v)$ para denotar el flujo total entrante. Es decir $f^+(v_i) = \sum_k f_{ik}$, la suma de los flujos de los arcos salientes de v_i y $f^-(v_i) = \sum_k f_{ki}$, la suma de los flujos de los arcos entrantes. El flujo debe cumplir dos condiciones (en ocasiones se distingue denominándolo flujo factible o válido):

- El flujo en cada arco debe ser no negativo y no exceder la capacidad del arco, $0 \leq f_{ij} \leq c_{ij}$
- En cada vértice $v \neq s$ y $v \neq t$, distinto de la fuente y el pozo, debe cumplirse que $f^-(v) = f^+(v)$

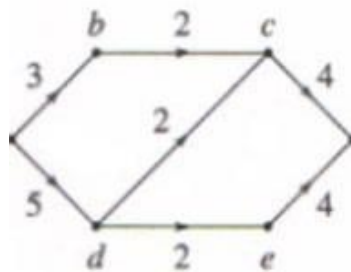
("Conservación del flujo" o "Ley de Kirchhoff")

Si bien el teorema del Flujo máximo y corte mínimo es el más conocido y admirado, no es útil para la obtención de un flujo máximo (en una red de n nodos tendríamos que chequear 2^{n-2} conjuntos de corte

Y eso sólo para conocer el valor del flujo máximo). El algoritmo se basa en el Teorema 40 anterior, se van buscando recorridos aumentadores del flujo hasta que no sea posible encontrar más. Para la búsqueda del recorrido aumentador, se usan dos procesos que conviene explicar, el etiquetado de un vértice y la exploración de un vértice. Explorar un vértice es buscar los vértices conectados con él mediante un arco, saliente o entrante, y son estos vértices encontrados en la exploración los que son etiquetados; sólo se etiquetan si no lo estaban ya y si tienen margen para aumentar el flujo. En el etiquetado, debe indicarse el vértice desde el que ha sido etiquetado y el margen posible de aumento; también suele indicarse si el arco para llegar a él se recorre "hacia adelante" o "hacia atrás". Inicialmente, se etiqueta solo s y se explora s , lo que produce el etiquetado de nuevos vértices. Para garantizar que la búsqueda avanza hacia t , se van explorando vértices ya etiquetados previamente —es decir para los que ya tenemos un recorrido aumentador— y no explorado, y se sigue así hasta llegar a t .

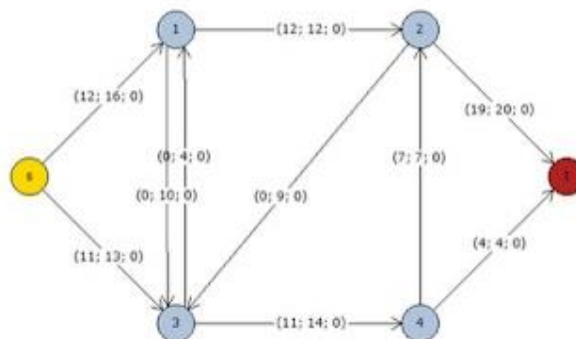
6.3.1 Teorema de flujo máximo

Teorema de flujo máximo: Siendo G una red de transporte, un flujo máximo es un flujo con valor máximo. En general, habrá varios flujos con el mismo valor máximo. La idea es sencilla: comenzar con cierto flujo inicial e incrementar de forma iterativa hasta que no pueda mejorarse más. El flujo resultante será el máximo. Para aumentar el valor de un flujo dado, debemos determinar un camino de la fuente al sumidero e incrementar el flujo a lo largo de ese camino.



6.3.2 Teorema de flujo mínimo

Teorema del flujo mínimo: En lo que respecta a las redes, un corte es un conjunto de corte en el cual quedando partes disjuntas del conjunto de vértices, V_1 y V_2 que, situados en la red, dejan la fuente en una de ellas y al sumidero en la otra. Se llama capacidad de un corte a la suma: $\sum_{(v,w) \in V_1 \times V_2} c(v,w)$. Sea F un flujo en G y sea (P, P) un corte en G . Entonces la capacidad de (p, p) es mayor o igual que el valor de F .



6.3.3 Pareos y redes de Petri

Redes de Petri: Una red de Petri es un grafo orientado con dos tipos de nodos: lugares (representados mediante circunferencias) y transiciones (representadas por segmentos rectos verticales). Los lugares y las transiciones se unen mediante arcos o flechas.

