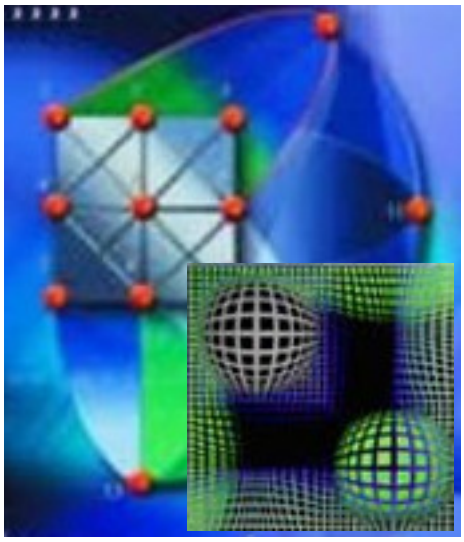
MATEMÁTICAS DISCRETAS
PARA LA CIENCIA DE LACOMPUTACIÓN



HUGO DAVID CALDERON VILCA

MATEMÁTICAS DISCRETAS
PARA LA CIENCIA COMPUTACIÓN

Autor: Hugo David Calderon Vilca

Editorial Pacífico
Jr. Cajamarca Nº 111
RUC: 10012176754
Abril 2008
Puno - Perú

INDICE

INTRODUCCIÓN

La matemática discreta es una rama de las matemáticas que trata de las estructuras finitas y numerables, lo discreto es lo finito por lo que presenta el aspecto de los números naturales, dándole fundamentos matemáticos para la ciencia de la computación en donde la información en los ordenadores se manipula en forma discreta (palabras formadas por ceros y uno).

En el capítulo I se presenta las matrices que son utilizados en la resolución de sistemas de ecuaciones lineales, además su utilidad mayor en este campo es en la presentación de árboles y grafos que se hace mediante matrices. En el Capítulo II presenta Álgebra de Boole que permite presentar funciones con dos estados. En el Capítulo III se presenta Mapas de Karnaugh que permiten simplificar las funciones algebraicas.

En el Capítulo IV Se tiene las técnicas de conteo las variaciones, permutaciones y combinaciones las cuales son parte de las Matemáticas Discretas que estudia las diversas formas de realizar agrupaciones con los elementos de un conjunto, formándolas y calculando su número.

En el Capítulo V y en el Capítulo VI se presenta la teoría de grafos, árboles y sus aplicaciones, para nadie es novedad observar en la vida cotidiana: carreteras, líneas telefónicas, líneas de televisión por cable, el transporte colectivo metro, circuitos eléctricos de nuestras casas, automóviles, etc, las cuales tienen su representación gráfica como sus recorridos y sus soluciones mediante grafos y árboles.

En el Capítulo VII se tiene autómatas de estado finito o máquinas de estado finito, es un modelo matemático de un sistema, herramienta muy útil para especificar aspectos relacionados con tiempo real, dominios reactivos o autónomos, computación reactiva, protocolos, circuitos y arquitecturas de software.

Finalmente en el Capítulo VIII se presenta el fundamento de Lenguajes formales y lenguajes naturales, en matemáticas, lógica, y ciencias de la computación, un lenguaje formal es un conjunto de palabras (cadenas de caracteres) de longitud finita formadas a partir de un alfabeto (conjunto de caracteres) finito.

CAPÍTULO I

MATRICES

Las matrices se utilizan en el cálculo numérico, en la resolución de sistemas de ecuaciones lineales, de las ecuaciones diferenciales y de las derivadas parciales. Además de su utilidad para el estudio de sistemas de ecuaciones lineales, las matrices aparecen de forma natural en geometría, estadística, economía, informática, física, etc...

La utilización de matrices (arrays) constituye actualmente una parte esencial dn los lenguajes de programación, ya que la mayoría de los datos se introducen en los ordenadores como tablas organizadas en filas y columnas : hojas de cálculo, bases de datos,...

Concepto de matriz

Una matriz es un conjunto de elementos de cualquier naturaleza aunque, en general, suelen ser números ordenados en filas y columnas.

Se llama **matriz** *de orden "m × n" a un conjunto rectangular de elementos* a_{ij} *dispuestos en m filas y en n columnas. El orden de una matriz también se denomina dimensión o tamaño, siendo m y n números naturales.*

Las matrices se denotan con letras mayúsculas: A, B, C, ... y los elementos de las mismas con letras minúsculas y subíndices que indican el lugar ocupado: a, b, c, ... Un elemento genérico que ocupe la fila i y la columna j se escribe a_{ij} . Si el elemento genérico aparece entre paréntesis también representa a toda la matriz : A = (a_{ij})

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & a_{ij} & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

$$Ej. \quad A_{2\times 3} = \begin{pmatrix} \frac{1}{6} & -3 & 5 \\ 7 & \sqrt{2} & 4 \end{pmatrix}$$

donde sus filas son : $\begin{pmatrix} \frac{1}{6} & -3 & 5 \end{pmatrix}$ y $\begin{pmatrix} 7 & \sqrt{2} & 4 \end{pmatrix}$

y sus columnas $\begin{pmatrix} \frac{1}{6} \\ 7 \end{pmatrix}$, $\begin{pmatrix} -3 \\ \sqrt{2} \end{pmatrix}$ y $\begin{pmatrix} 5 \\ 4 \end{pmatrix}$

Cuando nos referimos indistíntamente a filas o columnas hablamos de lineas.

El número total de elementos de una matriz A$m\times n$ es m·n
En matemáticas, tanto las **Listas** como las **Tablas** reciben el nombre genérico de matrices.

Una lista numérica es un conjunto de números dispuestos uno a continuación del otro.

Matrices iguales

Dos matrices A = $(a_{ij})m\times n$ y B = $(b_{ij})p\times q$ son iguales, sí y solo si, tienen en los mismo lugares elementos iguales, es decir : $m = p\ ,\ n = q\ \ ;\ a_{ij} = b_{ij}\ \ \forall i\ ,\forall j$

Algunos tipos de matrices

Hay algunas matrices que aparecen frecuentemente y que según su forma, sus elementos, ... reciben nombres diferentes :

Tipo de matriz	*Definición*
FILA	*Aquella matriz que tiene una sola fila, siendo su orden* 1×n
COLUMNA	*Aquella matriz que tiene una sola columna, siendo su orden* m×1
RECTANGULAR	*Aquella matriz que tiene distinto número de filas que de columnas, siendo su orden* $m\times n\ , m \neq n$
TRASPUESTA	*Dada una matriz* A*, se llama traspuesta de* A *a la matriz que se obtiene cambiando ordenadamente las filas por las columnas.* *Se representa por* A^t ó A^T
OPUESTA	*La matriz opuesta de una dada es la que resulta de sustituir cada elemento por su opuesto. La opuesta de* A *es* -A.
NULA	*Si todos sus elementos son cero. También se denomina matriz cero y se denota por* $0m\times n$
CUADRADA	*Aquella matriz que tiene igual número de filas que de columnas, m = n, diciendose que la matriz es de orden n.* Diagonal principal : *son los elementos* $a_{11}\ ,\ a_{22}\ ,\ ...,\ a_{nn}$ Diagonal secundaria : *son los elementos* a_{ij} *con* i+j = n+1 Traza de una matriz cuadrada : *es la suma de los elementos de la diagonal principal* tr A.
SIMÉTRICA	*Es una matriz cuadrada que es igual a su traspuesta.* $A = A^t\ , a_{ij} = a_{ji}$

ANTISIMÉTRICA	*Es una matriz cuadrada que es igual a la opuesta de su traspuesta.* $A = -A^t$, $a_{ij} = -a_{ji}$ *Necesariamente* $a_{ii} = 0$
DIAGONAL	*Es una matriz cuadrada que tiene todos sus elementos nulos excepto los de la diagonal principal*
ESCALAR	*Es una matriz cuadrada que tiene todos sus elementos nulos excepto los de la diagonal principal que son iguales*
IDENTIDAD	*Es una matriz cuadrada que tiene todos sus elementos nulos excepto los de la diagonal principal que son iguales a* 1. *Tambien se denomina matriz unidad.*
TRIANGULAR	*Es una matriz cuadrada que tiene todos los elementos por encima (por debajo) de la diagonal principal nulos.*
ORTOGONAL	*Una matriz ortogonal es necesariamente cuadrada e invertible :* $A^{-1} = A^T$ La inversa de una matriz ortogonal es una matriz ortogonal. El producto de dos matrices ortogonales es una matriz ortogonal. El determinante de una matriz ortogonal vale +1 ó -1.
NORMAL	*Una matriz es normal si conmuta con su traspuesta. Las matrices simétricas, antisimétricas u ortogonales son necesariamente normales.*
INVERSA	*Decimos que una matriz cuadrada* A *tiene inversa,* A-1, *si se verifica que :* $A \cdot A^{-1} = A^{-1} \cdot A = I$

Para establecer las reglas que rigen el cálculo con matrices se desarrolla un álgebra semejante al álgebra ordinaria, pero en lugar de operar con números lo hacemos con matrices.

OPERACIONES CON MATRICES

Suma de matrices

La suma de dos matrices $A = (a_{ij})_{m \times n}$ y $B = (b_{ij})_{p \times q}$ *de la misma dimensión (equidimensionales) :* $m = p$ *y* $n = q$ *es otra matriz* $C = A+B = (c_{ij})_{m \times n} = (a_{ij}+b_{ij})$

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \; ; \; B=\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$

$$A+B=\begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} \end{pmatrix}$$

p.ej.

$$A=\begin{pmatrix} 2 & -3 & 5 \\ 4 & 1 & -7 \end{pmatrix} ; B=\begin{pmatrix} 1 & 0 & 2 \\ -3 & 5 & 8 \end{pmatrix}$$

$$A+B=\begin{pmatrix} 3 & -3 & 7 \\ 1 & 6 & 1 \end{pmatrix}$$

Es una ley de composición interna con las siguientes **Propiedades :**

· Asociativa : A+(B+C) = (A+B)+C

· Conmutativa : A+B = B+A

· Elem. neutro : (matriz cero $0_{m\times n}$) , 0+A = A+0 = A

· Elem. simétrico :(matriz opuesta -A) , A + (-A) = (-A) + A=0

Al conjunto de las matrices de dimensión m×n cuyos elementos son números reales lo vamos a representar por $M_{m\times n}$ y como hemos visto, por cumplir las propiedades anteriores, (M, +) es un *grupo abeliano*.

¡¡ *La suma y diferencia de dos matrices NO está definida si sus dimensiones son distintas.* !!

Producto de un número real por una matriz

Para multiplicar un escalar por una matriz se multiplica el escalar por todos los elementos de la matriz, obteniéndose otra matriz del mismo orden.

$$A=(a_{ij})=\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \; ; \; \lambda\cdot A=\begin{pmatrix} \lambda a_{11} & \lambda a_{12} & \dots & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \dots & \lambda a_{2n} \\ \dots & \dots & \dots & \dots \\ \lambda a_{m1} & \lambda a_{m2} & \dots & \lambda a_{mn} \end{pmatrix}$$

p.ejm.

$$A=\begin{pmatrix} : & -2 & 3 \\ 0 & 1 & 8 \end{pmatrix} \; ; \; (-5)\cdot A=\begin{pmatrix} -5 & 10 & -15 \\ 0 & -5 & -40 \end{pmatrix}$$

Propiedades :

1) Asociativa: $\lambda(\mu A) = (\lambda\mu)A$
2) Distributiva I: $\lambda(A+B) = \lambda A + \mu B$
3) Distributiva II: $(\lambda+\mu)\cdot A = \lambda A + \mu A$
4) E. Neutro de escalares: $1\cdot A = A$

$\forall \lambda, \mu, 1 \in R \ ; \ \forall A \in M_{m\times n}$

por tanto la terna $\left[M_{m\times n}, +, \bullet R\right]$

constituye un ESPACIO VECTORIAL

Propiedades simplificativas

- A + C = B + C ➔ A = B.
- k A = k B ➔ A = B si k es distinto de 0.
- k A = h A ➔ h = k si A es distinto de 0.

Producto de matrices

Dadas dos matrices $A = (a_{ij})_{m\times n}$ y $B = (b_{ij})_{p\times q}$ *donde* $n = p$, *es decir, el número de columnas de la primera matriz* A *es igual al número de filas de la matriz* B , *se define el producto* A·B *de la siguiente forma :*

El elemento aque ocupa el lugar (i, j) en la matriz producto se obtiene sumando los productos de cada elemento de la fila i de la matriz A por el correspondiente de la columna j de la matriz B.

Dadas dos matrices A y B, su producto es otra matriz P cuyos elementos se obtienen multiplacando las filas de A por las columnas de B. De manera más formal, los elementos de P son de la forma:

$$p_{ij} = \sum a_{ik} \cdot b_{kj}$$

Es evidente que el número de columnas de A debe coincidir con el número de filas de B. Es más, si A tiene dimensión **m**x**n** y B dimensión **n**x**p**, la matriz P será de orden **m**x**p**. Es decir:

$$p_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Propiedades del producto de matrices

1. A·(B·C) = (A·B)·C
2. El producto de matrices en general no es conmutativo.

$$\begin{pmatrix} 1 & 1 \\ 5 & 3 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 6 \\ 12 & 22 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 5 & 3 \end{pmatrix} = \begin{pmatrix} 11 & 7 \\ 23 & 15 \end{pmatrix}$$

3. Si A es una matriz cuadrada de orden **n** se tiene $A \cdot I_n = I_n \cdot A = A$.
4. Dada una matriz cuadrada A de orden n, no siempre existe otra matriz B tal que $\mathbf{A \cdot B = B \cdot A = I_n}$. Si existe dicha matriz B, se dice que es la matriz inversa de A y se representa por $\mathbf{A^{-1}}$.
5. El producto de matrices es distributivo respecto de la suma de matrices, es decir: $\mathbf{A \cdot (B + C) = A \cdot B + A \cdot C}$

Consecuencias de las propiedades

1. Si $A \cdot B = 0$ no implica que A=0 ó B=0.

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

2. Si $A \cdot B = A \cdot C$ no implica que B = C.

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 \\ 3 & 2 \end{pmatrix}$$

MATRIZ INVERSA

Se llama matriz inversa de una matriz cuadrada An y la representamos por A^{-1} , a la matriz que verifica la siguiente propiedad : $A^{-1} \cdot A = A \cdot A^{-1} = I$

Decimos que una matriz cuadrada es *"regular"* si su determinante es distinto de cero, y es *"singular"* si su determinante es igual a cero.

$|A| \neq 0 \Rightarrow$ *Matriz* Regular
$|A| = 0 \Rightarrow$ *Matriz* Singular

Propiedades :

1) $A_n^{-1} \cdot A_n = A_n \cdot A_n^{-1} = I_n$

2) $\exists A_n^{-1} \Leftrightarrow |A_n| \neq 0$

3) $(A_n \cdot B_n)^{-1} = B_n^{-1} \cdot A_n^{-1}$

4) $\left(A_n^{-1}\right)^{-1} = A_n$

5) $(kA_n)^{-1} = \frac{1}{k} A_n^{-1}$

6) $\left(A^T\right)^{-1} = \left(A_n^{-1}\right)^T$

7) $A_n^{-1} = \frac{1}{|A_n|} \cdot \left[Adj(A_n)\right]^T$

- Sólo existe matriz inversa de una matriz cuadrada si ésta es *regular*.
- La matriz inversa de una matriz cuadrada, si existe, es única.
- Entre matrices NO existe la operación de división, la matriz inversa realiza funciones análogas.

Métodos para hallar la matriz inversa :

Aplicando la definición

Dada la matriz $A = \begin{pmatrix} 2 & -1 \\ 1 & 1 \end{pmatrix}$ buscamos una matriz que cumpla $A \cdot A^{-1} = I$, es decir

$$\begin{pmatrix} 2 & -1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Para ello planteamos el sistema de ecuaciones:

La matriz que se ha calculado realmente sería la inversa por la "derecha", pero es fácil comprobar que también cumple $A^{-1} \cdot A = I$, con lo cual es realmente la inversa de A.

Método de Gauss-Jordan para el cálculo de la matriz inversa

El método de Gauss-Jordan para calcular la matriz inversa de una dada se basa en una triangularización superior y luego otra inferior de la matriz a la cual se le quiere calcular la inversa.

Aplicando el método de Gauss-Jordan a la mtriz $A = \begin{pmatrix} 3 & 2 \\ 1 & 4 \end{pmatrix}$

- En primer lugar triangularizamos inferiormente:

$$\left(\begin{array}{cc|cc} 3 & 2 & 1 & 0 \\ 1 & 4 & 0 & 1 \end{array}\right) \xrightarrow{3F_2 - F_1} \left(\begin{array}{cc|cc} 3 & 2 & 1 & 0 \\ 0 & 10 & -1 & 3 \end{array}\right)$$

- Una vez que hemos triangularizado superiormente lo hacemos inferiormente:

$$\left(\begin{array}{cc|cc} 3 & 2 & 1 & 0 \\ 0 & 10 & -1 & 3 \end{array}\right) \xrightarrow{5F_1 - F_2} \left(\begin{array}{cc|cc} 15 & 0 & 6 & -3 \\ 0 & 10 & -1 & 3 \end{array}\right)$$

- Por último, habrá que convertir la matriz diagonal en la matriz identidad:

$$\left(\begin{array}{cc|cc} 15 & 0 & 6 & -3 \\ 0 & 10 & -1 & 3 \end{array}\right) \xrightarrow[F_2 = \frac{1}{10}F_2]{F_1 = \frac{1}{15}F_1} \left(\begin{array}{cc|cc} 1 & 0 & \frac{6}{15} & -\frac{3}{15} \\ 0 & 1 & -\frac{1}{10} & \frac{3}{10} \end{array}\right)$$

De donde, la matriz inversa de A es $A^{-1} = \begin{pmatrix} \frac{6}{15} & -\frac{3}{15} \\ -\frac{1}{10} & \frac{3}{10} \end{pmatrix}$

Queremos calcular la inversa de $A = \begin{pmatrix} 2 & -2 & 2 \\ 2 & 1 & 0 \\ 3 & -2 & 2 \end{pmatrix}$

1. Se escribe la matriz A junto a esta la matriz identidad,

$$\left(\begin{array}{ccc|ccc} 2 & -2 & 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 0 \\ 3 & -2 & 2 & 0 & 0 & 1 \end{array}\right)$$

2. Triangularizamos la matriz A de arriba a abajo y realizamos las mismas operaciones en la matriz de la derecha.

$$\left(\begin{array}{ccc|ccc} 2 & -2 & 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 0 \\ 3 & -2 & 2 & 0 & 0 & 1 \end{array}\right) \xrightarrow[2F_3-3F1]{F_2-F_1} \left(\begin{array}{ccc|ccc} 2 & -2 & 2 & 1 & 0 & 0 \\ 0 & 3 & -2 & -1 & 1 & 0 \\ 0 & 2 & -2 & -3 & 0 & 2 \end{array}\right)$$

$$\xrightarrow{3F_3-2F_2} \left(\begin{array}{ccc|ccc} 2 & -2 & 2 & 1 & 0 & 0 \\ 0 & 3 & -2 & -1 & 1 & 0 \\ 0 & 0 & -2 & -7 & -2 & 6 \end{array}\right)$$

Como podemos observar el rango de la matriz es máximo (en este caso 3), por tanto la matriz A es regular (tiene inversa), podemos calcular su inversa.

3. Triangularizamos la matriz de abajo a arriba, realizando las mismas operaciones en la matriz de la derecha.

$$\left(\begin{array}{ccc|ccc} 2 & -2 & 2 & 1 & 0 & 0 \\ 0 & 3 & -2 & -1 & 1 & 0 \\ 0 & 0 & -2 & -7 & -2 & 6 \end{array}\right) \xrightarrow[F_1+F_3]{F_2-F_3} \left(\begin{array}{ccc|ccc} 2 & -2 & 0 & -6 & -2 & 6 \\ 0 & 3 & 0 & 6 & 3 & -6 \\ 0 & 0 & -2 & -7 & -2 & 6 \end{array}\right)$$

$$\xrightarrow{3F_1+2F_2} \left(\begin{array}{ccc|ccc} 2 & 0 & 0 & -6 & 0 & 6 \\ 0 & 3 & 0 & 6 & 3 & -6 \\ 0 & 0 & -2 & -7 & -2 & 6 \end{array}\right)$$

4. Por último se divide cada fila por el elemento diagonal correspondiente.

$$\left(\begin{array}{ccc|ccc} 2 & 0 & 0 & -6 & 0 & 6 \\ 0 & 3 & 0 & 6 & 3 & -6 \\ 0 & 0 & -2 & -7 & -2 & 6 \end{array}\right) \xrightarrow[F_3=-\frac{1}{2}F_3]{\substack{F_1=\frac{1}{6}F_1 \\ F_2=\frac{1}{3}F_2}} \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 2 & 1 & -2 \\ 0 & 0 & 1 & \frac{7}{2} & 1 & -3 \end{array}\right)$$

$$\Rightarrow \quad A^{-1} = \begin{pmatrix} -1 & 0 & 1 \\ 2 & 1 & -2 \\ \frac{7}{2} & 1 & -3 \end{pmatrix}$$

Para aplicar el método se necesita una matriz cuadrada de rango máximo. Sabemos que no siempre una matriz tiene inversa, por lo cual comprobaremos que la matriz tenga rango máximo al aplicar el método de Gauss para realizar la triangularización superior. Si al aplicar el método de Gauss (triangularización inferior) se obtiene una línea de ceros, la matriz no tiene inversa.

Aplicando el método de Gauss-Jordan a la matriz $A = \begin{pmatrix} 1 & 2 \\ -2 & -4 \end{pmatrix}$ se tiene:

$$\left(\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ -2 & -4 & 0 & 1 \end{array}\right) \xrightarrow{F_1+2F_2} \left(\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \end{array}\right)$$

Como hay una fila completa de ceros, la matriz A no tiene rango máximo, en este caso 2, por tanto no tiene inversa pues es una matriz singular

RANGO DE UNA MATRIZ

Se llama menor de orden p de una matriz al determinante que resulta de eliminar ciertas filas y columnas hasta quedar una matriz cuadrada de orden p. Es decir, al determinante de cualquier submatriz cuadrada de A (submatriz obtenida suprimiendo alguna fila o columna de la matriz A).

En una matriz cualquiera $A_{m\times n}$ puede haber varios menores de un cierto orden p dado.

Definición

El RANGO (o característica) de una matriz es el orden del mayor de los menores distintos de cero.

El rango o característica de una matriz A se representa por rg(A).

Cálculo del rango usando determinantes

Si a un menor M de orden h de la matriz A se le añade la fila p y la columna q de A (que antes no estaban en el menor), obtenemos un menor N de orden h+1 que se dice obtenido de M orlando este menor con la fila p y la columna q.

$M = \begin{vmatrix} 1 & -1 \\ 1 & 0 \end{vmatrix}$ es un menor de orden 2 de la matriz $A = \begin{pmatrix} 1 & -1 & 3 & 2 \\ 1 & 0 & 5 & -2 \\ 4 & 1 & -3 & 1 \\ 0 & 1 & 2 & 3 \end{pmatrix}$

$\begin{vmatrix} 1 & -1 & 3 \\ 1 & 0 & 3 \\ 4 & 1 & -3 \end{vmatrix}$ y $\begin{vmatrix} 1 & -1 & 2 \\ 1 & 0 & -2 \\ 0 & 1 & 3 \end{vmatrix}$ son menores de orden 3 que se han obtenido orlando M

El método para el cálculo del rango es un proceso iterado que sigue los siguientes pasos:

Antes de comenzar el método se busca un elemento no nulo, ya que si todos los elementos son 0, el rango será 0. El elemento encontrado será el menor de orden k=1 de partida.

1. Se orla el menor de orden k hasta encontrar un menor de orden k+1 no nulo. Cuando se encuentra un menor de orden k+1 no nulo se aplica a éste el método.

2. Si todos los menores orlados obtenidos añadiéndole al menor de partida los elementos de una línea i_0 son nulos, podemos eliminar dicha línea porque es combinación de las que componen el menor de orden k.
3. Si todos los menores de orden k+1 son nulos el rango es k. (Si aplicamos bien el método en realidad, al llegar a este punto, la matriz tiene orden k).

$$A = \begin{pmatrix} 2 & -3 & 1 & 0 & 1 & 0 \\ -1 & 0 & 1 & 2 & 0 & 0 \\ 1 & -6 & 5 & 6 & 2 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$1 \neq 0 \Rightarrow r(A) \geq 1$.

$$\begin{vmatrix} 2 & 0 \\ 1 & 1 \end{vmatrix} = 2 \neq 0 \Rightarrow r(A) \geq 2$$

$$\begin{vmatrix} 2 & 0 & 0 \\ 6 & 2 & 0 \\ 0 & 1 & 1 \end{vmatrix} = 4 \neq 0 \Rightarrow r(A) \geq 3$$

$$\begin{vmatrix} 1 & 0 & 1 & 0 \\ 1 & 2 & 0 & 0 \\ 5 & 6 & 2 & 0 \\ 1 & 0 & 1 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 1 \\ 1 & 2 & 0 \\ 5 & 6 & 2 \end{vmatrix} = 4 + 6 - 10 = 0$$

$$\begin{vmatrix} -3 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ -6 & 6 & 2 & 0 \\ 0 & 0 & 1 & 1 \end{vmatrix} = \begin{vmatrix} -3 & 0 & 1 \\ 0 & 2 & 0 \\ -6 & 6 & 2 \end{vmatrix} = -12 + 12 = 0$$

$$\begin{vmatrix} 2 & 0 & 1 & 0 \\ -1 & 2 & 0 & 0 \\ 1 & 6 & 2 & 0 \\ 0 & 0 & 1 & 1 \end{vmatrix} = \begin{vmatrix} 2 & 0 & 1 \\ -1 & 2 & 0 \\ 1 & 6 & 2 \end{vmatrix} = 8 - 6 - 2 = 0$$

Por tanto rg(A)=3

Cálculo del rango de una matriz por el método de Gauss

Transformaciones elementales

Son las transformaciones que podemos realizarle a una matriz sin que su rango varíe. Es fácil comprobar que estas transformaciones no varían el rango usando las propiedades de los determinantes

1. Si se permutan 2 filas ó 2 columnas el rango no varía.
2. Si se multiplica o divide una línea por un número no nulo el rango no cambia.
3. Si a una línea de una matriz se le suma o resta otra paralela multiplicada por un número no nulo el rango no varía.
4. Se pueden suprimir las filas o columnas que sean nulas, las filas o columnas que sean que sean proporcionales a otras, sin que el rango de la matriz varíe.

Método de Gauss

El método de Gauss consiste en aplicar transformaciones elementales a una matriz con objeto de conseguir que los elementos que están por debajo de la diagonal principal se anulen ($a_{ij} = 0$, $i>j$). Para conseguir "triangularizar" la matriz debemos dejar en la diagonal principal elementos no nulos, salvo que la fila sea nula.

Una vez aplicado este proceso de triangularización, el rango de la matriz es el número de filas no nulas de la matriz obtenida. Esto es fácil probarlo usando las propiedades de los determinantes.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -1 & 7 \\ 0 & 0 & 4 \end{pmatrix} \Rightarrow rg(A) = 3$$

$$A = \begin{pmatrix} 1 & 0 & -3 \\ 2 & 3 & -6 \\ 4 & 6 & -11 \end{pmatrix} \xrightarrow[F_3-4F_1]{F_2-2F_1} \begin{pmatrix} 1 & 0 & -3 \\ 0 & 3 & 0 \\ 0 & 6 & 1 \end{pmatrix} \xrightarrow{F_3-2F_2} \begin{pmatrix} 1 & 0 & -3 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow rg(A) = 3$$

$$A = \begin{pmatrix} 1 & 2 & -1 & 3 \\ -4 & -8 & 4 & -12 \\ -3 & -6 & 3 & -9 \end{pmatrix} \xrightarrow[F_3+3F_1]{F_2+4F_1} \begin{pmatrix} 1 & 2 & -1 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 2 & -1 & 3 \end{pmatrix} \Rightarrow rg(A) = 1$$

$$A=\begin{pmatrix}1&2&3&4\\5&6&7&8\\9&10&11&12\\13&14&15&16\end{pmatrix}\xrightarrow[F_4-F_3]{\substack{F_2-F_1\\F_3-F_2}}\begin{pmatrix}1&2&3&4\\4&4&4&4\\4&4&4&4\\4&4&4&4\end{pmatrix}\longrightarrow\begin{pmatrix}1&2&3&4\\4&4&4&4\end{pmatrix}\Rightarrow rg(A)=2$$

$$A=\begin{pmatrix}5&7&8&9\\0&0&0&0\\0&2&1&-1\\0&0&6&0\end{pmatrix}\rightarrow\begin{pmatrix}5&7&8&9\\0&2&1&-1\\0&0&6&0\end{pmatrix}\Rightarrow rg(A)=3$$

DETERMINANTES

Dada una matriz cuadrada

$$A=\begin{pmatrix}a_{11}&a_{12}&\cdots&a_{1j}&\cdots&a_{1n}\\a_{21}&a_{22}&\cdots&a_{2j}&\cdots&a_{2n}\\\cdots&\cdots&\cdots&\cdots&\cdots&\cdots\\a_{i1}&a_{i2}&\cdots&a_{ij}&\cdots&a_{in}\\\cdots&\cdots&\cdots&\cdots&\cdots&\cdots\\a_{n1}&a_{n2}&\cdots&a_{nj}&\cdots&a_{nn}\end{pmatrix}$$

se llama determinante de A, y se representa por |A| ó det(A), al número:

$$|A|=\sum_{\sigma} i(\sigma)a_{1\sigma(1)}a_{2\sigma(2)}\cdots a_{n\sigma(n)}$$, con $\sigma\in S_n$

También se suele escribir:

$$\begin{vmatrix}a_{11}&a_{12}&\cdots&a_{1j}&\cdots&a_{1n}\\a_{21}&a_{22}&\cdots&a_{2j}&\cdots&a_{2n}\\\cdots&\cdots&\cdots&\cdots&\cdots&\cdots\\a_{i1}&a_{i2}&\cdots&a_{ij}&\cdots&a_{in}\\\cdots&\cdots&\cdots&\cdots&\cdots&\cdots\\a_{n1}&a_{n2}&\cdots&a_{nj}&\cdots&a_{nn}\end{vmatrix}$$

Cálculo de determinantes de órdenes 1, 2 y 3

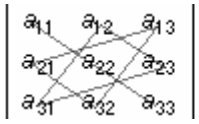
Es fácil comprobar que aplicando la definición se tiene: $A=(a_{11})\ \Rightarrow\ \det(A)=a_{11}$

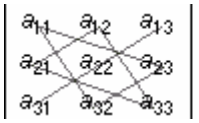
$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \Rightarrow \det(A) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \Rightarrow \det(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} =$$

$$= (a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32}) - (a_{13}a_{22}a_{31} + a_{12}a_{21}a_{33} + a_{11}a_{23}a_{32})$$

En este último caso, para acordarnos de todos los productos posibles y sus correspondientes signos se suele usar la Regla de Sarrus, que consiste en un esquema gráfico para los productos positivos y otro para los negativos:

 (Para los tres productos positivos).

 (Para los tres productos negativos).

Cálculo de un determinante por los adjuntos de una línea

Sea A una matriz cuadrada y a_{ij} uno cualquiera de sus elementos. Si se suprime la fila i y la columna j de la matriz A se obtiene una submatriz $\mathbf{M_{ij}}$ que recibe el nombre de **matriz**

complementaria del elemento a_{ij}.

Dada la matriz

$$A = \left(\begin{array}{c|ccccc} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ \hline a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nj} & \cdots & a_{nn} \end{array}\right)$$

la matriz complementaria del elemento a_{11} es la matriz que resulta de suprimir en la matriz A la fila 1 y la columna 1; es decir:

$$M_{11} = \begin{pmatrix} a_{22} & \cdots & a_{2n} \\ \cdots & & \cdots \\ a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

Llamamos menor complementario del elemento a_{ij} al determinante de la matriz complementaria del elemento a_{ij} , y se representa por a_{ij}

Se llama **adjunto de a_{ij}** , y se representa por por $\mathbf{A_{ij}}$, al número $(-1)^{i+j}a_{ij}$.

El determinante de una matriz cuadrada es igual a la suma de los elementos de una fila o columna cualquiera, multiplicados por sus adjuntos.

Por ejemplo, si desarrollamos un determinante de orden *n* por los adjuntos de la 1ª fila se tiene:

$$|A| = \sum_{i=1}^{n} a_{1i}A_{1j}$$

La demostración es muy fácil, basta con aplicar la definición de determinante a ambos lados de la igualdad.

Nota

Esta regla rebaja el orden del determinante que se pretende calcular en una unidad. Para evitar el cálculo de muchos determinantes conviene elegir líneas con muchos ceros

Desarrollando por la primera columna:

$$\begin{vmatrix} x & 1 & 0 & 0 \\ 0 & x & 1 & 0 \\ 0 & 0 & x & 1 \\ 1 & 0 & 0 & x \end{vmatrix} = x\begin{vmatrix} x & 1 & 0 \\ 0 & x & 1 \\ 0 & 0 & x \end{vmatrix} - 1\begin{vmatrix} 1 & 0 & 0 \\ x & 1 & 0 \\ 0 & x & 1 \end{vmatrix} = x^4 - 1$$

Cálculo de determinantes por el método de Gauss

Se conoce cómo método de Gauss a un método para facilitar el cálculo de determinantes usando las propiedades de éstos. Dicho método consiste en hallar un determinante equivalente (con el mismo valor) al que se pretende calcular, pero triangular. De esta forma el problema se reduce a calcular un determinante de una matriz triangular, cosa que es bastante fácil usando las propiedades de los determinantes.

Para conseguir triangularizar el determinante se pueden aplicar las siguientes operaciones:

- Permutar 2 filas ó 2 columnas.
- Multiplicar o dividir una línea por un número no nulo.
- Sumarle o restarle a una línea otra paralela multiplicada por un número no nulo.

$$\begin{vmatrix} 1 & 0 & 3 \\ 2 & 1 & 4 \\ 1 & 0 & -1 \end{vmatrix} \begin{matrix} F_2-2F_1 \\ = \\ F_3-F_1 \end{matrix} \begin{vmatrix} 1 & 0 & 3 \\ 0 & 1 & -2 \\ 0 & 0 & 4 \end{vmatrix} = -4$$

$$\begin{vmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & -1 \\ 0 & -1 & -2 & 1 \\ 6 & 1 & 1 & 0 \end{vmatrix} \begin{matrix} \\ = \\ F_4-6F_1 \end{matrix} \begin{vmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & -1 \\ 0 & -1 & -2 & 1 \\ 0 & 1 & 1 & -12 \end{vmatrix} \begin{matrix} \\ F_3+F_2 \\ = \\ F_4-F_2 \end{matrix} \begin{vmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & -1 \\ 0 & 0 & -1 & 2 \\ 0 & 0 & 0 & -13 \end{vmatrix} = 13$$

MATRICES BOOLEANAS

Una *matriz booleana* es una matriz cuyos elementos son ceros o unos.

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

Se emplean para representar estructuras discretas (representación de relaciones en programas informáticos, modelos de redes de comunicación y sistemas de transporte).

Operaciones booleanas

- *Matriz intersección:* A ^ B = (aij ^ bij)ij

  a ^ b = 1 si a = b = 1 y 0 en otro caso; a, b Є {0, 1}

  Ejemplo:

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

- Matriz *unión:*

  A v B = (aij _ bij)ij

  a v b = 1 si a = 1 o b = 1; y 0 en otro caso; a, b Є {0, 1}

  Ejemplo:

  $$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \text{v} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

- La matriz *complementaria* de A es la matriz cuyos elementos son unos donde A tiene ceros, y ceros donde A tiene unos.

  Ejemplo: la matriz complementaria de

Dado la matriz

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

Su complemento es

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

- La *matriz* diferencia *simétrica* de A y B es la matriz booleana cuyo elemento (i, j) es uno cuando el primer elemento es 1 ó 0 y el segundo elemento es el completo; y el elemento (i, j) es cero cuando ambos elementos son ceros o unos.

  Ejemplo: la diferencia simétrica de

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \text{v} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

GUIA DE LABORATORIO TEMA MATRICES

1. Implementar el algoritmo y programa en C++, para leer dos matrices A y B y obtener otra matriz C que es la suma de A y B.

2. Implementar el algoritmo y programa en C++ para obtener los valores de senos y cosenos desde 1 hasta 360 grados sexagesimales esto un arras unidimensionales, luego mostrar aquellos valores.

```
//ALGORITMO QUE ALMACENA Y MUESTRA LOS SENOS Y  COSENOS EN ARRAYS
UNIDIMENSIONALES

Inicio
Variable senos[360], cosenos[360], i
Para i=0 hasta 360 incremento de 1 hacer
    Senos[i] = sin(i)
    Cosenos[i] = cos(i)

Para i=0 hasta 360 incremento de 1 hacer
    Imprimir cosenos[i]

Para i=0 hasta 360 incremento de 1 hacer
    Imprimir senos[i]
fin
```

```
//PROGRAMA QUE ALMACENA LOS SENOS Y COSENOS UN ARRAY
UNIDIMENSIONAL
      #include <stdio.h>
      #include <math.h>
      main()
      {
      float   senos[360]; /* Almacenamos senos */
```

```
    float   cosenos[360];
    int     i;

    /* Inicializamos las matrices */
    for (i=0;i<360;i++)
        {           senos[i]   = sin (3.14159*i/180);
                cosenos[i] = cos (3.14159*i/180);
        }
   printf ("\nEl coseno de 30 es : %f",cosenos[30]);
    printf ("\nEl seno de 30 es   : %f",senos[30]);

}
```

3. Implementar el algoritmo y programa en C++ para leer dos matrices A y B de distintas dimensiones, además que calcules la matriz C el resultante de la multiplicación de A y B.

ALGORITMO PARA MULTIPLICACIÓN DE MATRICES

1. inicio
2. variable A[20][20], B[][], C[][]
3. leer m,n de A
4. para i=1 hasta m de 1 en 1
   para j=1 hasta n de 1 en 1
   leer A[i][j]
5. leer p,r de B
6. para i=1 hasta m de 1 en 1
   para j=1 hasta n de 1 en 1
   leer B[i][j]
7. si n = r entonces
   para i=1 hasta m de 1 en 1
   para j=1 hasta n de 1 en 1
   c[i][]=0
   para k=1 hasta n de 1 en 1
   C[i][j]=C[i][j] +A[i][k]*B[k][j]

4. Implementar el algoritmo y programa en C++, para obtener la matriz T transpuesta a partir de una matriz A.

ALGORITMO PARA HALLAR LA MATRIZ TRANSPUESTA

1. inicio
2. variable A[20][20], B[][]
3. leer m,n de A
4. para i=1 hasta m de 1 en 1
   para j=1 hasta n de 1 en 1
   leer A[i][j]
5. para i=1 hasta m de 1 en 1
   para j=1 hasta n de 1 en 1
   B[j][i]=A[i][j]

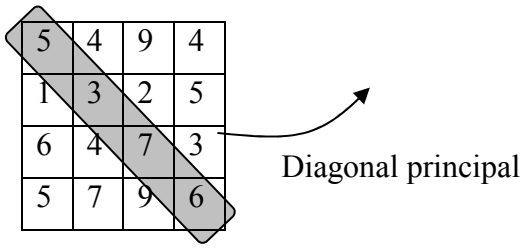
5. Desarrollar un programa que calcule el mayor elemento de una matriz.

```
 #include <conio.h>
#include <iostream.h>
#include <dos.h>
void main()
```

```
{
        clrscr();
        int i,j,m,n,mayor;
        int ind_i,ind_j;
        int A[100][100];
        cout<<"Ingrese tama¤o de filas de la matriz A: ";
        cin>>m;
        cout<<"Ingrese tama¤o de columnas de la matriz A: ";
        cin>>n;
        cout<<"Ingrese los elementos de A:\n";
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {       cout<<"A["<<i<<"]["<<j<<"]: ";
                        cin>>A[i][j];
                }
        }
        mayor=A[0][0];
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        if(A[i][j]>mayor)
                        {       mayor=A[i][j];
                                ind_i=i;
                                ind_j=j;
                        }
                }
        }
        clrscr();
        cout<<"LA MATRIZ ES:\n\n";

        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {  cout<<A[i][j]<<"\t";
                }
                cout<<endl;
        }
        cout<<"El mayor elemento es: "<<mayor<<"\n";
        cout<<"Y se encuentra en la fila "<<ind_i;
        cout<<" columna "<<ind_j;
        getch();
}
```

6. Desarrollar un programa que muestre el elementos menor y menor de de cada fila.
7. Calcular la suma de los elementos de la diagonal principal de una matriz.



8. Ordenar en forma ascendente los elementos de una matriz

9. Llenar una matriz de la siguiente manera, la matriz debe de ser de m filas por n columnas.

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8

10. SISTEMA DE ECUACIONES POR GOUS JORDAN

```
#include<conio.h>
#include<stdio.h>
void leermatriz(int n),escribir(int n),gousjordan(int n);
intercambio(int n,int l);
float A[30][30];
main()
{ int N;
 clrscr();
 printf("Ingrese n\n");
 scanf("%d",&N);
 leermatriz(N);
 escribir(N);
 gousjordan(N);
 printf("\n El resultado es ");
 escribir(N);
 getch();
}
void leermatriz(int n)
{ int i,j;
 for(i=1;i<=n;i++)
  {
    for(j=1;j<=n+1;j++)
     {printf("Dato[%d][%d]: ",i,j);
      scanf("%f",&A[i][j]);
     }
  }
}

void escribir(int n)
{ int i,j;
 printf("\n");
 for(i=1;i<=n;i++)
  { printf("\n");
    for(j=1;j<=n+1;j++)
     {
       printf("%.2f    ",A[i][j]);
     }
  }
  getch();
```

```
}
intercambio(int n,int l)
{       int j,k;
        float temp;
        k=l;
        while (k<=n && A[k][l]==0)
         {      k++;
                if(k<=n)
                        {   for(j=1;j<=n+1;j++)
                            {   temp=A[l][j];
                                A[l][j]=A[k][j];
                                A[k][j]=temp;
                                }
                                escribir(n);
                        }
        }

}
void gousjordan(int n)
{       int i,j,flag,k;
        float pivote,piv;
        for (i=1; i<=n; i++)
                {       if(A[i][i]==0)
                        intercambio(n,i);
                        piv=A[i][i];
                        for(j=i;j<=n+1;j++)
                          {
                                A[i][j]=A[i][j]/piv;
                          }
                        escribir(n);
                        for(k=1;k<=n; k++)
                                if(k!=i)
                                        {       pivote=A[k][i];
                                                for(j=i;j<=n+1;j++)
                                                    {A[k][j]=A[k][j]-pivote*A[i][j];
                                                    }
                                                 escribir(n);
                                        }
                escribir(n);
                }
}
```

CAPÍTULO II

ÁLGEBRA DE BOOLE

LA ALGEBRA DE BOOLE COMO RETÍCULA

El álgebra de Boole es un retículo (A, $\cdot$, +), donde el conjunto A esta formado por dos elementos A={0, 1}, como retículo presenta las siguientes propiedades:

1. Ley de Idempotente:

$$a \cdot a = a$$
$$a + a = a$$

2. Ley de Asociatividad:

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$
$$a + (b + c) = (a + b) + c$$

3. Ley de Conmutatividad:

$$a \cdot b = b \cdot a$$
$$a + b = b + a$$

4. Ley de Cancelativo

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

ALGEBRA DE BOOL COMO GRUPO ABELIANO RESPECTO A (+)

El conjunto A={0,1} es un Grupo abeliano respecto a (+):

1. (+) es una operación interna en A:

$$(a + b) \in A;\ \forall a, b \in A$$

2. Es asociativa:

$$a + (b + c) = (a + b) + c;\ \forall a, b, c \in A$$

3. Tiene elemento neutro

$$\exists 0 \in A;\ \forall a \in A : a + 0 = 0 + a = a$$

4. Tiene elemento simétrico:

$$\forall a \in A;\ \exists \bar{a} \in A\ /\ a + \bar{a} = \bar{a} + a = 1$$

5. es conmutativa:

$$a + b = b + a;\ \forall a, b \in A$$

ALGEBRA DE BOOL COMO GRUPO ABELIANO RESPECTO A (·)

El conjunto A={0,1} es un Grupo abeliano respecto a (·):

6. (·) es una operación interna en A:

$$(a \cdot b) \in A;\ \forall a, b \in A$$

7. Es asociativa:

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c;\ \forall a, b, c \in A$$

8. Tiene elemento neutro

$$\exists 1 \in A;\ \forall a \in A:\ a \cdot 1 = 1 \cdot a = a$$

9. Tiene elemento simétrico:

$$\forall a \in A;\ \exists \bar{a} \in A\ /\ a \cdot \bar{a} = \bar{a} \cdot a = 0$$

10. es conmutativa:

$$a \cdot b = b \cdot a;\ \forall a, b \in A$$

Distributivo

El conjunto A={0,1} es un Grupo abeliano respecto a (+) y (·) y es distributiva:

11. La operación (+) es distributiva respecto a (·):

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$
$$(a + b) \cdot c = (a \cdot c) + (b \cdot c)$$

12. La operación (·) es distributiva respecto a (+):

$$a + (b \cdot c) = (a + b) \cdot (a + c)$$
$$(a \cdot b) + c = (a + c) \cdot (b + c)$$

OPERACIONES CON ÁLGEBRA DE BOOL

Hemos definido el conjunto A = {0,1} como el conjunto universal sobre el que se aplica el álgebra de Boole, sobre estos elementos se definen varias operaciones, veamos las mas fundamentales:

Operación suma

La operación suma (+) asigna a cada par de valores **a**, **b** de **A** un valor **c** de **A**:

$$a + b = c$$

Si uno de los valores de **a** o **b** es 1, el resultado será 1, es necesario que los dos sumandos sean 0, para que el resultado sea 0.

A	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

Operación producto

La operación producto ($\cdot$) asigna a cada par de valores **a**, **b** de **A** un valor **c** de **A**:

$$a \cdot b = c$$

solo si los dos valores **a** y **b** son 1, el resultado será 1, si uno solo de ellos es 0 el resultado será 0.

A	b	a $\cdot$ b
0	0	0
0	1	0
1	0	0
1	1	1

Operación negación

La operación negación presenta el opuesto del valor de **a**:

$$\bar{a} = c$$

A	$\bar{a}$
0	1
1	0

Leyes fundamentales

El resultado de aplicar cualquiera de las tres operaciones definidas a variables del sistema booleano resulta en otra variable del sistema, y este resultado es único.

1. Ley de idempotencia:

$$a \cdot a = a$$
$$a + a = a$$

2. Ley de involución:

$$\overline{\bar{a}} = a$$

3. Ley conmutativa:

$$a \cdot b = b \cdot a$$
$$a + b = b + a$$

4. Ley asociativa:

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$
$$a + (b + c) = (a + b) + c$$

5. Ley distributiva:

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$
$$(a + b) \cdot c = (a \cdot c) + (b \cdot c)$$
$$a + (b \cdot c) = (a + b) \cdot (a + c)$$
$$(a \cdot b) + c = (a + c) \cdot (b + c)$$

6. Ley de cancelación:

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

7. Ley de De Morgan:

$$\overline{(a + b)} = \bar{a} \cdot \bar{b}$$
$$\overline{(a \cdot b)} = \bar{a} + \bar{b}$$

Principio de dualidad

El concepto de dualidad permite formalizar este hecho: a toda relación o ley lógica le corresponderá su dual, formada mediante el intercambio de los operadores unión con los de intersección, y de los 1 con los 0.

	Adición	**Producto**
1	$a + \bar{a} = 1$	$a \cdot \bar{a} = 0$

2	$a + 0 = a$	$a \cdot 0 = 0$
3	$a + 1 = 1$	$a \cdot 1 = a$
4	$a + a = a$	$a \cdot a = a$
5	$a + b = b + a$	$a \cdot b = b \cdot a$
6	$a + (b + c) = (a + b) + c$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
7	$a + (b \cdot c) = (a + b) \cdot (a + c)$	$a \cdot (b + c) = a \cdot b + a \cdot c$
8	$a + a \cdot b = a$	$a \cdot (a + b) = a$
9	$\overline{(a + b)} = \bar{a} \cdot \bar{b}$	$\overline{(a \cdot b)} = \bar{a} + \bar{b}$

Otras formas de notación del álgebra de Boole

En matemática se emplea la notación empleada hasta ahora ({0,1}, + , ⋅) siendo la forma más usual y la mas cómoda de representar.
Por ejemplo las leyes de De Morgan se representan así:

$$\overline{a + b} = \bar{a} \cdot \bar{b}$$
$$\overline{a \cdot b} = \bar{a} + \bar{b}$$

Cuando el álgebra de Boole se emplea en electrónica, suele emplearse la misma denominación que para las puerta lógica AND (Y), OR (O) y NOT (NO), ampliándose en ocasiones con X-OR (O exclusiva) y su negadas NAND (NO Y), NOR (NO O) y X-NOR (equivalencia). las variables pueden representarse con letras mayúsculas o minúsculas, y pueden tomar los valores {0, 1}

Empleando esta notación las leyes de De Morgan se representan:

$$\text{NOT } (a \text{ OR } b) = \text{NOT } a \text{ AND NOT } b$$
$$\text{NOT } (a \text{ AND } b) = \text{NOT } a \text{ OR NOT } b$$

En su aplicación a la lógica se emplea la notación $\wedge$ $\vee$ $\neg$y las variables pueden tomar los valores {F, V}, falso o verdadero, equivalentes a {0, 1}

Con la notación lógica las leyes de De Morgan serian así:

$$\neg(a \vee b) = \neg a \wedge \neg b$$
$$\neg(a \wedge b) = \neg a \vee \neg b$$

Desde el punto de vista práctico existe una forma simplificada de representar expresiones booleanas. Se emplean apóstrofes (') para indicar la negación, la operación suma (+) se representa de la forma normal en álgebra, y para el producto no se emplea ningún signo, las variables se representan, normalmente con una letra mayúscula, la sucesión de dos variables indica el producto entre ellas, no una variable nombrada con dos letras.

La representación de las leyes de De Morgan con este sistema quedaría así, con letra minúsculas para las variables:

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

y así, empleando letras mayúsculas para representar las variables:

$$(A+B)' = A'B'$$
$$(AB)' = A' + B'$$

Todas estas formas de representación son correctas, se utilizan de hecho, y pueden verse al consultar bibliografía. La utilización de una u otra notación no modifica el álgebra de Boole, solo su aspecto, y depende de la rama de las matemáticas o la tecnología en la que se este utilizando para emplear una u otra notación.

Ejemplo. Simplificar las siguientes expresiones

1.- A(BC + AC) + BC Distribuyendo el factor A en el paréntesis:
= ABC + AAC + BC, conmutando y aplicando idempotencia:
= ABC + BC + AC, usando absorción:
= BC +AC

2.- $\overline{\overline{XYZ+XZ}}$ Usando el Teorema de De Morgan:
= $\overline{\overline{XYZ} \cdot \overline{XZ}}$, por De Morgan nuevamente e involución:
= $(XY+\overline{Z})(\overline{X}+\overline{Z})$, distribuyendo:
= $XY\overline{X} + XY\overline{Z} + \overline{X}\overline{Z} + \overline{Z}\,\overline{Z}$, como X $\overline{X}$ es cero, y por idempotencia:
= $0 + XY\overline{Z} + \overline{X}\overline{Z} + \overline{Z}$, por absorción:
= $\overline{Z}$

3.- $\overline{\overline{(X+Y+YZW)}\overline{XY}}$ Por el teorema de De Morgan:
= $((X+Y)\ \overline{YZW})\ \overline{XY}$, nuevamente:
= $(X+Y)\ (\overline{Y}+\overline{Z}+\overline{W})\ (\overline{X}+\overline{Y})$, distribuyendo el primero con el tercer factor:
= $(X\overline{Y}+\overline{X}\overline{Y})\ (\overline{Y}+\overline{Z}+\overline{W})$, distribuyendo nuevamente
= $(X\overline{Y}+X\overline{Y}\overline{Z}+XY\overline{W}+\overline{X}YZ+\overline{X}YW$, por absorción:
=$(X\overline{Y}+\overline{X}YZ+\overline{X}YW)$.

Ejemplo. Simplificar

$X + YZ$

$$(X+Y)(X+Z)$$

$$(X+Y+ZZ)(X+Z+YY)$$

$$(X+Y+Z)(X+Y+Z)(X+Z+Y)(X+Z+Y)$$

$$(X+Y+Z)(X+Y+Z)(X+Y+Z)(X+Y+Z)$$

$$(X+Y+Z)(X+Y+Z)(X+Y+Z)$$

$$(X+Y+Z)(X+Y+Z)(X+Y+Z)$$

FUNCIONES BOOLEANAS

En forma similar a como se define en los cursos de álgebra de números reales, es posible definir una relación de dependencia de una ***variable booleana o variable lógica*** con otras variables booleanas independientes. Es decir, es posible definir ***funciones booleanas o funciones lógicas***.

Definición. Sean X1,X2,...,Xn, variables booleanas, es decir, variables que pueden tomar el valor de 0 o de 1, entonces la expresión $Y = f(X1,X2,...,Xn)$

Ejemplo: La siguiente es una función booleana

$Y= f(A,B,C) = AB + AC + A C$

Esta función se puede evaluar para diversos valores de sus variables independientes A, B, C:

Si A = 1, B = 0, C = 0 entonces $Y= f(1,0,0) = 1.0 + 0.0 + 1.1 = 1$,

Si A = 1, B = 1, C = 0 entonces $Y= f(1,1,0) = 1.1 + 0.0 + 1.1 = 1$,

Si A = 0, B = 1, C = 0 entonces $Y= f(0,1,0) = 0.1 + 1.0 + 0.1 = 0$, etc.

A diferencia de las funciones de variable real, las cuales no pueden representarse completamente usando una tabla de valores, **las funciones booleanas sí quedan totalmente especificadas por una tabla** que incluya todas las posibles combinaciones de valores que pueden tomar las variables independientes, dicha tabla se denomina ***tabla de verdad*** y es completamente equivalente a la expresión booleana, ya que incluye todas sus posibilidades.

Ejemplo. La siguiente es la tabla de verdad para la función del ejemplo anterior

X	Y	Z	$F=X+\overline{YZ}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

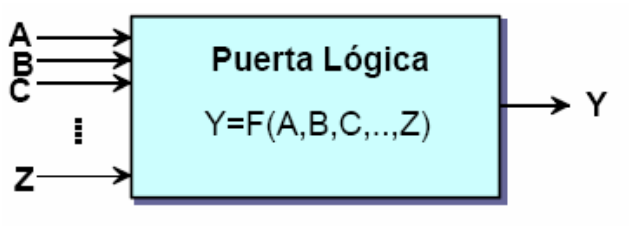
TABLA DE VERDAD, Y SU EXPRESIÓN LÓGICA (BOOLEANA).

		Const. CERO	AND		Identidad		Identidad	EXOR	OR
A	B	0	A B	$A\overline{B}$	A	$\overline{A}B$	B	$A \oplus B$	A + B
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

		NOR	EQUIVALENCIA	NOT		NOT		NAND	Const. UNO
A	B	$\overline{A+B}$	$A \odot B$	$\overline{B}$	$A+\overline{B}$	$\overline{A}$	$\overline{A}+B$	$\overline{A\ B}$	1
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

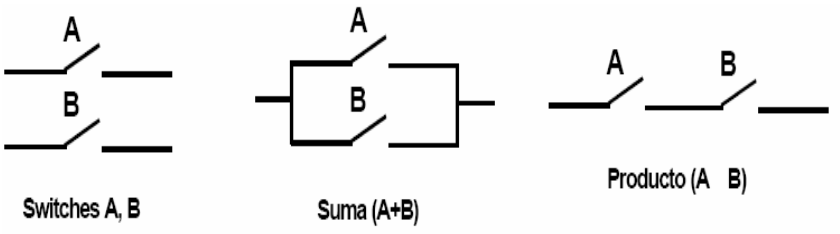
SÍMBOLOS DE PUERTAS LÓGICAS

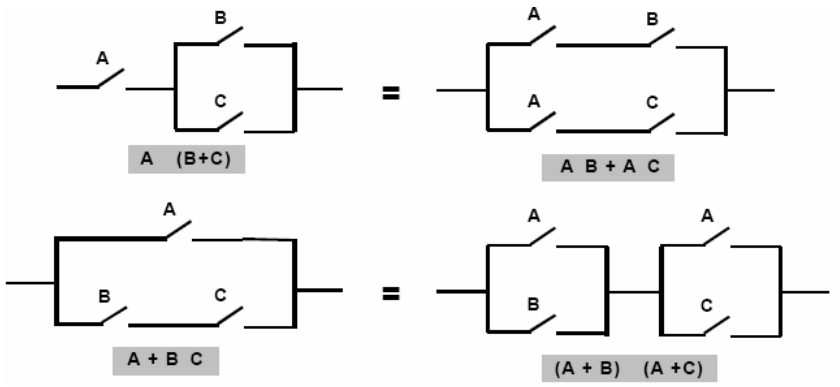
Una manera generalizada de representar las funciones lógicas es el uso de símbolos o bloques lógicos denominados ***puertas o compuertas lógicas***. Estas puertas en general representan bloques funcionales que reciben un conjunto de entradas (variables independientes) y producen una salida (variable dependiente) como se muestra en la figura siguiente.



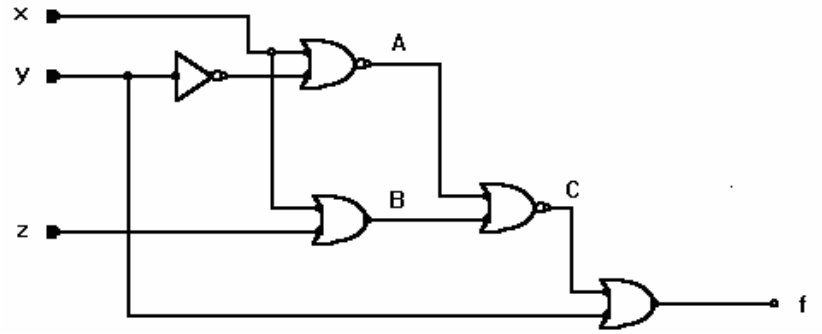
FUNCIÓN	SÍMBOLO	ECUACIÓN LÓGIC
Sumadora OR		S = a + b
Multiplicadora AND		S = a · b
Inversora NOT		S = a'
Sumadora negadora NOR		S = (a + b)'
Multiplicadora negadora NAND		S = (a · b)'
Suma exclusiva OR EXCLUSIVA		S = a' ·b + a · b'
Suma exclusiva negada NOR EXCLUSIVA		S = a ·b + a' · b'

Circuitos de conmutación





Ejemplo hallar la función f dado el siguiente circuito lógico.



Solución. Calculemos las funciones booleanas en los puntos A, B, C.
A = (x · y')' = x' y.
B = x · z.
C = (A · B)' = A' B' = (x · y')(x · z)'.
f = C · y = (x · y')x' z' · y
f = x x' z' · x' y' z' · y.
f = x' y' z' · y.
f = (y · y')(y · x' z')
f = x' z' · y.

CAPÍTULO III

MAPAS DE KARNAUGH

Los Mapas de Karnaugh son una herramienta muy utilizada para la simplificación de circuitos lógicos. Cuando se tiene una función lógica con su tabla de verdad y se desea implementar esa función de la manera más económica posible se utiliza este método.

Ejemplo: Se tiene la siguiente tabla de verdad para tres variables.
Se desarrolla la función lógica basada en ella. (primera forma canónica). Ver que en la fórmula se incluyen solamente las variables (A, B, C) cuando F cuando es igual a "1". Si A en la tabla de verdad es "0" se pone A, si B = "1" se pone B, Si C = "0" se pone C, etc.

	A	B	C	F	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	1	$\bar{A}\,B\,\bar{C}$
3	0	1	1	1	$\bar{A}\,B\,C$
4	1	0	0	1	$A\,\bar{B}\,\bar{C}$
5	1	0	1	1	$A\,\bar{B}\,C$
6	1	1	0	1	$A\,B\,\bar{C}$
7	1	1	1	1	$A\,B\,C$

www.unicrom.com

F = A B C + A B C + A B C + A B C + A B C + A B C

Una vez obtenida la función lógica, se implementa el mapa de Karnaugh.
Este mapa tiene 8 casillas que corresponden a 2^n, donde n = 3 (número de variables (A, B, C))

A \ B C	0 0	0 1	1 1	1 0
0	0 (0)	0 (1)	1 (3)	1 (2)
1	1 (4)	1 (5)	1 (7)	1 (6)

La primera fila corresponde a A = 0
La segunda fila corresponde a A = 1
La primera columna corresponde a BC = 00 (B=0 y C=0)
La segunda columna corresponde a BC = 01 (B=0 y C=1)

La tercera columna corresponde a BC = 11 (B=1 y C=1)

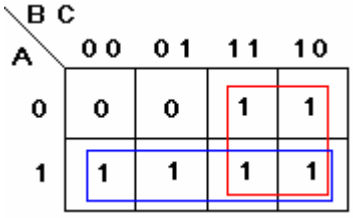
La cuarta columna corresponde a BC = 10 (B=1 y C=0)

En el mapa de Karnaugh se han puesto "1" en las casillas que corresponden a los valores de F = "1" en la tabla de verdad. Tomar en cuenta la numeración de las filas de la tabla de verdad y la numeración de las casillas en el mapa de Karnaugh

Para proceder con la simplificación, se crean grupos de "1"s que tengan 1, 2, 4, 8, 16, etc. (sólo potencias de 2) . Los "1"s deben estar adyacentes (no en diagonal) y mientras más "1"s tenga el grupo, mejor.

La función mejor simplificada es aquella que tiene el menor número de grupos con el mayor número de "1"s en cada grupo

Se ve del gráfico que hay dos grupos cada uno de cuatro "1"s, (se permite compartir casillas entre los grupos).



La nueva expresión de la función boolena simplificada se deduce del mapa de Karnaugh.

- Para el primer grupo (cuadro): la simplificación da **B** (los "1"s de la tercera y cuarta columna) corresponden a B sin negar)
- Para el segundo grupo (horizontal): la simplificación da **A** (los "1"s están en la fila inferior que corresponde a A sin negar)

Entonces el resultado es **F = B + A ó F = A + B**

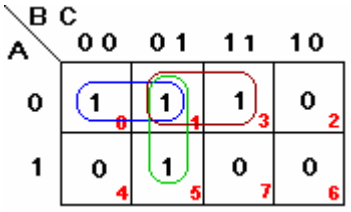
Ejemplo:

	A	B	C	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Una tabla de verdad como la de la, izquierda da la siguiente función booleana:

F = A B C + A B C + A B C + A B C

Se ve claramente que la función es un reflejo del contenido de la tabla de verdad cuando F = "1"
Con esta ecuación se crea el mapa de Karnaugh y se escogen los grupos. Se lograron hacer 3 grupos de dos "1"s cada uno.



Se puede ver que no es posible hacer grupos de 3, porque 3 no es potencia de 2. Se observa que hay una casilla que es compartida por los tres grupos.
La función simplificada es:

F = A B + A C + B C

EJERCCIOS MATEMATICAS DISCRETAS

ALGEBRA BOOLENA

1. Demuestre o refute cada una de las siguientes igualdades propuestas en un álgebra booleana:

$$\overline{x \wedge \overline{(y \vee z)}} = \overline{(x \wedge y)} \vee (x \wedge z)$$

$$x = \overline{(\bar{x} \vee \bar{y})} \vee (z \vee \overline{(x \vee y)})$$

2. Simplifique las siguientes funciones booleanas a un número mínimo de literales utilizando Álgebra Booleana.
   - x y + x y'
   - (x + y)(x + y')
   - x y z + x' y + x y z'

- z x + z x' y
- (A + B)'(A' +B')'
- y (w z' + w z) + x y

3. Simplifique las funciones T1 y T2 a un número mínimo de literales.

A	B	C	T1	T2
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

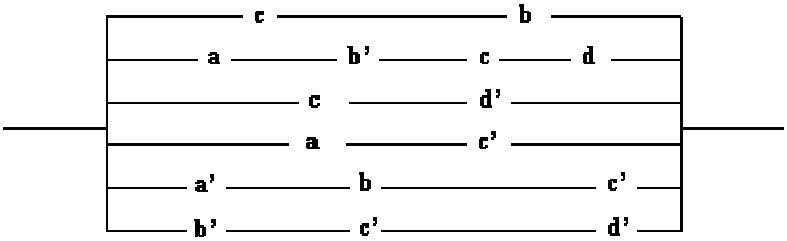
4. Implementar las funciones booleanas de los puntos 1 y 2 tanto la original como la simplificada con las compuertas lógicas.

MAPAS DE KARNAUGH

5. Realice la simplificación de la funciones Booleanas de los puntos 1 y 2, utilizando mapas de harnaugh. Además simplifique los siguientes ejercicios:

- F (x, y, z) = " (0, 2, 4, 5, 6)
- F (w, x, y, z) = " (0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)
- F (A, B, C, D) = " (0, 1, 2, 6, 8, 9, 10)
- F (A, B, C, D, E) = " (0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31)

6. Simplificar el siguiente circuito:

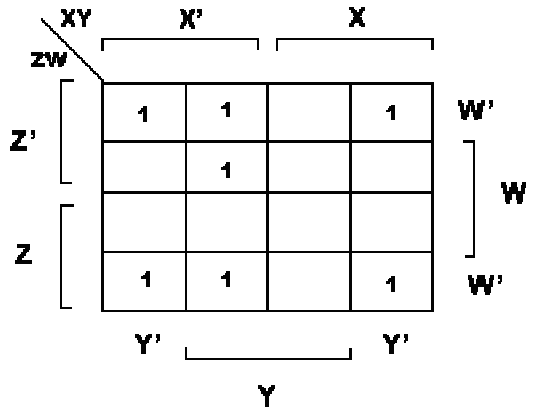


7. De la siguiente tabla deduzca la función f, llévela a un mapa de Karnaugh y simplifíquela.

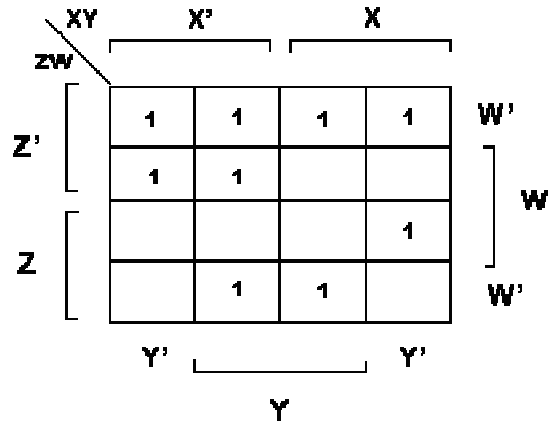
x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

8. simplificar f = x' z + x' y + x y' z + yz, usando:
   - Propiedades del álgebra Booleana.
   - Mapas de Karnaugh.

9. Del siguiente mapa de Karnaugh, deduzca la función simplificada.



10. Igual que el punto 3 deduzca las funciones más simples.



Simplifique las siguientes funciones Booleanos usando teoremas de álgebra de Booleana y mapas de Karnaugh.

- x y + (x + y)z' + y.
- x + y + [(x' + y + z)].
- y z + w x + z + [w z(x y + w z)].
- x y z + x' y z + x' y' z' + x' y' z + x y' z + x y' z'.

11. Lleve a mapas de karnaugh.
    - f = x' y z' w + y z' + x' w.
    - g = x' y' z + x' y z' + x y' z.
    - h = x y + z'.

12. De la siguiente tabla de verdad, deduzca f. Llévela a un mapa de Karnaugh y simplifíquela. Dibuje el circuito de conmutación simplificado.

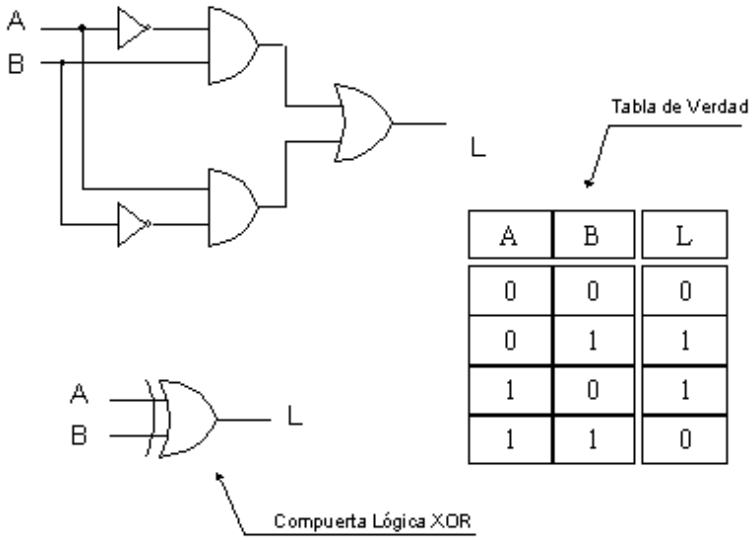
x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

CIRCUITOS LOGICOS

13. Mediante inversores y compuertas AND y OR construir las redes compuertas para:
    - f = x z'. y z'. x.
    - f = (x. z')(y. z')x'.
    - f = (x y. y z)'.

14. Simplifique las siguientes funciones Booleanos usando teoremas de álgebra de Booleana ó mapas de Karnaugh luego diseñe su circuito lógico
    - x y + (x + y)z' + y.
    - x + y + [(x' + y + z)].
    - y z + w x + z + [w z(x y + w z)].
    - x y z + x' y z + x' y' z' + x' y' z + x y' z + x y' z'.

15. Hallar la función lógica del siguiente circuito



A	B	L
0	0	0
0	1	1
1	0	1
1	1	0

CAPÍTULO IV

TECNICAS DE CONTEO

Se les denomina técnicas de conteo a: las variaciones, permutaciones y combinaciones las cuales son parte de las Matemáticas Discretas que estudia las diversas formas de realizar agrupaciones con los elementos de un conjunto, formándolas y calculando su número, existen distintas formas de realizar estas agrupaciones, según se repitan los elementos o no, según se puedan tomar todos los elementos de que disponemos o no y si influye o no el orden de colocación de los elementos, etc.

Las bases para entender el uso de las técnicas de conteo son el principio multiplicativo y el aditivo, los que a continuación se definen y se hace uso de ellos.

a) Principio Multiplicativo.- Si se desea realizar una actividad que consta de r pasos, en donde el primer paso de la actividad a realizar puede ser llevado a cabo de N1 maneras o formas, el segundo paso de N2 maneras o formas y el r-ésimo paso de Nr maneras o formas, entonces esta actividad puede ser llevada a efecto de:

N1 x N2 x .....x Nr maneras o formas

El principio multiplicativo implica que cada uno de los pasos de la actividad deben ser llevados a efecto, uno tras otro.

Ejemplo:

Una persona desea construir su casa, para lo cuál considera que puede construir los cimientos de su casa de cualquiera de dos maneras (concreto o block de cemento), mientras que las paredes las puede hacer de adobe, adobón o ladrillo, el techo puede ser de concreto o lámina galvanizada y por último los acabados los puede realizar de una sola manera ¿cuántas maneras tiene esta persona de construir su casa?

Solución:

Considerando que r = 4 pasos

N1= maneras de hacer cimientos = 2
N2= maneras de construir paredes = 3
N3= maneras de hacer techos = 2

N4= maneras de hacer acabados = 1

N1 x N2 x N3 x N4 = 2 x 3 x 2 x 1 = 12 maneras de construir la casa

b) Principio Aditivo.- Si se desea llevar a efecto una actividad, la cuál tiene formas alternativas para ser realizada, donde la primera de esas alternativas puede ser realizada de M maneras o formas, la segunda alternativa puede realizarse de N maneras o formas ..... y la última de las alternativas puede ser realizada de W maneras o formas, entonces esa actividad puede ser llevada a cabo de

M + N + ...+ W maneras o formas

Ejemplo

Una persona desea comprar una lavadora de ropa, para lo cuál ha pensado que puede seleccionar de entre las marcas Whirpool, Easy y General Electric, cuando acude a hacer la compra se encuentra que la lavadora de la marca W se presenta en dos tipos de carga (8 u 11 kilogramos), en cuatro colores diferentes y puede ser automática o semiautomática, mientras que la lavadora de la marca E, se presenta en tres tipos de carga (8, 11 o 15 kilogramos), en dos colores diferentes y puede ser automática o semiautomática y la lavadora de la marca GE, se presenta en solo un tipo de carga, que es de 11 kilogramos, dos colores diferentes y solo hay semiautomática. ¿Cuántas maneras tiene esta persona de comprar una lavadora?

Solución:

M = Número de maneras de seleccionar una lavadora Whirpool
N = Número de maneras de seleccionar una lavadora de la marca Easy
W = Número de maneras de seleccionar una lavadora de la marca General Electric

M = 2 x 4 x 2 = 16 maneras
N = 3 x 2 x 2 = 12 maneras
W = 1 x 2 x 1 = 2 maneras
M + N + W = 16 + 12 + 2 = 30 maneras de seleccionar una lavadora

VARIACIONES

Las variaciones son aquellas formas de agrupar los elementos de un conjunto teniendo en cuenta que: la selección de elementos, orden en que se colocan y la repetición de elementos.

Variaciones sin repetición

Las variaciones sin repetición de *n* elementos tomados de *p* en *p* se definen como las distintas agrupaciones formadas con ***p*** elementos distintos, eligiéndolos de entre los ***n*** elementos de que disponemos, considerando una variación distinta a otra tanto si difieren en algún elemento como si están situados en distinto orden. El número de variaciones que se pueden construir se puede calcular mediante la fórmula:

$$V_n^p = \frac{n!}{(n-p)!}$$

Ejemplo:

Si tengo 5 objetos {a, b, c, d, e}, puedo formar grupos ordenados de 3 de ellos de muchas maneras: cada grupo ordenado decimos que es una variación de estos 5 elementos de orden 3, o también, tomados de 3 en 3.

Solución:
n= 5
p=3
sin repetición

El número de *variaciones* de 5 elementos tomados de 3 en 3 se denota por V_5^3 y equivale a:

$$V_5^3 = 5.4.3 = \frac{5.4.3.2.1}{2.1} = \frac{5!}{2!} = 60$$

Si tengo 5 objetos {a, b, c, d, e} , los puedo colocar ordenadamente poniendo como primer elemento del grupo o bien la 'a' o la 'b' o la 'c' o la 'd' o la 'e'. Por tanto, hay 5 posibilidades para empezar:

a _ _
b _ _
c _ _
d _ _
e _ _

Por cada una de estas 5 posibilidades, para colocar el 2º elemento tengo 4 posibilidades: elegir una cualquiera de las letras restantes. Por ejemplo, suponiendo que he colocado 1º la 'a', tendría:

a b _
a c _
a d _
a e _

De forma que si por cada elección del 1º tengo 4 posibilidades para el 2º, en conjunto tendré para los dos primeros elementos 5x4 = 20 posibilidades.

Análogamente, para colocar el 3º elemento, tendré, por cada elección del 1º y 2º, 3 nuevas posibilidades. Por ejemplo, si había colocado 1º la 'b' y 2º la 'e', tendría las siguientes posibilidades:

b e a
b e c
b e d

Así que para el conjunto de los tres primeros elementos tengo 5x4x3 = 60 posibilidades.

Variaciones con repetición

Las variaciones con repetición de ***n*** **elementos** tomados de ***p*** en ***p*** se definen como las distintas agrupaciones formadas con ***p*** elementos que pueden repetirse, eligiéndolos de entre los ***n*** elementos de que disponemos, considerando una variación distinta a otra tanto si difieren en algún elemento como si están situados en distinto orden.

El número de variaciones que se pueden construir se puede calcular mediante la fórmula:

$$VR_n^p = n^p$$

Ejemplo:

Si tengo 5 objetos {a, b, c, d, e}, puedo formar grupos ordenados de 3 de ellos, pudiéndose repetir los objetos en un mismo grupo, de la manera siguiente: cada grupo ordenado decimos que es una *variación con repetición* de estos 5 elementos de orden 3, o también, tomados de 3 en 3.

Donde:
n = 5
p = 3
con repetición

El número de *variaciones con repetición* de 5 elementos tomados de 3 en 3 se denota por VR_5^3 y equivale a:

$VR_5^3 = 5.5.5 = 5^3 = 125$

Si tengo 5 objetos {a, b, c, d, e} , los puedo colocar ordenadamente poniendo como primer elemento del grupo o bien la 'a' o la 'b' o la 'c' o la 'd' o la 'e'. Por tanto, hay 5 posibilidades para empezar:

a _ _
b _ _
c _ _
d _ _
e _ _

Por cada una de estas 5 posibilidades, para colocar el 2º elemento tengo otras 5 posibilidades: elegir una cualquiera de las letras. Por ejemplo, suponiendo que he colocado 1º la 'a', tendría:

a a_
a b _
a c _
a d _
a e _

De forma que si por cada elección del 1º tengo 5 posibilidades para el 2º, en conjunto tendré para los dos primeros elementos 5x5 = 25 posibilidades.

Análogamente, para colocar el 3º elemento, tendré, por cada elección del 1º y 2º, 5 nuevas posibilidades. Por ejemplo, si había colocado 1º la 'b' y 2º la 'e', tendría las siguientes posibilidades:

b e a
b e b
b e c
b e d
b e e

Así que para el conjunto de los tres primeros elementos tengo 5x5x5 = 125 posibilidades.

PERMUTACIONES

Una permutación es una combinación en donde el orden es importante. La notación para permutaciones es P(n,r) que es la cantidad de permutaciones de “n” elementos si solamente se seleccionan “r”.

Permutaciones SIN repetición

Las permutaciones sin repetición de n elementos se definen como las distintas formas de ordenar todos esos elementos distintos, por lo que la única diferencia entre ellas es el orden de colocación de sus elementos. Para formar un grupo se toman todos los elementos, no hay que seleccionar unos

pocos, hay que tener en cuenta el orden en que se colocan los elementos; si se altera el orden, se tiene un grupo distinto y no se repiten los elementos dentro de un mismo grupo.

El número de estas permutaciones será:

$$P_n = n!$$

Ejemplo:

Si tengo 5 objetos {a, b, c, d, e} , los puedo colocar ordenadamente de muchas maneras, cada ordenación decimos que es una permutación de estos 5 elementos. El número de permutaciones de 5 elementos se denota por P_5 y equivale a:

$P_5 = 5.4.3.2.1 = 120$

Si tengo 5 objetos {a, b, c, d, e}, los puedo colocar ordenadamente poniendo como primer elemento del grupo o bien la 'a' o la 'b' o la 'c' o la 'd' o la 'e'. Por tanto, hay 5 posibilidades para empezar:

a _ _ _ _
b _ _ _ _
c _ _ _ _
d _ _ _ _
e _ _ _ _

Por cada una de estas 5 posibilidades, para colocar el 2º elemento tengo 4 posibilidades: elegir una cualquiera de las letras restantes. Por ejemplo, suponiendo que he colocado 1º la 'a', tendría:

a b _ _ _
a c _ _ _
a d _ _ _
a e _ _ _

De forma que si por cada elección del 1º tengo 4 posibilidades para el 2º, en conjunto tendré para los dos primeros elementos 5x4 = 20 posibilidades.

Análogamente, para colocar el 3º elemento, tendré, por cada elección del 1º y 2º, 3 nuevas posibilidades. Por ejemplo, si había colocado 1º la 'b' y 2º la 'e', tendría las siguientes posibilidades:

b e a _ _
b e c _ _
b e d _ _

Así que para el conjunto de los tres primeros elementos tengo 5x4x3 = 60 posibilidades.

Análogamente, para los cuatro primeros elementos tengo 5x4x3x2 = 120 posibilidades.
Y para los cinco, 5x4x3x2x1 = 120 colocaciones posibles.

Permutaciones CON repetición

Llamamos a las permutaciones con repetición de n elementos tomados de a en a, de b en b, de c en c, etc, cuando en los n elementos existen elementos repetidos (un elemento aparece a veces, otro b veces, otro c veces, etc) verificándose que a+b+c+...=n. Para formar un grupo se toman todos los elementos, no hay que seleccionar unos pocos, hay que tener en cuenta el orden en que se colocan los elementos; si se altera el orden, se tiene un grupo distinto y hay repetición de los elementos dentro de un mismo grupo. El número de estas permutaciones será:

$$PR_n^{a,b,c} = \frac{n!}{a!\,b!\,c!}$$

Ejemplo:

Si tengo 3 objetos {a, b, c} , los puedo colocar ordenadamente de manera que la 'a' aparezca 2 veces, la 'b' otras 2 veces y la 'c' 1 sola vez, cada uno de estos grupos decimos que es una *permutación con repetición* de estos 3 elementos.

El número de permutaciones con repetición de 3 elementos que se repiten 2 veces, 2 veces y 1 vez, teniendo por tanto cada grupo 5 elementos, se denota por $P_5^{2,2,1}$ y equivale a:

$$P_5^{2,2,1} = \frac{5!}{2!\ 2!\ 1!} = 30$$

Si los 5 objetos que aparecen en las permutaciones fueran todos distintos, pongamos $\{a_1, a_2, b_1, b_2, c\}$, en lugar de estar repetidos algunos, evidentemente estaríamos en el caso de las permutaciones ordinarias y el número de grupos sería $P_5 = 120$.

Si en uno de estos grupos cambiáramos el orden de las 'a' entre sí tendríamos una permutación distinta, pero si suprimiéramos los subíndices, entonces sería la misma. Lo mismo podríamos decir de las 'b'. Pero las distintas ordenaciones que se pueden hacer con las dos 'a' y las dos 'b' son 2! . 2! = 4, así que por cada 4 permutaciones ordinarias tenemos una permutación por repetición. Luego el número de estas últimas debe ser 120 / 4 = 30.

COMBINACIONES

Una combinación, es un arreglo de elementos en donde no nos interesa el lugar o posición que ocupan los mismos dentro del arreglo. En una combinación nos interesa formar grupos y el contenido de los mismos.

Combinaciones sin repetición.

Las **combinaciones sin repetición de *n* elementos tomados de *p* en *p*** se definen como las distintas agrupaciones formadas con ***p*** elementos distintos, eligiéndolos de entre los ***n*** elementos de que disponemos, considerando una variación distinta a otra sólo si difieren en algún elemento, (No influye el orden de colocación de sus elementos).

El número de combinaciones que se pueden construir esta dada por la fórmula:

$$C_n^m = \frac{V_n^m}{P_m} = \frac{n!}{(n-m)!m!}$$

Ejemplo:

Si tengo 5 objetos {a, b, c, d, e}, puedo formar grupos no ordenados (subconjuntos) seleccionando 3 de ellos de muchas maneras, cada grupo decimos que es una combinación de estos 5 elementos de orden 3, o también, tomados de 3 en 3. No se tiene en cuenta el orden: si cambiamos el orden de los elementos en un grupo, sigue siendo el mismo grupo.

En el apartado dedicado a la Variaciones, se ha estudiado que a partir de 5 objetos {a, b, c, d, e} tomando de 3 en 3 se pueden formar 60 variaciones (grupos ordenados). Dos variaciones pueden estar formadas con los mismos objetos pero en distinto orden, por ejemplo: " **b e a** " , " **e b a** ". Estos dos grupos son distintos considerados como variaciones, pero son el mismo considerados como combinaciones, o sea, es la misma combinación, puesto que el orden no se tiene en cuenta.

¿Cuántas variaciones hay con las mismas letras " **b e a** " y que sólo se diferencian entre sí en el orden en que están escritas? Es decir, ¿de cuántas maneras se pueden ordenar 3 letras?

$P_3 = 3! = 3.2.1 = 6$

Luego entonces por cada combinación salen 6 variaciones. Como en total hay 60 variaciones, entonces el número de combinaciones debe ser 60 / 6 = 10.

Las siguientes con combinaciones encontradas:

abc, abd, abe, acd, ace, ade, bcd, bce, bde, cde

Combinaciones con repetición

Las **combinaciones con repetición de *n* elementos tomados de *p* en *p*** se definen como las distintas agrupaciones formadas con ***p*** elementos que pueden repetirse, eligiéndolos de entre los ***n*** elementos de que disponemos, considerando una variación distinta a otra sólo si difieren en algún elemento, (No influye el orden de colocación de sus elementos). El número de combinaciones que se pueden construir se puede calcular mediante la fórmula:

$$CR_n^m = C^m_{n+m-1} = \frac{V^m_{n+m-1}}{P_m}$$

Ejemplo:

Si tengo 5 objetos {a, b, c, d, e}, puedo formar grupos tomando 3 de ellos, pudiéndose repetir los elementos en un mismo grupo, cada grupo decimos que es una *combinación con repetición* de estos 5 elementos de orden 3. *No se tiene en cuenta el orden*: si cambiamos el orden de los elementos en un grupo, sigue siendo el mismo grupo.

El número de *combinaciones con repetición* de 5 elementos tomados de 3 en 3 se denota por CR_5^3 y equivale a:

$$CR_5^3 = C_7^3 = \frac{V_7^3}{P_3} = \frac{7.6.5}{3.2.1} = 35$$

Las siguientes son combinaciones encontradas:

aaa abb acc add aee
aab abc acd ade
aac abd ace
aad abe
aae

bbb bcc bdd bee
bbc bcd bde
bbd bce
bbe

ccc cdd cee
ccd cde
cce

ddd dee
dde

eeee

EJERCICIOS DE COMBINACIÓN

Para los siguiente problemas realizar la programación en computadora

a) En una liga de baloncesto juegan 10 equipos, todos contra todos dos veces (ida y vuelta). ¿Cuántos partidos se habrán jugado al final de la misma?.

b) Con los dígitos: 1, 2, 3, 4 y 5 ¿cuántos números de cinco cifras, sin repetición, se pueden formar?. [120 números]
   A. ¿Cuántos de esos números empiezan por 1?. [24]
   B. ¿Cuántos terminan en 5?. [24]
   C. ¿Cuántos empiezan por 1 y acaban en 5?. [6]
   D. ¿Cuántos son pares?. [48]
   E. ¿Cuántos son múltiplos de 5?. [24]
   F. ¿Cuántos son mayores que 20.000?. [96]

c) Un club de baloncesto dispone de 10 jugadores de los cuales juegan 5 a la vez. ¿Cuántos equipos distintos de 5 jugadores pueden sacar el entrenador para cada partido?.

d) Con las letras de la palabra CINEMA ¿Cuántas palabras distintas, tengan sentido o no, se pueden formar?. [720]
   A. ¿Cuántas terminan en A?. [120]
   B. ¿Cuántas empiezan con N?. [120]
   C. ¿Cuántas empiezan con C y terminan en I?. [24]
   D. ¿Cuántas empiezan con vocal?. [360]
   E. ¿Cuántas tienen vocal y consonante alternadas?. [72]

e) Siete chicos e igual número de chicas quieren formar pareja para el baile. ¿Cuántas parejas distintas se pueden formar?.

f) Suponiendo que existiera 100 elementos distintos en la naturaleza y que cada sustancia estuviese formada por 3 exclusivamente. ¿Cuántas sustancias distintas tendríamos?.

PROGRAMA METODO DE CONTENO COMBINATORIA

```
<SCRIPT language=JavaScript>
function factorial(n){
//      n=Math.abs(Math.floor(n));
        var f=1;
                if (n!=0) {
                for (var i=1 ; i<(n+1) ; i++){
                        f=i*f ;
                }
        }
        return f;
}

function vsrep(n,p){
        return (factorial(n)/(factorial(n-p)*factorial(+p)));
}

function compvarsin(elem,pe){
        var devolver="-";
        var lc=elem.length;
        var numvarsin=vsrep(lc,pe);
        var devol= "COMBINACIONES sin repetición "+lc+" elementos: "+numvarsin+" (números). son: "
        var carac = new Array();
        for (var i=0; i<lc;i++){
                carac[i]=elem.substr(i,1);
        }
        switch (eval(pe)) {
        case 0:
                devol="El número de variaciones sin repetición de esos "+lc+" elementos tomados de 0 (cero) en 0 es
"+numvarsin+" y esa variación es el conjunto vacío."
        break;
        case 1:
                for (var j=0; j<lc;j++){
                        devolver=devolver+carac[j]+"-";
                }
        break;
        case 2:
                var cont=0;
                for (var j=0; j<lc;j++)
                {
                        for (var jj=j;jj<lc;jj++)
                        {
                                if (jj!=j )
                                  {  devolver=devolver+carac[j]+carac[jj]+"-";
                                        if (parseInt(carac[j])==1)
                                                cont=cont+1;
                                  }

                        }
                }

        break;
        case 3:
                for (var j=0; j<lc;j++){
                        for (var jj=j;jj<lc;jj++){
```

```
                                        for (var jjj=jj;jjj<lc;jjj++){
                                                if ((jjj!=jj)&&(jjj!=j)&&(jj!=j)){
                                                        devolver=devolver+carac[j]+carac[jj]+carac[jjj]+"-";
                                                }
                                        }
                                }
                        }
                break;
                case 4:
                        var cont=0;
                        for (var j=0; j<lc;j++){
                                for (var jj=j;jj<lc;jj++){
                                        for (var jjj=jj;jjj<lc;jjj++){
                                                for (var jjjj=jjj;jjjj<lc;jjjj++){
                                                        if
((jjjj!=jjj)&&(jjjj!=jj)&&(jjjj!=j)&&(jjj!=jj)&&(jjj!=j)&&(jj!=j)){

                devolver=devolver+carac[j]+carac[jj]+carac[jjj]+carac[jjjj]+"-";
                                                                if (parseInt(carac[j])==1)
                                                                        cont=cont+1;

                                                        }
                                                }
                                        }
                                }
                        }
                break;
                case 5:
                        for (var j=0; j<lc;j++){
                                for (var jj=j;jj<lc;jj++){
                                        for (var jjj=jj;jjj<lc;jjj++){
                                                for (var jjjj=jjj;jjjj<lc;jjjj++){
                                                        for (var jjjjj=jjjj;jjjjj<lc;jjjjj++){
                                                                if                      ((jjjjj!=jjjj)&&(jjjjj!=jjj)&&(jjjjj!=jj)&&
(jjjjj!=j)&&(jjjj!=jjj)&&(jjjj!=jj)&&(jjjj!=j)&&(jjj!=jj)&&(jjj!=j)&&(jj!=j)){

                devolver=devolver+carac[j]+carac[jj]+carac[jjj]+carac[jjjj]+carac[jjjjj]+"-";
                                                                }
                                                        }
                                                }
                                        }
                                }
                        }
                break;
                }
//              document.write(cont);
                return devol+devolver;
}

function completa(form){
        form.forvarsinrep.value=compvarsin("0123456789ABCDEFGHIJ","4");
}

 </SCRIPT>
<FORM>
 <TABLE cols=2 width="100%" border=1>
 <TBODY>
 <TR>
  <TD width="65%"><DIV
   align=right><INPUT onclick=completa(form) type=button value=Solución></DIV></TD>
  <TD width="35%"><BR><TEXTAREA  name=forvarsinrep rows=10 cols=30></TEXTAREA>

<DIV
```

CAPÍTULO V

TEORÍA DE GRAFOS Y SU APLICACIÓN

Para nadie es novedad observar en la vida cotidiana: carreteras, líneas telefónicas, líneas de televisión por cable, el transporte colectivo metro, circuitos eléctricos de nuestras casas, automóviles, y tantas cosas mas; lo que no pensamos frecuentemente es que estos forman parte de algo que en matemáticas se denomina como grafos.

En este trabajo se tratará brevemente de explicar lo que son los grafos, sus tipos, y algunas derivaciones de ellos, así como su representación gráfica y en algunos casos, su representación en algún programa informático, así como en la memoria.

APLICACIÓN A PROBLEMAS DE VIDA REAL

La mayor parte de los problemas de la teoría de grafo pueden ser aplicados a:

1. Problemas de Existencia

- El problema de los siete puentes de Königsberg: Existe una trayectoria cerrada que cruce cada uno de los siete puentes exactamente una vez?
- El problema del Caballo de Ajedrez: Existe una secuencia de los movimientos del caballo tal que visite cada cuadrado de un tablero de ajedrez exactamente una vez y regresando a la posición de partida?
- El problema de los Cuatro Colores: Puede colorearse todo mapa con cuatro colores de modo que los países vecinos tengan colores diferentes?

2. Problemas de Construcción

- Determinar si un grafo dado es euleriano y construir un camino euleriano (algoritmo de Fleury).

3. Problemas de Enumeración

- Grafos etiquetados.
- Digrafos etiquetados.

- Árboles etiquetados.

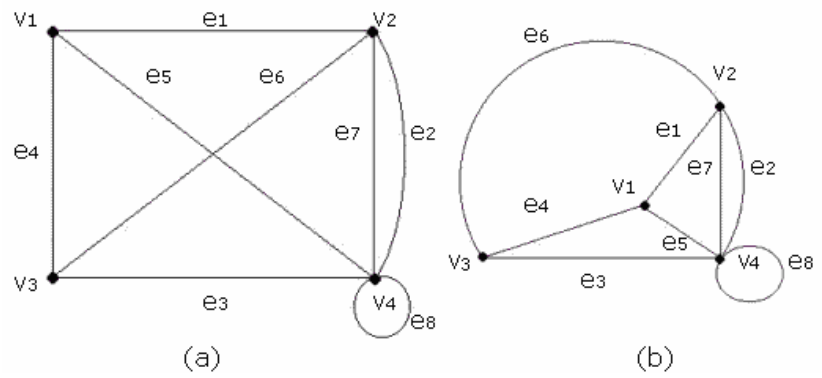
4. Problemas de Optimización

- Problema de encontrar el camino mínimo entre dos vértices en dígrafo pesado.
- Problema del viajante de comercio

CONCEPTO GRAFOS

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.



En los grafos anteriores, los vértices son v_1, v_2, v_3, v_4 mientras que los aristas son e_1, e_2, e_3, e_4, e_5, e_6, e_7. Las aristas e2 y e7 se llaman aristas paralelas porque unen un mismo par de vértices. La arista e8 se llama lazo o bucle porque une un vértice consigo mismo.

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.

- *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.

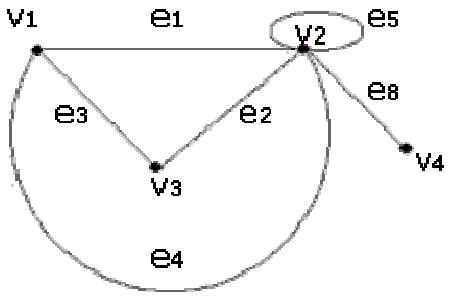
Vértices

Son los puntos o nodos con los que esta 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.

- *Vértices Adyacentes:* si tenemos un par de vértices de un grafo (U, V) y si tenemos un 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.

Ejemplo 1

Para el grafo siguiente:



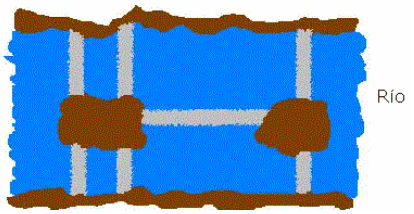
a. Escribir el conjunto de vértices.
b. Escribir el conjunto de aristas.
c. Hallar los vértices aislados.
d. Hallar los lazos.
e. Hallar las aristas paralelas.

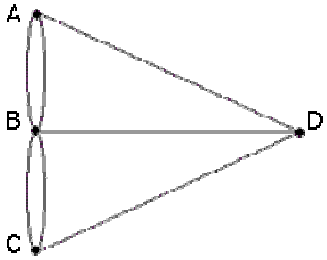
Solución

a. El conjunto de vértices es:
   $V = \{ v_1, v_2, v_3, v_4\}$
b. El conjunto de aristas es:
   $E = \{ e_1, e_2, e_3, e_4, e_5\}$
c. No hay vértices aislados.
d. e_5 es el único lazo.
e. e_1 y e_4 son aristas paralelas.

Ejemplo

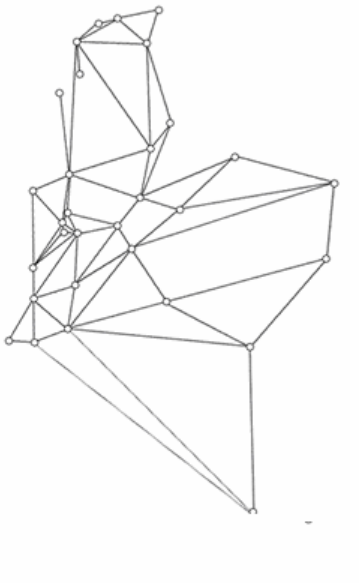
De la figura (Los puentes de Königsberg), diseñar la gráfica (grafo) correspondiente





- A y C son las orillas del río.
- B y D son las islas.
- Los siete aristas son los siete puentes

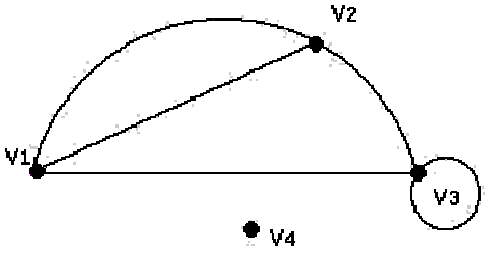
Otro Ejemplo



Definición.- Sea G = (V,E) un grafo no dirigido, dado un vértice v, se llama grado del vértice al número de aristas incidentes en él. Si existe un lazo, lo contaremos dos veces.

Ejemplo

Dado el siguiente grafo, encuentre el grado de cada vértice.

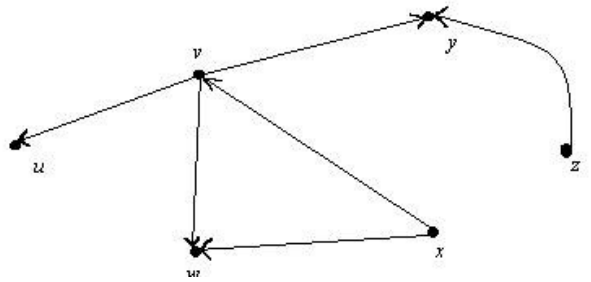


Solución

grado (v1) = 3, grado (v2) = 3

grado (v3) = 4, grado (v4) = 0

Definición.- Sea G = (V,E) un grafo (dirigidos o no) que no tienen lazos ni más de una arista adyacente al mismo par de vértices se llaman **grafos simples**.



Teorema. Sea G un grafo con vértices v_1, v_2,..., v_n. Entonces la suma de los grados de todos los vértices de G es igual a dos veces el número de aristas en G. Es decir, grad (v_1) + grad (v_2) + ……+ grad (v_n) = 2 A, donde A es el número de aristas de G.

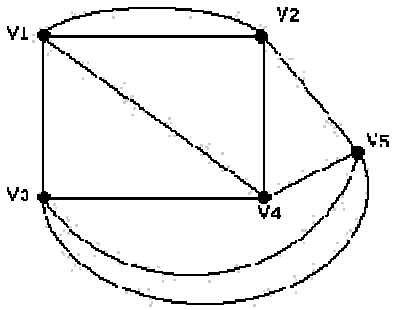
Así, 2A es el total de la suma de los grados de los vértices de G. Como consecuencia del teorema anterior se tiene que para cualquier grafo, el número de vértices de grado impar, debe ser par.

Ejemplo

¿Es posible tener un grafo, en el que cada vértice tiene grado 4 y hay 10 aristas?.

Solución

Por el teorema anterior se tiene: 2A = 20 o sea que deben existir 10 aristas. De otra parte, como los vértices tienen el mismo grado 4, se debe cumplir que, 20=4 V, donde V es el número de vértices. Por tanto V = 5. La figura siguiente muestra uno de eso grafos:



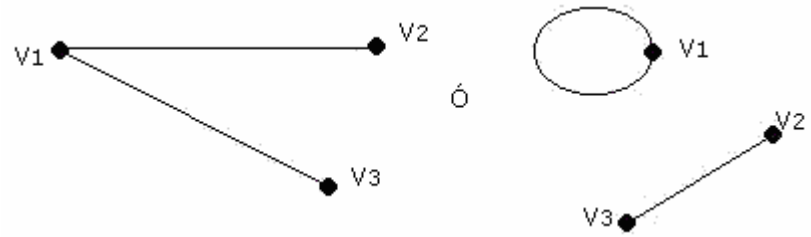
Ejemplo

¿Se puede dibujar un grafo G con tres vértices v_1 v_2 y v_3, donde,

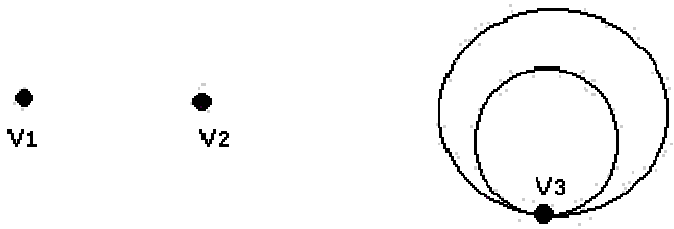
a. grad (v_1) = 1, grad (v_2) = 2, grad (v_3) = 2
b. grad (v_1) = 2, grad (v_2) = 1, grad (v_3) = 1
c. grad (v_1) = 0, grad (v_2) = 0, grad (v_3) = 4

Solución

a. No es posible porque la suma de los grados de los vértices es 5 que el un número impar.
b. Sí, porque grad (v1)+grad (v2) + grad (v3) = 4; que es un número par. El número de aristas es 2.



c. Sí, porque grad (v1) + grad (v2) + grad (v3) = 4; que es un número par. El único grafo es:



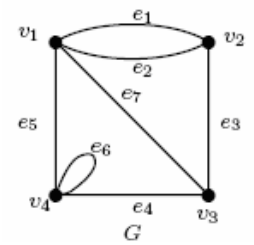
REPRESENTACIÓN DE GRAFOS EN PROGRAMAS

Representación mediante matrices: La forma más fácil de guardar la información de los nodos es mediante la utilización de un vector que indexe los nodos, de manera que los arcos entre los

nodos se pueden ver como relaciones entre los índices. Esta relación entre índices se puede guardar en una matriz, que llamaremos de adyacencia.

Sea G = (V,E) un grafo con "v" vértices y "e" aristas, entonces le corresponde una matriz denominada la matriz de incidencia de G. Si denotamos los vértices de G por v1, v2, . . . , vº y las aristas por e1, e2, . . . , e". Entonces la matriz de incidencia de G es la matriz M(G) = [mij] donde mij es el numero de veces que la arista ej incide en el vértice vi; los valores son 0,1 ó 2 (2 en el caso que la arista sea un lazo).

Otra matriz asociada a G es la matriz de adyacencia, esta es una matriz V×V A(G)[aij], en donde aij es el numero de aristas que van de vi hasta vj. A continuación damos un ejemplo de un grafo con su correspondiente matriz de incidencia y matriz de adyacencia.



G

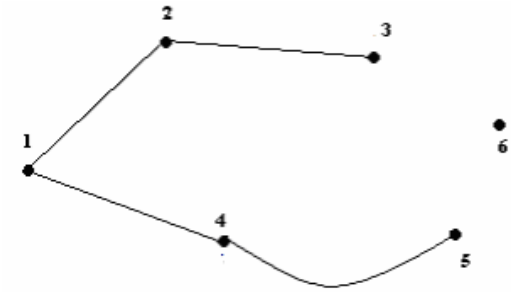
	e_1	e_2	e_3	e_4	e_5	e_6	e_7
v_1	1	1	0	0	1	0	1
v_2	1	1	1	0	0	0	0
v_3	0	0	1	1	0	0	1
v_4	0	0	0	1	1	2	0

$\mathbf{M}(G)$

	v_1	v_2	v_3	v_4
v_1	0	2	1	1
v_2	2	0	1	0
v_3	1	1	0	1
v_4	1	0	1	1

$\mathbf{A}(G)$

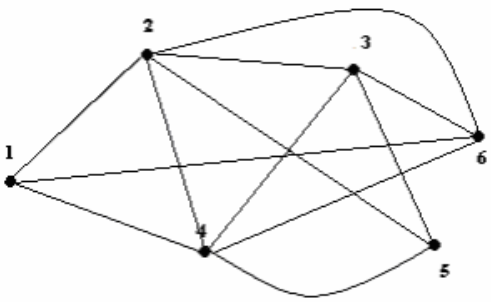
Representación mediante listas: En las listas de adyacencia lo que haremos será guardar por cada nodo, además de la información que pueda contener el propio nodo, una lista dinámica con los nodos a los que se puede acceder desde él. La información de los nodos se puede guardar en un vector, al igual que antes, o en otra lista dinámica.



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

Lista de Adyacencia

1, 2, 4
2, 1, 3
3, 2
4, 1, 5
5, 4
6



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

Lista de Adyacencia

1, 2, 4, 6
2, 1, 3, 4, 5, 6
3, 2, 4, 5, 6
4, 1, 2, 3, 5, 6
5, 2, 3, 6
6, 1, 2, 3, 4

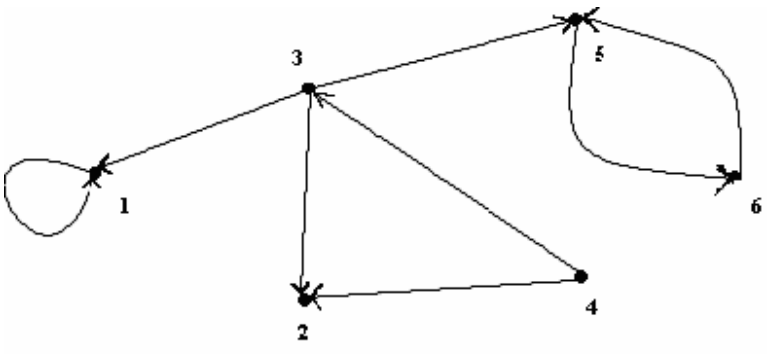
CLASIFICACION DE GRAFOS

a) Grafo dirigido (dígrafo)

Definición. Dado un grafo dirigido o dígrafo $D = (V, E)$ con n vértices $\{v_1, ..., v_n\}$ su **matriz de adyacencia** es la matriz de orden $n \times n$, $A(D)=(a_{ij})$ donde a_{ij} es el número de arcos que tienen a v_i como extremo inicial y a v_j como extremo final.

En un grafo dirigido cada arco está representado por un par ordenado de vértices, de forma que los pares (v1, v2) y (v2, v1) representan dos arcos diferentes.

Ejemplo



$$A(D)=\begin{pmatrix}1&0&0&0&0&0\\0&0&0&0&0&0\\1&1&0&0&1&0\\0&1&1&0&0&0\\0&0&0&0&0&1\\0&0&0&0&0&1\end{pmatrix}$$

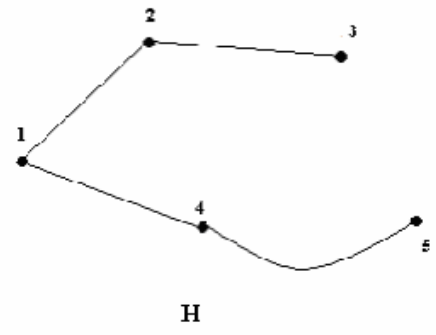
G3 = (V3, A3) V3 = {1, 2, 3,4,5,6} A3 = { (1,1),(3,1),(3,2),(3,5),(4,2),(4,3),(5,6),(6,5)}

La matriz de adyacencia de un dígrafo no es simétrica. Es una matriz binaria. El número de unos que aparecen en una fila es igual al grado de salida del correspondiente vértice y el número de unos que aparecen en una determinada columna es igual al grado de entrada del correspondiente vértice.

b) Grafo no dirigido

Definición.- En un grafo no dirigido el par de vértices que representa un arco no está ordenado. Por lo tanto, los pares (v1, v2) y (v2, v1) representan el mismo arco. su **matriz de adyacencia** es la matriz de orden $n \times n$, $A(G)=(a_{ij})$ donde a_{ij} es el número de aristas que unen los vértices v_i y v_j.

Ejemplo



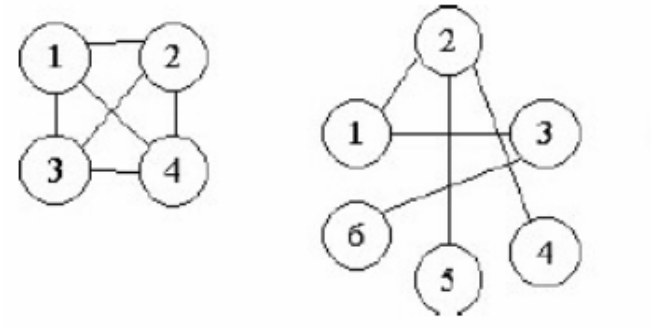
H

$$A(H) = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

La matriz de adyacencia de un grafo es simétrica. Si un vértice es aislado entonces la correspondiente fila (columna) esta compuesta sólo por ceros. Si el grafo es simple entonces la matriz de adyacencia contiene solo ceros y unos (matriz binaria) y la diagonal esta compuesta sólo por ceros.

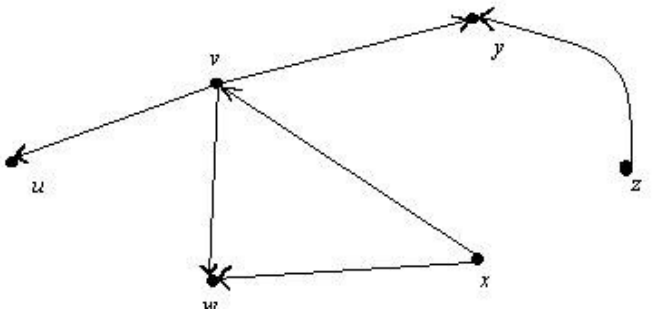
Mas ejemplos

G1 = (V1, A1) V1 = {1, 2, 3, 4} A1 = {(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)}
G2 = (V2, A2) V2 = {1, 2, 3, 4, 5, 6} A2 = {(1, 2), (1, 3), (2, 4), (2, 5), (3, 6)}



a) **Grafo simple.-** Los grafos (dirigidos o no) que no tienen lazos ni más de una arista adyacente al

mismo par de vértices se llaman grafos simples.

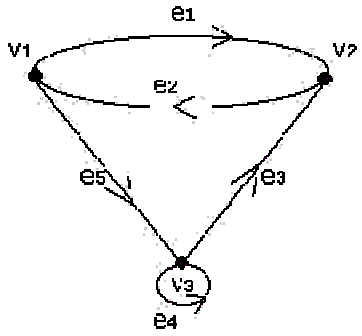


GRAFOS DIRIGIDOS O GRAFOS ORIENTADOS (DÍGRAFO)

Definición. Sea G un grafo. Si cada arista en G tiene una dirección, entonces G se llama grafo dirigido o dígrafo y sus aristas se llaman arcos. El vértice donde empieza un arco se llama punto inicial y el vértice donde termina se llama punto Terminal. Cuando no se consideran las direcciones de las aristas en G, el grafo que se obtiene se llama grafo subyacente de G.

Ejemplo 15

Dado el digrafo siguiente:



a. Dar los puntos inicial y Terminal de cada arco.

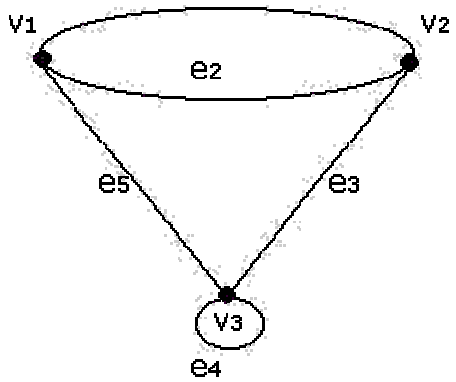
b. Dibujar el grafo subyacente.

Solución

a. La tabla siguiente detalla todos los arcos con sus puntos inicial y Terminal.

Arco	Punto Inicial	Punto Terminal
e1	v1	v2
e2	v2	v1
e3	v3	v2
e4	v3	v3
e5	v1	v3

b. El grafo subyacente es:



Definición. Sea v un vértice de un dígrafo G. el grado de entrada de v, denotado por gradent (v) es el numero de arcos en G cuyo punto terminal es v. El grado de salida de v, denotado por gradsal (v) es el numero de arcos en G cuyo punto inicial es v.

Ejemplo

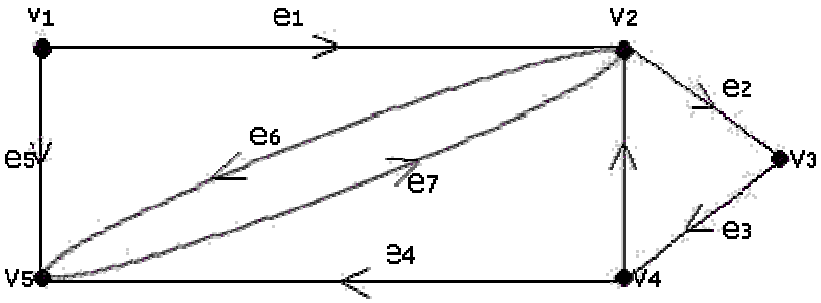
En el ejemplo anterior, los grados de entrada y de salida de cada vértice se detallan en la siguiente tabla.

Vértice	Grado entrada	Grado salida
v1	1	2
v2	2	1
v3	2	2

Definición. Una trayectoria dirigida en un dígrafo G es una sucesión de vértices y aristas de modo que el punto Terminal de un arco es el punto inicial del siguiente. Si en G existe una trayectoria orientada que va del vértice vi al vértice vk entonces se dice que vk es asequible a partir de vi .

Ejemplo

Considérese el digrafo siguiente:

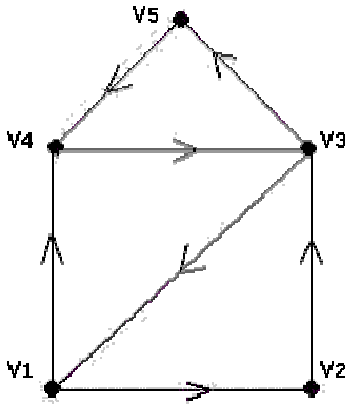


Una trayectoria dirigida de v2 a v5 es: v2 e2 v3 e3 v4 e4 v5.
v1 no es asequible desde ningún vértice porque gradent (v1) =0
v3 es asequible desde cualquier otro vértice.

Definición. Sea G un dígrafo. Si cada vértice en G es asequible a partir de cualquier otro vértice en G, entonces el dígrafo se denomina fuertemente conexo. Si el grafo subyacente de G es conexo, entonces se dice que G es débilmente conexo.

Ejemplo

El siguiente digrafo es fuertemente conexo.

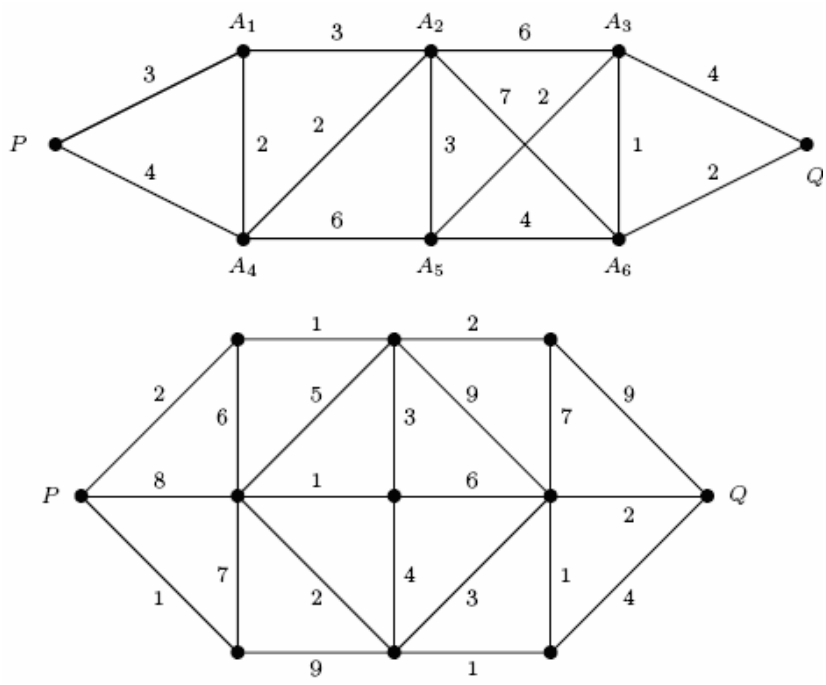


En este dígrafo cada vértice es asequible desde cualquier otro vértice.

GRAFOS ETIQUETADOS Y PONDERADOS

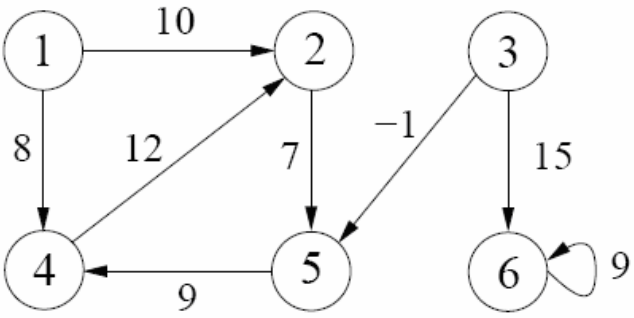
Aunque ya hemos usado los grafos etiquetados, damos una definición en esta sección. Un grafo G es un grafo etiquetado si sus aristas y/o vértices tienen asignado alguna identificación. En particular, G es un grafo ponderado si a cada arista e de G se le asigna un numero no negativo w(e) denominado peso o longitud de e. El peso (o longitud de un camino en un grafo ponderado G se

define como la suma de los pesos de las aristas del camino. Un importante problema en teoría de grafos es encontrar el camino más corto (liviano), esto es, el camino con el peso (longitud) mínimo entre dos vértices dados.



El peso de (i,j) se almacena en $A[i,j]$.

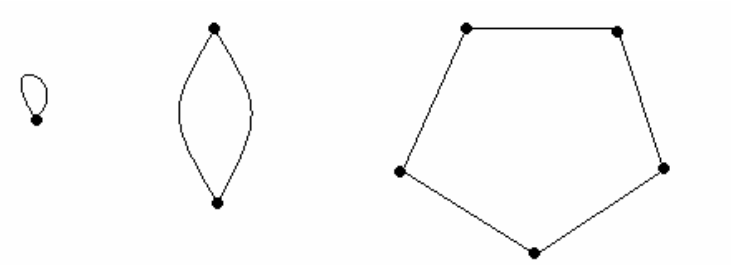
$$a_{ij} = \begin{cases} w(i,j) & \text{si } (i,j) \in E \\ 0 \ o \ \infty & \text{en cualquier otro caso} \end{cases}$$



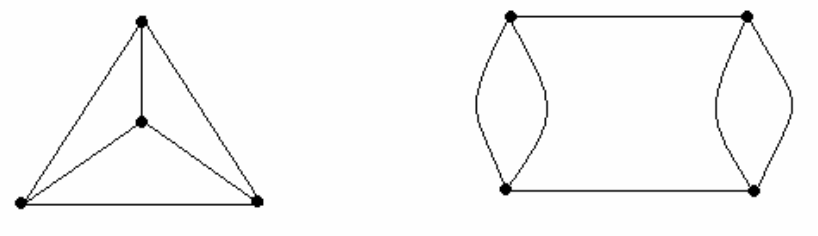
	1	2	3	4	5	6
1	0	10	0	8	0	0
2	0	0	0	0	7	0
3	0	0	0	0	−1	15
4	0	12	0	0	0	0
5	0	0	0	9	0	0
6	0	0	0	0	0	9

TIPOS DE GRAFOS

a) Es un **grafo regular** de grado *n* si todos sus vértices tienen grado *n*.



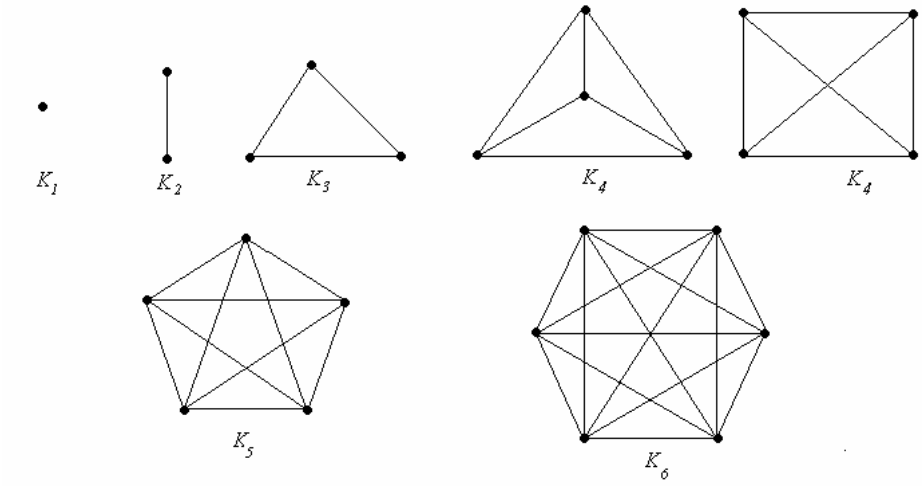
Grafos regulares de grado 2.



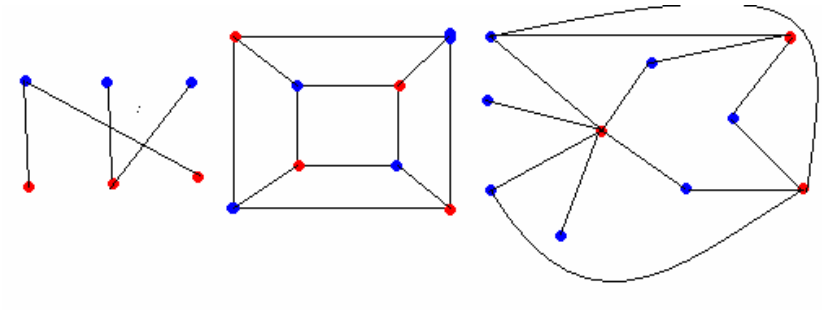
Grafos regulares de grado 3.

b) El **grafo completo** de orden n, que se denota por K_n, es el grafo que tiene n vértices y cada vértice está unido a los demás por exactamente una arista. Un grafo completo de *n* vértices tiene

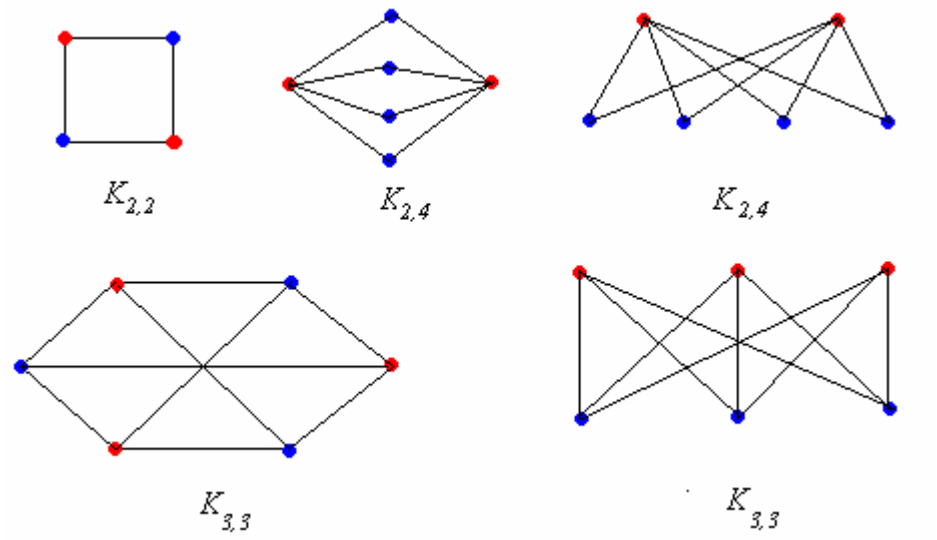
exactamente $\frac{n(n-1)}{2}$ aristas.



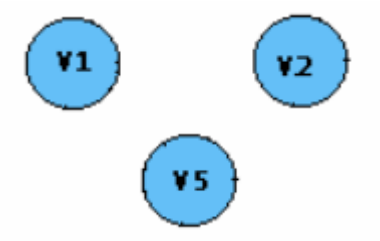
c) **Grafo bipartido.-** Es aquel con cuyos vértices pueden formarse dos conjuntos disjuntos de modo que no haya adyacencias entre vértices pertenecientes al mismo conjunto es decir un grafo G = (V,E) diremos que es un grafo bipartido si se puede dividir el conjunto de vértices en dos subconjuntos V = V1 U V2, tales que son disjuntos, V1 ∩ V2 = a conjunto vacío y **cada arista de *E* une un vértice de V_1 y otro de V_2.**



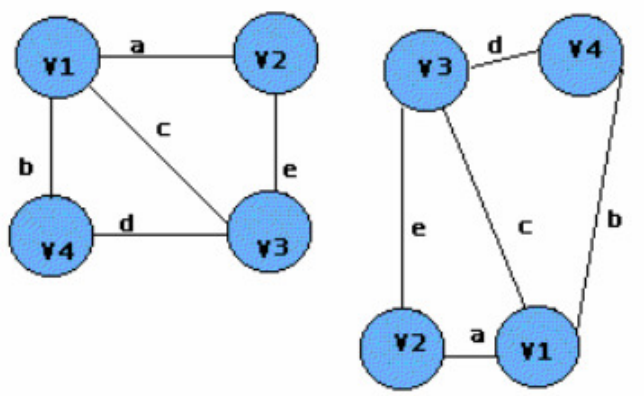
d) Un **grafo bipartido completo** si $V=V_1 \cup V_2$ y dos vértices de V están unidos por una arista de *E* si y solo si un vértice está en V_1 y el otro en V_2. Se denota por $K_{r,s}$ al grafo bipartido completo donde V_1 tiene *r* vértices y V_2 tiene *s* vértices

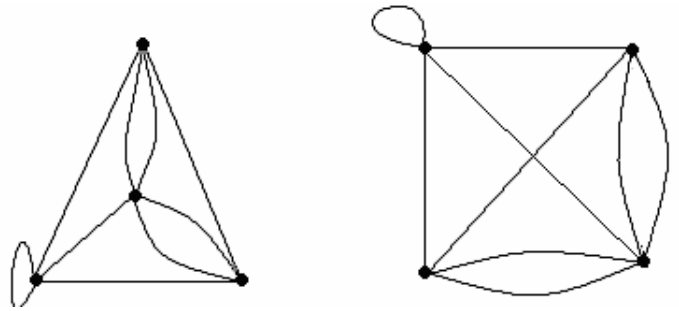


e) **Grafo nulo:** Se dice que un grafo es nulo cuando los vértices que lo componen no están conectados, esto es, que son vértices aislados.



f) **Grafos Isomorfos:** Dos grafos son isomorfos cuando existe una correspondencia biunívoca (uno a uno), entre sus vértices de tal forma que dos de estos quedan unidos por una arista en común.



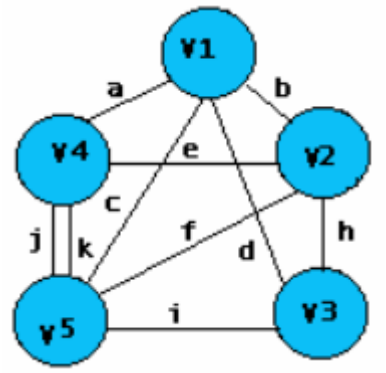


EJERCICIOS

1. Dibuje todos los grafos simples que tienen dos vértices.
2. Dibuje todos los grafos simples que tienen cuatro vértices y seis aristas.
3. Sea G un grafo con vértices v1, v2, v3, v4, v5 de grados 1, 2, 3, 4 y 5 respectivamente. ¿Cuántos aristas tiene G? Justifique su respuesta,
4. ¿Se puede dibujar un grafo simple con vértices v1, v2, v3, v4 de grados 1, 2, 3, 4 respectivamente? Justifique su respuesta.
5. Dibujar los grafos completos de orden 1, 2, 3, 4, 5.
6. ¿Cuántas aristas tiene el grafo completo de orden 6? Justifique su respuesta.

GRAFOS CONEXOS CONECTIVIDAD

Un grafo se puede definir como conexo si cualquier vértice V pertenece al conjunto de vértices y es alcanzable por algún otro. Otra definición que dejaría esto más claro sería: un grafo conexo es un grafo no dirigido de modo que para cualquier par de nodos existe al menos un camino que los une.

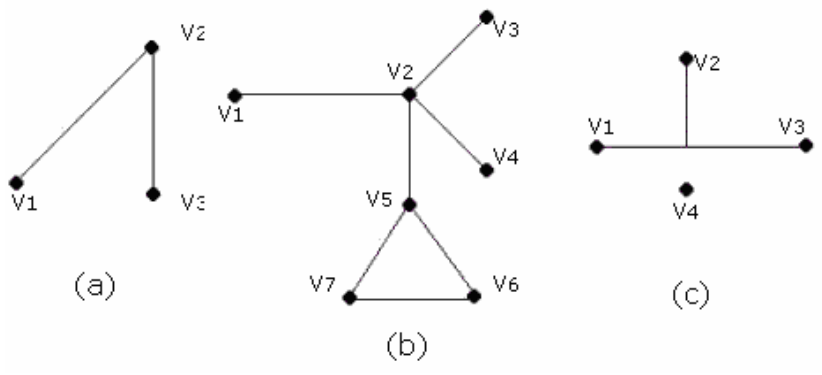


Teorema. Sea G un grafo conexo con n vértices. Entonces G debe tener al menos n -1 aristas. Si el grafo es simple y con n vértices y si tiene más de ((n-1)/2) aristas, entonces el grafo es conexo.

Definición. Sea G un grafo. Se dice que G es un grafo conexo si para cada par de vértices vi, vj en G, existe una trayectoria entre v_i y v_j.

Ejemplo

¿Cuál de los grafos siguientes es conexo?

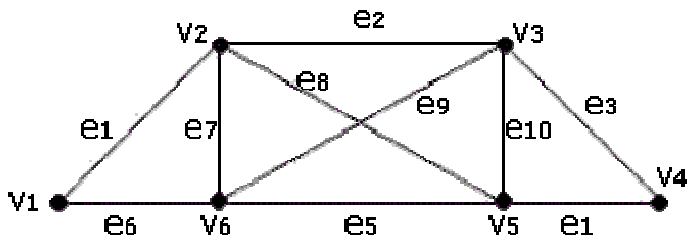


Solución

a. Conexo.
b. Conexo.
c. No es conexo.

Ejercicios

1. Dado el grafo siguiente:



Hallar:
a. Cuatro trayectorias simples diferentes.
b. Cuatro circuitos diferentes
c. Cuatro ciclos

2. Dibuje un circuito simple que consista en:
   a. Una sola arista.
   b. Sólo dos aristas.
3. Si G es un grafo simple con:
   - Seis vértices y once aristas, ¿Puede ser inconexo? ¿Por qué?
   - Seis vértices y diez aristas, ¿Puede ser inconexo? ¿Por qué?

TRAYECTORIAS O CAMINOS Y CIRCUITOS O CICLOS.

Definición. Sean v_i y v_j dos vértices de un grafo G. Una trayectoria o camino de v_i a v_j es una sucesión alternada de vértices y aristas de G que comienza en vi y termina en vj. Sí $v_i = v_j$ entonces la trayectoria es trivial, sin aristas y se denota por v_i ó v_j.

Definición. Sí una trayectoria o camino de v_i a v_j no tiene vértices repetidos, se llama **trayectoria simple**. Un circuito o ciclo es una trayectoria o camino que empieza y termina en el mismo vértice y no tiene aristas repetidas. El circuito se llamará simple si no tiene aristas ni vértices repetidos, excepto el primero y el último.

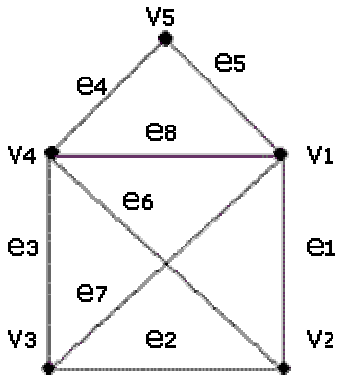
Definición .- Dado un camino de extremos v y w en un grafo no dirigido (V,E), sino se repite ninguna arista diremos que es un **recorrido.** Un recorrido cerrado, es decir, un recorrido tal que v=w será un circuito. Cuando ningún vértice del grafo se repite en un camino, se dice que es un

camino simple. Si el único vértice que se repite es el extremo se dice **Ciclo** o **camino simple cerrado.**

Vértices repetidos	Aristas repetidas	Abierto	Nombre
Sí	Sí	Sí	Camino
Sí	Sí	No	Camino cerrado
Sí	No	Sí	Recorrido
Sí	No	No	Circuito
No	No	Sí	Camino simple
No	No	No	Ciclo

Ejemplo

Dado el siguiente grafo, determinar cuál de las sucesiones siguientes son trayectorias, trayectorias simples, circuitos y circuitos simples.



a. $v_1\ e_1\ v_2\ e_6\ v_4\ e_3\ v_3\ e_2\ v_2$
b. $v_1\ e_8\ v_4\ e_3\ v_3\ e_7\ v_1\ e_8\ v_4$
c. $v_2\ e_2\ v_3\ e_3\ v_4\ e_4\ v_5\ e_5\ v_1\ e_1\ v_2$

Solución

a. Es una trayectoria de v_1 a v_2, no es simple.
b. Es una trayectoria de v_1 a v_4, no es simple.
c. Es un circuito simple.

GRAFOS EULERIANOS

Un camino euleriano se define de la manera más sencilla como un camino que contiene todos los arcos del grafo, sea G=(V,E) un grafo no dirigido, un recorrido que recorra las aristas de E se llama recorrido auleriano, Un circuito que contiene todas las aristas de G recibe el nombre de circuito euleriano. Lo anterior quiere decir que un circuito euleriano es una trayectoria que **empieza y termina en el mismo vértice, pasa por cada vértice al menos una vez y sólo una vez por cada arista**.

Existe un criterio preciso para saber cuando un grafo admite un circuito euleriano. Este criterio lo proporciona el siguiente teorema.

Teorema. Sea G un grafo. G contiene un circuito euleriano sí y sólo sí:

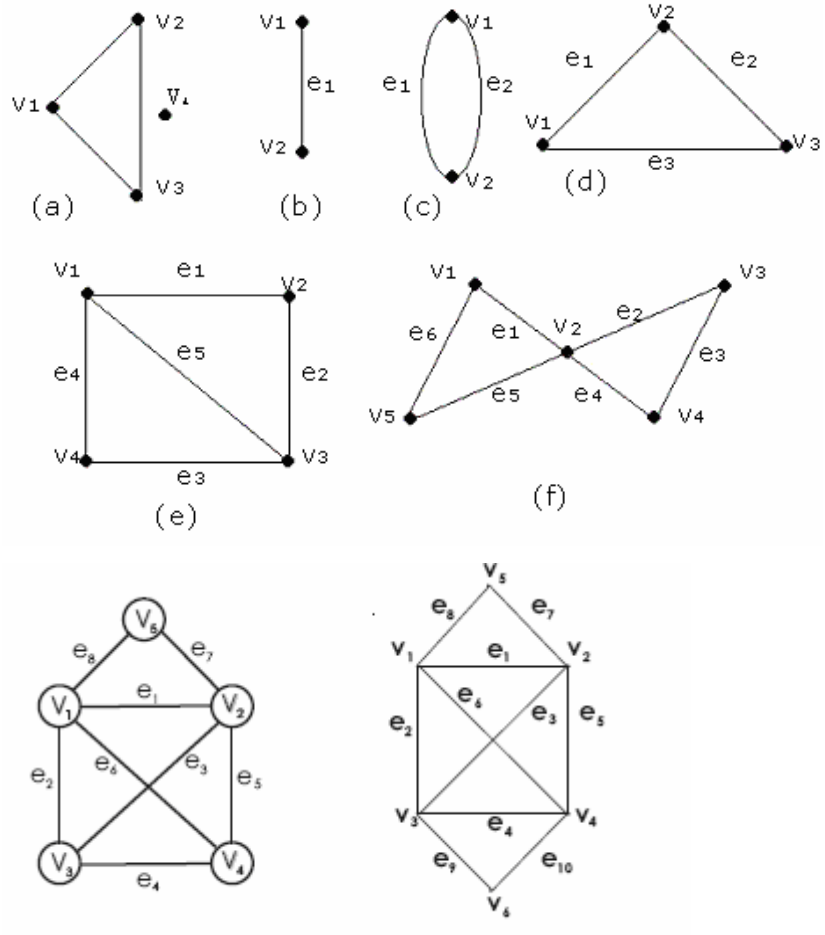
- G es conexo.

- Cada vértice de G es de grado par.

Si G tiene un ciclo de euler, para todo v_i, v_j ε V existe una trayectoria que hace parte del ciclo. Entonces G es conexo. Sea v_i el vértice donde comienza el circuito de euler. Para cualquier otro vértice v_k de G, cada vez que el ciclo llegue allí, partirá de ese vértice. Así, el circuito ha pasado por dos aristas nuevas con él o por un lazo de él. En cada caso se añade 2 al grado de ese vértice. Como este vértice v_k no es punto inicial se añade 2 cada vez que el ciclo pasa por v_k, de modo que el grado de v_k es par. En el vértice inicial v_i, la primera arista del ciclo debe ser distinta de la última, y de cualquier otra que pase por v_i, por tanto se tiene que el grado de v_i también es par.

Ejemplo

En los grafos siguientes, cuales admiten circuitos eulerianos?



Solución

a. No lo admite porque v_4 es un vértice aislado.
b. No lo admite porque cualquier ciclo utilizará la arista e1 dos veces.
c. El circuito v_1 e_1 v_2 e_2 v_1 es euleriano.
d. El circuito v_3 e_3 v_1 e_1 v_2 e_2 v_3 es euleriano.

e. No admite ningún circuito euleriano.
f. v_1 e_1 v_2 e_2 v_3 e_3 v_4 e_4 v_2 e_5 v_5 e_6 v_1 es un circuito euleriano.

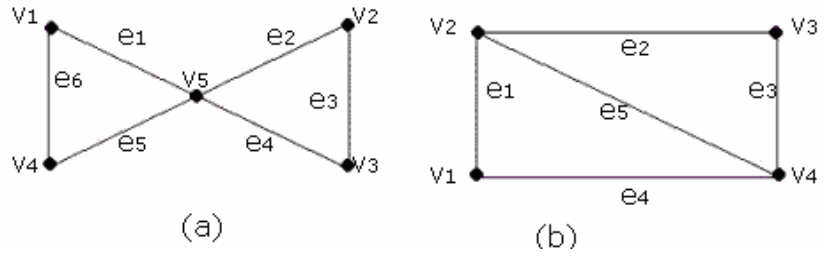
CAMINOS HAMILTONIANOS

Un ciclo es un camino, es decir una sucesión de aristas adyacentes, donde no se recorre dos veces la misma arista, y donde se regresa al punto inicial. Un ciclo hamiltoniano tiene además que recorrer todos los vértices exactamente una vez (excepto el vértice del que parte y al cual llega). Por ejemplo, en un museo grande, lo idóneo sería recorrer todas las salas una sola vez, esto es buscar un ciclo hamiltoniano en el grafo que representa el museo (los vértices son las salas, y las aristas los corredores o puertas entre ellas).

Definición. Un circuito o ciclo hamiltoniano es un ciclo simple que contiene todos los vértices de G. Lo anterior quiere decir que un circuito hamiltoniano es una trayectoria que empieza y termina en el mismo vértice, no tiene aristas repetidas y pasa por cada vértice una sola vez.

Ejemplo

¿Cuál de los grafos siguientes admite un circuito hamiltoniano?



Solución

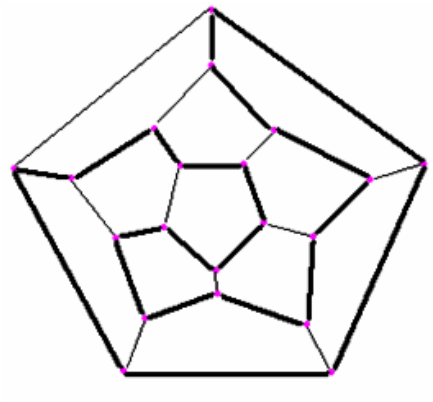
a. No admite circuitos hamiltonianos. El razonamiento es el siguiente: Si se empieza en v_1, v_2, v_3, v_4 y si se está en los demás vértices, en el v_5 se estará dos veces.

   Si se empieza en v_5, para luego ir a los vértices v_1 o v_4 ó a v_3 o v_2 respectivamente, se tendrá que pasar de nuevo por v_5 (puesto que se empezará en v_5). Para completar el circuito, se debe regresar a v_5, por lo que se pasa tres veces por él.

b. Un ciclo hamiltoniano es:

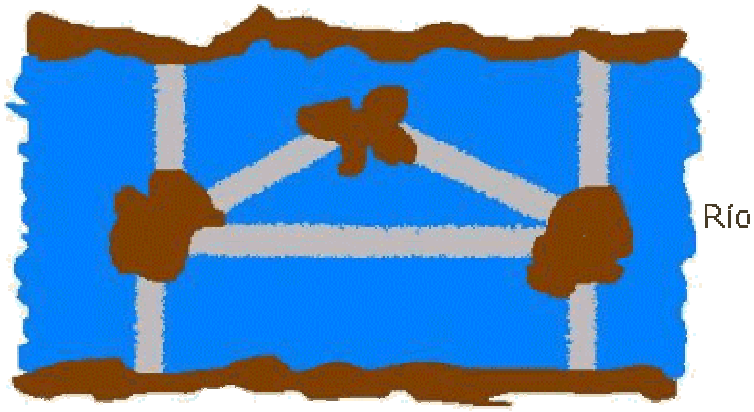
   v_1 e_1 v_2 e_2 v_3 e_3 v_4 e_4 v_1

Teorema. Sea G un grafo conexo con n vértices, donde $n \geq 3$. Si la suma de los grados de cada par de vértices no adyacentes es mayor o igual a n, entonces G tiene un circuito hamiltoniano.



Ejercicios

1. ¿Contiene un circuito euleriano el grafo completo k4?
2. ¿Contiene un circuito euleriano el grafo completo k5?
3. Una ciudad consiste en dos masas de tierra, situadas en ambas orillas de un río que tiene islas y puentes como lo detalla la gráfica siguiente:



4. ¿Hay una forma de empezar en cualquier punto para hacer un viaje redondo por todas los masas de tierra y pasar exactamente una vez por cada puente? ¿Cómo puede hacerse?

RECORRIDO DE UN GRAFO

Recorrer un grafo significa tratar de alcanzar todos los nodos que estén relacionados con uno que llamaremos nodo de salida. Existen básicamente dos técnicas para recorrer un grafo: el recorrido en anchura; y el recorrido en profundidad.

Recorrido en anchura: El recorrido en anchura supone recorrer el grafo, a partir de un nodo dado, en niveles, es decir, primero los que están a una distancia de un arco del nodo de salida, después los que están a dos arcos de distancia, y así sucesivamente hasta alcanzar todos los nodos a los que se pudiese llegar desde el nodo salida.

Recorrido en profundidad: el recorrido en profundidad trata de buscar los caminos que parten desde el nodo de salida hasta que ya no es posible avanzar más. Cuando ya no puede avanzarse más sobre el camino elegido, se vuelve atrás en busca de caminos alternativos, que no se estudiaron previamente.

ALGORITMO DE FLOY WARSHALL

El problema que intenta resolver este algoritmo es el de encontrar el camino más corto entre todos los pares de nodos o vértices de un grafo. Esto es semejante a construir una tabla con todas las distancias mínimas entre pares de ciudades de un mapa, indicando además la ruta a seguir para ir de la primera ciudad a la segunda. Este es uno de los problemas más interesantes que se pueden resolver con algoritmos de grafos.

ALGORITMO FLOY WARSHAL

```
Function(n,matriz[])
Para k = '0' hasta n hacer
   Para i = '0' hasta n hacer
      Para j = '0' hasta n hacer
         // ALGORITMO ORGINAL DE WARSHALL
         // A[i,j] = mínimo(A[i,j],A[i,k] + A[k,j])
         si  matriz[i][j] > matriz[i,k] + matriz[k,j] then
              matriz[i][j] = matriz[i,k] + matriz[k,j]
         fin de si
           Fin de para
  Fin de para
     Fin de para
```

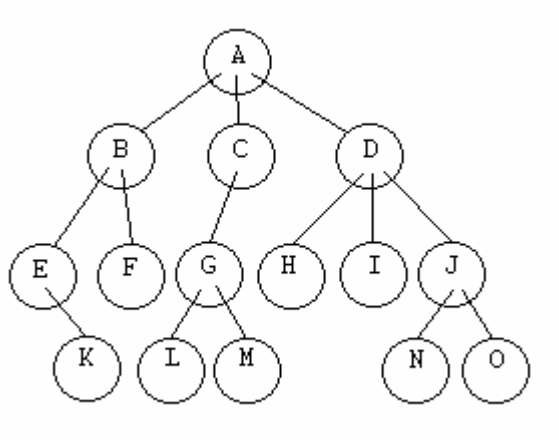
CODIGO EN C++

```
// Programa Floyd Warshall
// por: Hugo David Calderon Vilca
void warshall(int n, int matriz[20][20])
{  int i, j, k;
    float dist;
   for (k=0; k<n; k++)
         for (i=0; i<n; i++)
           for (j=0; j<n; j++)
                  if (matriz[i][j] > matriz[i][k] + matriz[k][j])
                  matriz[i][j] = matriz[i][k] + matriz[k][j];
         }
```

CAPÍTULO VI

ÁRBOLES

Un árbol es una estructura no lineal en la que cada nodo puede apuntar a uno o varios nodos. También se suele dar una definición recursiva: un árbol es una estructura en compuesta por un dato y varios árboles.



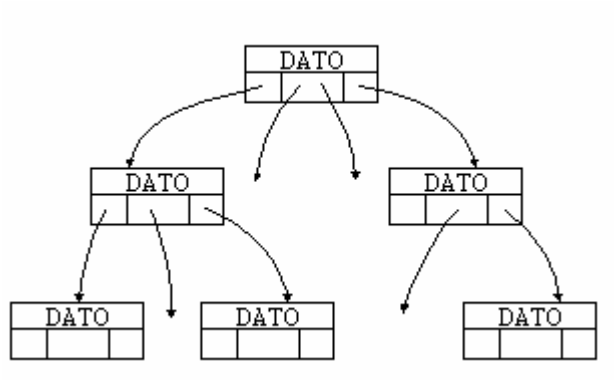
Definiremos varios conceptos. En relación con otros nodos:

- **Nodo hijo:** cualquiera de los nodos apuntados por uno de los nodos del árbol. En el ejemplo, 'L' y 'M' son hijos de 'G'.
- **Nodo padre:** nodo que contiene un puntero al nodo actual. En el ejemplo, el nodo 'A' es padre de 'B', 'C' y 'D'.

Los árboles con los que trabajaremos tienen otra característica importante: cada nodo sólo puede ser apuntado por otro nodo, es decir, cada nodo sólo tendrá un padre. Esto hace que estos árboles estén fuertemente jerarquizados, y es lo que en realidad les da la apariencia de árboles.
En cuanto a la posición dentro del árbol:

- **Nodo raíz:** nodo que no tiene padre. Este es el nodo que usaremos para referirnos al árbol. En el ejemplo, ese nodo es el 'A'.
- **Nodo hoja:** nodo que no tiene hijos. En el ejemplo hay varios: 'F', 'H', 'I', 'K', 'L', 'M', 'N' y 'O'.
- **Nodo rama:** son los nodos que no pertenecen a ninguna de las dos categorías anteriores. En el ejemplo: 'B', 'C', 'D', 'E', 'G' y 'J'.

Ejemplo de aplicación de árboles



Características del árbol, en relación a su tamaño:

- **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.
- **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.
- **Nivel**: se define para cada elemento del árbol como la distancia a la raíz, medida en nodos. El nivel de la raíz es cero y el de sus hijos uno. Así sucesivamente. En el ejemplo, el nodo 'D' tiene nivel 1, el nodo 'G' tiene nivel 2, y el nodo 'N', nivel 3.
- **Altura**: la altura de un árbol se define como el nivel del nodo de mayor nivel. Como cada nodo de un árbol puede considerarse a su vez como la raíz de un árbol, también podemos hablar de altura de ramas. El árbol del ejemplo tiene altura 3, la rama 'B' tiene altura 2, la rama 'G' tiene altura 1, la 'H' cero, etc.

Definición 1: Sea G = (V,E) un grafo no dirigido, diremos que G es un árbol T, si G es conexo y acíclico.

Definición 2: Diremos que T es un árbol generador de un grafo G si T es árbol y subgrafo generador de G.

Ejercicio: Buscar un árbol generador del grafo siguiente.

Teorema

1. En un árbol, dos vértices cualesquiera están unidos por un único camino.
2. Un grafo G es conexo si y sólo si tiene un árbol generador.
3. Si G es un árbol, entonces el número de aristas es igual al número de vértices menos uno.
4. Todo árbol T no trivial (más de 1 vértice) tiene al menos dos vértices de grado 1.

ÁRBOLES CON RAIZ O ENRAIZADOS

Definición 3: Sea T un árbol. Eligiendo un vértice r_0 de T que llamamos raíz, al ser el árbol conexo, todo otro vértice estará conectado con r_0.

Definición 4: Sea T un árbol enraizado y u un vértice de T. Llamamos nivel del vértice *u* a la longitud del camino que va de la raíz a dicho vértice. La altura de un árbol es el valor del nivel máximo.

Definición 5: Sea T un árbol con raíz r_0 y un vértice del árbol T es una hoja si está en el nivel i, y no es adyacente a ningún vértice a ningún del nivel i+1. un vértice que no es una hoja, se llama vértice interno.

Ejercicio: Construir dos árboles con raíz no isomorfos con 12 vértices, 6 hojas y altura 4.

Definición 5: Sea T un árbol con raíz r_0. Supongamos que x, y, z son vértices de T y que $v_0\ v_1\ \ldots\ v_{n-1}\ v_n$ es un camino en T. Entonces:

- v_{n-1} es el padre de v_n.
- v0, . . . vn−1 son los antepasados de vn.
- vn es el hijo de vn−1.
- Si x es un antepasado de y, entonces y es un descendiente de x.
- Si x e y son hijos de z, entonces x e y son hermanos.
- Si x no tiene hijos diremos que es un vértice Terminal.
- Si x no es un vértice Terminal diremos que es interno.
- El subgrafo de T que consiste en x y todos sus descendientes, con x como raíz se llama subárbol de T que tiene a x como raíz.

Ejercicio: Dibujar un árbol con raíz y determinar los padres, hijos, hermanos, hojas, vértices internos y número de niveles.

TIPOS DE ÁRBOLES

- Árboles Binarios
- Árbol de búsqueda binario auto-balanceable
- Árboles Rojo-Negro

- Árboles AVL
- Árboles B
- Árboles Multicamino

ÁRBOLES BINARIOS

Un árbol binario es una estructura de datos en el cual cada nodo tiene como máximo dos nodos hijos. Típicamente los nodos hijos son llamados izquierdo y derecho. Usos comunes de los árboles binarios son los árboles binarios de búsqueda y los montículos binarios.

En teoría de grafos, se usa la siguiente definición: «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 sólo existe un camino entre un par de nodos.

Un árbol binario con enraizado es como un grafo que tiene uno de sus vértices, llamado raíz, de grado no mayor a 2. Con la raíz escogida, cada vértice tendrá un único padre, y nunca más de dos hijos

Definición 6: Un árbol binario es un árbol enraizado en el cual cada vértice tiene un hijo a la derecha, o un hijo a la izquierda, o un hijo a la derecha y un hijo a la izquierda, o bien ningún hijo.

Definición 7: Un árbol binario completo es un árbol binario en el que cada vértice tiene un hijo a la derecha y otro a la izquierda o bien ningún hijo.

Teorema

1. Si T es un árbol binario completo con i vértices internos, entonces T tiene i+1 vértices terminales y 2i+1 vértices en total.
2. Sea T un árbol binario de altura h y con t vértices terminales, entonces $t \leq 2h$.

Definición 8: Un árbol binario de búsqueda es un árbol binario T en donde se han asociado datos a los vértices. Los datos se disponen de manera que para cualquier vértice v en T, cada dato en el subárbol a la izquierda (derecha, respectivamente) de v es menor que (mayor que, respectivamente) el dato correspondiente a v.

RECORRIDOS SOBRE ÁRBOLES BINARIOS

Recorrido en un árbol binario permite rescatar los datos en formas diferentes. Aunque existen varias maneras de hacerlo, aquí se verán las más conocidas : inorden , preorden , postorden. La técnica que usualmente se usa para hacer el recorrido, es ir almacenando los datos en una estructura lineal: **Cola** , **Lista** o **Pila**.

El criterio para escoger una de las tres depende del problema , pero generalmente los criterios generales son los siguientes :

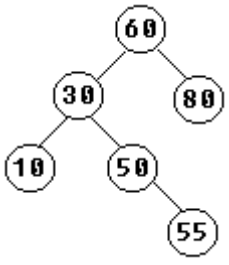
Cola : los datos quieren ser vistos en el mismo orden en el cual fueron recorridos y la cola pasa a ser un instrumento de almacenamiento de "corto plazo" : (almacenar , ver , vaciar).

Lista : los datos necesitan ser almacenados y se requieren operaciones en donde es necesario acceder a los datos en cualquier posición.

Pila : se necesita que los datos se almacenen en forma de pila, pasando a ser un instrumento de almacenamiento de "corto plazo".

Las implementaciones de recorrido serán usando una Cola , ya que los problemas que vienen, requieren los datos en forma ordenada.

Ejemplo



Recorridos

inorden : 10 , 30 , 50 , 55 , 60 , 80

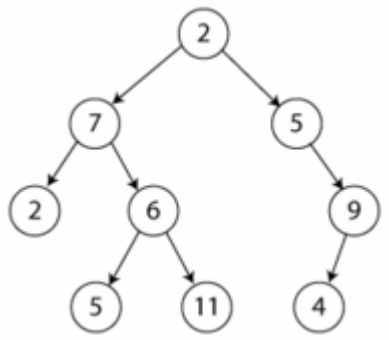
preorden : 60 , 30 , 10 , 50 , 55 , 80

postorden : 10 , 55 , 50 , 30 , 80 , 60

Recorridos en profundidad

i) Recorrido en inorden

recorrer en inorden el subárbol izquierdo, visitar el elemento de la raíz y luego recorrer en inorden el subárbol derecho. Ejm en el grafo siguiente el recorrido es: 2, 7, 5, 6, 11, 2, 5, 4 y 9.



ALGORITMO INORDEN(v)

- Paso 1. Listar el subárbol de la izquierda [Utilizar INORDEN(w) para el hijo w a la izquierda de v].
- Paso 2. Listar el subárbol de la derecha [Utilizar INORDEN(w) para el hijo w a la derecha de v].
- Paso 3. Listar T_v poniendo en una sucesión las listas del paso 1, después v y luego el resultado del paso 2. Si v no tiene hijos, la lista de T_v es solamente v.

SEUDOCÓDIGO

```
funcion inorden(nodo)
inicio
   si(existe(nodo))
      inicio
         inorden(hijo_izquierdo(nodo));
         tratar(nodo);                  //Realiza una operación en nodo
         inorden(hijo_derecho(nodo));
      fin;
fin;
```

ii) Recorrido en preorden

Paso 1. Listar los subárboles con los hijos de v como raíz [Utilizar PREORDEN(w) para listar T para cada hijo w de v].

Paso 2. Listar T_v poniendo en sucesión v seguido por las listas del paso 1 en el orden de izquierda a derecha. Si v no tiene hijos, la lista de T_v es solamente v.

visitar el elemento de la raíz , recorrer en preorden el subárbol izquierdo y luego recorrer en preorden el subárbol derecho. El recorrido es 2, 7, 2, 6,5,11,5, 9, 4;

iii) Recorrido en porstorden

Paso 1. Listar los subárboles con los hijos de v como raíz [Utilizar POSTORDEN(w) para listar T para cada hijo w de v].

Paso 2. Listar T_v poniendo en sucesión las listas del paso 1 en el orden de izquierda a derecha seguidas por v. Si v no tiene hijos, la lista de T_v es solamente v.

recorrer en postorden el subárbol izquierdo, luego recorrer en postorden el subárbol derecho y finalmente visitar el elemento de la raíz. El recorrido es: 2, 5, 11, 6, 7, 4, 9, 5 y 2.

ÁRBOLES BINARIOS DE BÚSQUEDA (ABB)

Buscar un elemento.

ALGORITMO DE BÚSQUEDA

Sea T un árbol binario de búsqueda con raíz RAIZ. Si v es un vértice:

- ✓ IZQUIERDA (v) es el hijo a la izquierda de v.
- ✓ DERECHA (v) es el hijo a la derecha de v.
- ✓ Si v no tiene hijos a la izquierda haremos IZQUIERDA (v) = λ.
- ✓ Si v no tiene hijos a la derecha haremos DERECHA(v) = λ.
- ✓ VALOR(v) proporciona el dato asociado al vértice v.

- Paso 1. P := RAIZ
- Paso 2. Si P = λ , STOP. En otro caso si VALOR (P) = W, STOP (P es el vértice que contiene el dato W.)
- Paso 3. Si W > VALOR(P), tómese
- P :=DERECHA(P), e ir a 2.
- En otro caso, t´omese

  P :=IZQUIERDA(P), e ir a 2.

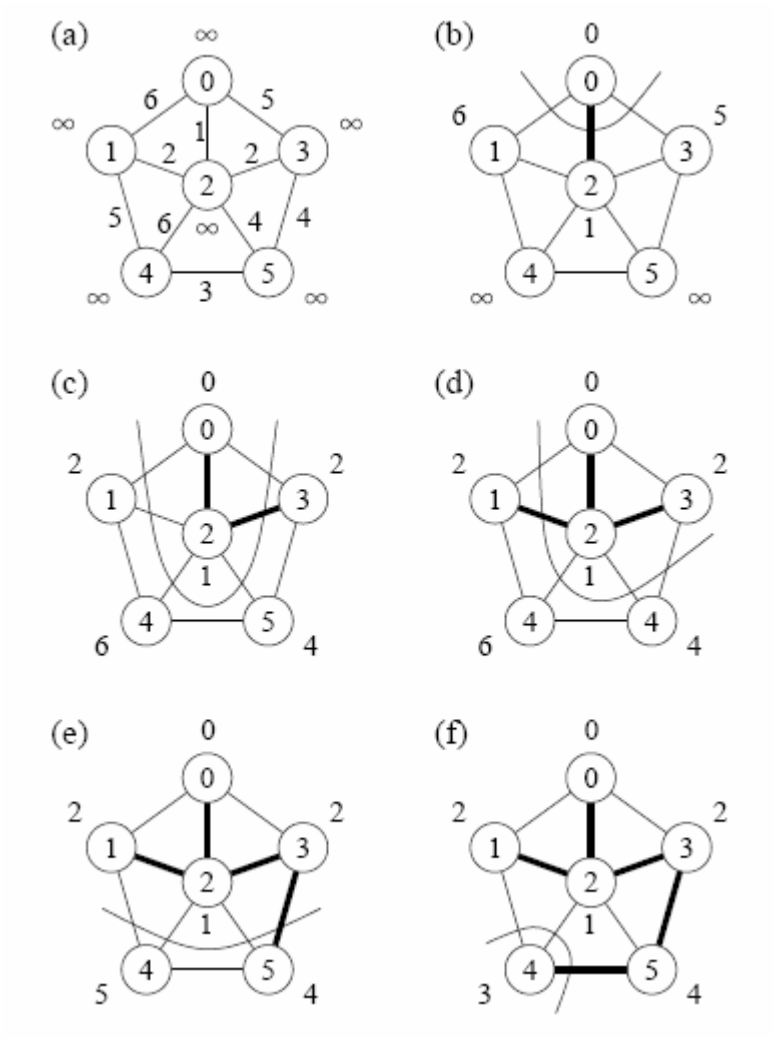
ALGORITMOS DE OPERACIÓN ÁRBOLES

ALGORITMO DE PRIM

El algoritmo de Prim es un algoritmo de la teoría de los grafos para encontrar un árbol de expansión mínimo en un grafo conexo, no dirigido y cuyas aristas están etiquetadas.

En otras palabras, el algoritmo encuentra un subconjunto de aristas que forman un árbol con todos los vértices, donde el peso total de todas las aristas en el árbol es el mínimo posible. Si el grafo no es conexo, entonces el algoritmo encontrará el árbol de expansión mínimo para uno de los componentes conexos que forman dicho grafo no conexo.

El algoritmo de Prim construye un árbol de recubrimiento mínimo sobre un grafo ponderado no dirigido. Se considera el coste del árbol la suma de los pesos de las aristas que lo componen. La construcción del árbol sigue los siguientes pasos.



Algoritmo:

La idea básica consiste en añadir, en cada paso, una arista de peso mínimo a un árbol previamente construido:

1. Empezar en un vértice arbitrario v. El árbol consta inicialmente sólo del nodo v.
2. Del resto de vértices, buscar el que esté más próximo a v, es decir, con la arista (v, w) o (w, v) de coste mínimo. Añadir w y la arista al árbol.
3. Buscar el vértice más próximo a cualquiera de estos dos. Añadir ese vértice y la arista al árbol de expansión.
4. Repetir sucesivamente hasta añadir los n vértices.

SEUDO CÓDIGO

```
Prim(L[1..n,1..n])
entero n  // número de vertices
L[n,n]    //matriz de adyacencia
infinito = 999
T=0
mas_proximo[n]        //matriz de los mas cercanos
```

```
distmin[n]              //matriz de distancia mínima

Para i=2 hasta n
        mas_proximo[i]=1
        distmin[i]=L[i,1]

para i=1 hasta n-1
        min=infinito
        Para j=2 hasta n
                Si 0 <= distmin[j]< min entonces
                        min=distmin[j]
                        k=j
        T=TU{mas_proximo[k],k};
        distmin[k]=-1
        Para j=2 hasta n
                Si L[j,k]<distmin[j]
                        distmin[j]=L[j,k]
                        mas_proximo[j]=k
devolver T
```

ALGORITMO DE KRUSKAL

El objetivo del algoritmo de Kruskal es construir un árbol (subgrafo sin ciclos) formado por arcos sucesivamente seleccionados de mínimo peso a partir de un grafo con pesos en los arcos. Un árbol (spanning tree) de un grafo es un subgrafo que contiene todos sus vértices o nodos. Un grafo puede tener múltiples árboles. Por ejemplo, un grafo completo de cuatro nodos (todos relacionados con todos) tendría 16 árboles.

La aplicación típica de este problema es el diseño de redes telefónicas. Una empresa con diferentes oficinas, trata de trazar líneas de teléfono para conectarlas unas con otras. La compañía telefónica le ofrece esta interconexión, pero ofrece tarifas diferentes o costes por conectar cada par de oficinas. Cómo conectar entonces las oficinas al mínimo coste total.

Dado un grafo G con nodos conectados por arcos con peso (coste o longitud): el peso o coste total de un árbol será la suma de pesos de sus arcos. Obviamente, árboles diferentes tendrán un coste diferente. El problema es entonces ¿cómo encontrar el árbol de coste total mínimo? Una manera de encontrar la solución al problema del árbol de coste total mínimo, es la enumeración completa. Aunque esta forma de resolución es eficaz, no se puede considerar un algoritmo, y además no es nada eficiente.

```
Kruskal (G)
E(1)=0, E(2)= todos los Arcos del grafo G
    Mientras E(1) contenga menos de n-1 arcos y E(2)=0 do
      De los arcos de E(2) seleccionar el de menor coste -->e(ij)
```

```
E(2)= E(2) - {e(ij)}
    Si V(i), V(j) no están en el mismo árbol entonces
      juntar los árboles
```

ALGORITMO DE BELLMAN-FORD (CAMINO MÍNIMO)

Soluciona el problema de la ruta más corta o camino mínimo desde un nodo origen, de un modo más general que el Algoritmo de Dijkstra, ya que permite valores negativos en los arcos. El algoritmo devuelve un valor booleano si encuentra un circuito o lazo de peso negativo. En caso contrario calcula y devuelve el camino mínimo con su coste. Para cada vértice v perteneciente a V, se mantiene el atributo d[v] como cota superior o coste del camino mínimo desde el origen s al vértice v.

```
Bellman-Ford (G,s)

Inicializar
   for cada v perteneciente a V[G]
      do d[v] = infinito
         p[v] = nulo
   p[s] = 0

for i=1 to V[G]-1
   do for cada arco (u,v) perteneciente a A[G]
      Relajación
      if d[v] > d[u] + w(u,v) then
         d[v] = d[u] + w(u,v)
         p(v) = u

for cada arco (u,v) chequea lazo de peso negativo
   do if d[v] > d[u] + w(u,v) then
      return FALSO 'el algoritmo no converge
return VERDADERO
```

ALGORITMO DE BELLMAN-FORD (CAMINO MÁXIMO)

El problema de la ruta más larga puede ser transformado en el de ruta más corta cambiando el signo de los costes de los arcos. De manera alternativa se puede transformar también cambiando los procesos de inicialización y relajación. En este caso el problema es inconsistente para circuitos de peso positivo.

```
Inicializar
   for cada v perteneciente a V[G]
      do d[v] = - infinito
         p[v] = nulo
   p[s] = 0
Relajación
      if d[v] < d[u] + w(u,v) then
         d[v] = d[u] + w(u,v)
```

```
        p(v) = u
```

ALGORITMO DE DIJKSTRA (RUTA MÁS CORTA - ÁRBOL MÍNIMO - CAMINO MÍNIMO)

```
:Dijkstra (G,s)
Inicializar
   for cada v perteneciente a V[G]
      do d[v] = infinito
         p[v] = nulo
   d[s] = 0
S =  vacio
Q = V[G]
mientras Q no vacio

do u = nodo v con min d[v]

S = S unión u 'se añade al conjunto de nodos finalizados
for cada v perteneciente Adyacente u
   Relajación
      if d[v] > d[u] + w(u,v) then
         d[v] = d[u] + w(u,v)
         p(v) = u
```

CAPÍTULO VII

MAQUINAS DE ESTADO FINITO

Un autómata finito o máquina de estado finito es un modelo matemático de un sistema que recibe una cadena constituida por símbolos de un alfabeto y determina si esa cadena pertenece al lenguaje que el autómata reconoce.

Las máquinas de estado finito son una herramienta muy útil para especificar aspectos relacionados con tiempo real, dominios reactivos o autónomos, computación reactiva, protocolos, circuitos, arquitecturas de software, etc. El modelo de FSM (Finite State Machine) es un modelo que posee sintaxis y semántica formales y que sirve para representar aspectos dinámicos que no se expresan en otros diagramas.

Los nodos representan los posibles estados de aquello que se desea modelar. Las etiquetas representan eventos que provocan un cambio. Las aristas determinan de qué manera cada estado, dado un evento, deriva en otro estado.

APLICACIONES EN INTELIGENCIA ARTIFICIAL

Las maquinas de estado finito tienen un gran futuro en la aplicación en inteligencia artificial. En la inteligencia artificial su principal objetivo es la creación de un agente inteligente que sea capaz de actuar y razonar como un humano. Para la creación de este agente es necesario contar con un total conocimiento de la gramática y los lenguajes formales que se requiere que el agente utilice, para esto debe contar un una maquina que sea capaz de aceptar los símbolos y reconocer las cadenas que se están usando.

APLICACIONES EN ELECTRONICA

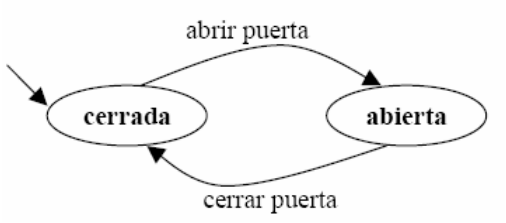
Las maquinas de estado finito son importantes en la aplicación en la electrónica, son aceptadores de símbolos las aplicaciones mas importantes que tenemos en la electrónica es en la creación de los circuitos, ya que tiene que reconocer que una cadena de símbolos se idéntica a otra y así dejar pasar los datos.

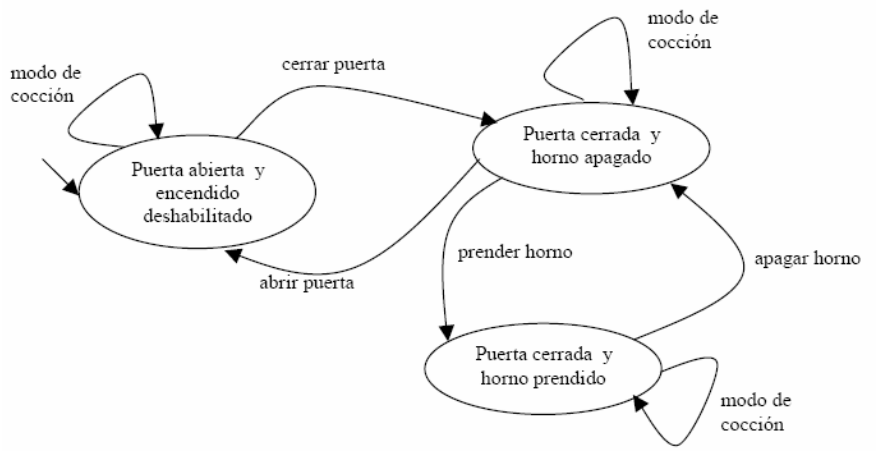
Este tipo de maquinas los podemos encontrar como lo es en los semáforos, en los circuitos integrados, en los apagadores automáticos de luz, en los sistemas automáticos de riego, ect.

Ejemplo

Supongamos que se quiere modelar el comportamiento de una puerta. La puerta, inicialmente cerrada, puede pasar a estar abierta tras el evento "abrir puerta". Una vez abierta, puede pasar al estado cerrada, tras el evento "cerrar puerta".

Ejemplo





Definición.- Un autómata finito (AF) puede ser descrito como una 5-tupla (S,Σ,T,s,A) donde:

- S un conjunto de estados;
- Σ es un conjunto de entradas
- T es la función de transición:
- $s \in S$ es el estado inicial;
- $A \subseteq S$ es un conjunto de estados de aceptación o finales.

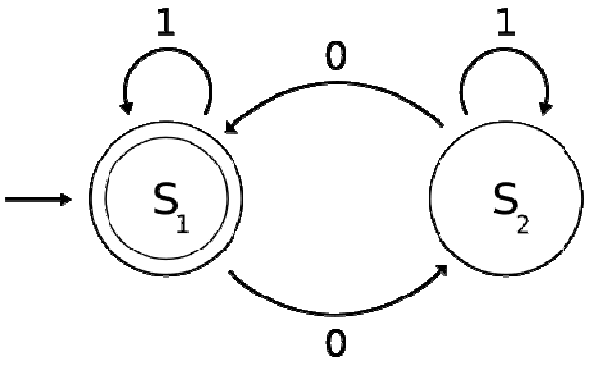
Ejemplo

- $S = \{S_1, S_2\}$,
- $\Sigma = \{0,1\}$,
- $T = \{(S_1,0,\{S_2\});(S_1,1,\{S_1\});(S_2,0,\{S_1\});(S_2,1,\{S_2\})\}$
- $s = S_1$
- $A = \{S_1\}$.

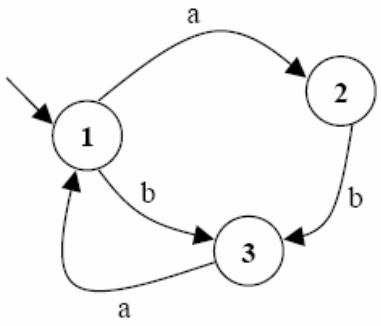
Construya la tabla de transición y el diagrama o dígrafo de transición

Tabla de Transición de Estados		
Entrada Estado	**1**	**0**
S_1	S_1	S_2
S_2	S_2	S_1

Diagrama de estados



Ejemplo



- $\langle S = \{1,2,3\}$,
- $\Sigma = \{a,b\}$,
- $\mathbf{T} = \{(1,a,2),(2,b,3),(3,a,1),(1,b,3)\}$
- $\mathbf{s} = 1 \rangle$

Las trazas de esta FSM son:

- {a, b, a} correspondiente a 1, 2, 3, 1
- {b, a} correspondiente a 1, 3, 1
- {a, b, a, b, a} correspondiente a 1, 2, 3, 1, 3, 1
- {b, a, a, b, a} correspondiente a 1, 3, 1, 2, 3, 1
- {b, a, b, a, ... , b, a} 1, 3, 1, 3, …, 1, 3
- Etc…

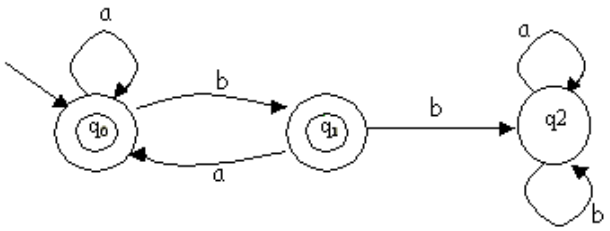
Por lo general, en los autómatas de estado finito, los estados de aceptación se dibujan con círculos dobles

Ejemplo

Muestre que la máquina de estado finito donde T función de estado esta dado por la tabla

	a	b
q0	q0	q1
q1	q0	q2
q2	q2	q2

Solución



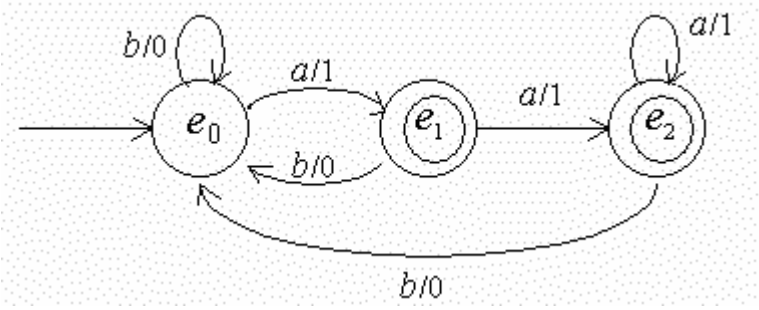
Ejemplo

Teniendo tablas de transición de estados y de salida realizar el diagrama.

	F				*g*	
	a	*b*			*a*	*b*
e_0	E_1	e_0		e_0	1	0
e_1	E_2	e_0		e_1	1	0
e_2	E_2	e_0		e_2	1	0

Solución.

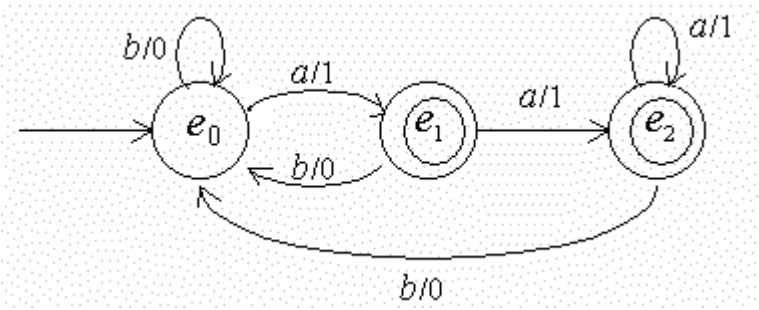
La gráfica de la máquina es la siguiente:



Definición. Sea a = $X_1X_2...X_n$ una cadena de entrada a un autómata de estado finito. Se dice que *a* es aceptada, si el estado final de la cadena termina en un estado de aceptación.

Ejemplo

Es aceptada la cadena *abaa* por el autómata de estado finito descrito por la gráfica siguiente?



Solución

- Comenzando en el estado e_0, se tiene que cuando entra *a*, se pasa a e_1.
- Estando en e1, si la entrada es b, pasamos al estado e0.
- Estando en e0, si la entrada es a, pasamos al estado e1.
- Por último, estando en e_1, si la entrada es *a* pasamos al estado e_s que es un estado de aceptación.

Por lo tanto, a=abaa es aceptada por el autómata de estado finito.

Ejemplo

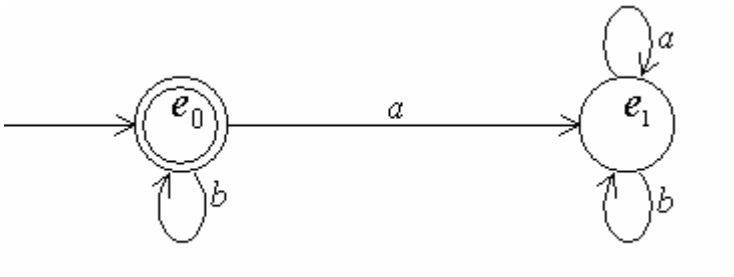
Diseñe un autómata de estado finito que acepte aquellas cadenas del conjunto A = { *a, b* } que no tengan letras *a.*

Solución.

Consideremos dos estados.
eo: No se encontró una *a.*
e_1: Se encontró una *a*.

	F			*g*	
	A	*b*		*a*	*b*
e_0	E_1	e_0	e_0	0	1
e_1	E_1	e_1	e_1	0	0

La gráfica será:



Ejemplo

Diseñe un autómata de estado finito que acepte aquellas cadenas del conjunto A = { *a, b* } que contienen un número impar de letras *a*.

Solución.

Se consideran dos estados:

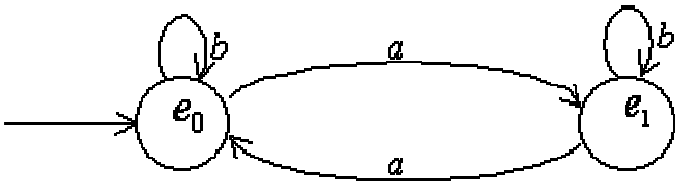
e_0: Hay un número par de letras *a.*

e_1: Hay un número impar de letras *a*.

es claro que el estado de aceptación es e_1.

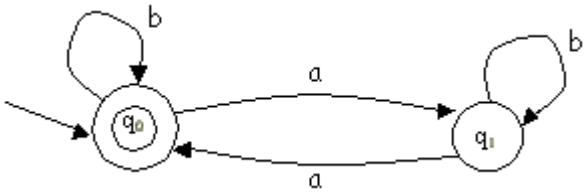
	f				*G*	
	a	*b*			*A*	*b*
E_0	e_1	e_0		e_0	1	1
E_1	e_0	e_1		e_1	0	0

La gráfica será:



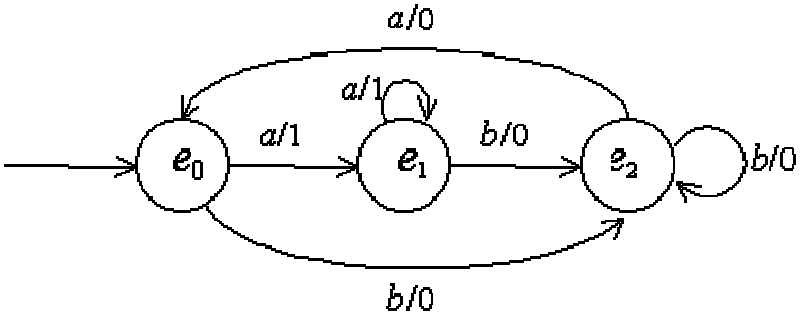
Ejemplo

Sean a y b los símbolos de entrada, construya un autómata finito M que acepte precisamente aquellas cadenas en a y b que tienen un numero par de a's.



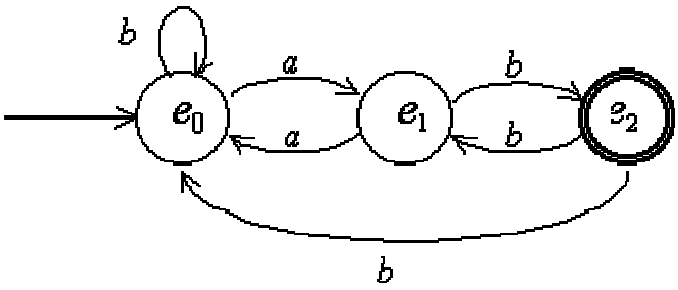
Ejercicios

1. Muestre que la máquina de estado finito descrita por la siguiente gráfica es un autómata de estado finito.



Cuál es el estado de aceptación?

2. Dada la gráfica del autómata de estado finito



   Dibuje la tabla de transición de estados y la tabla de salida.

3. Determine si la cadena *abbaa* es aceptada por los autómatas de los ejercicios 1 y 2.
4. Realice el diagrama correspondiente a partir de los siguientes datos.

   Símbolo de entrada={a, b, c},

   Estados {q0, q1, q2, q3}

   Estados de aceptación {q0,q1}

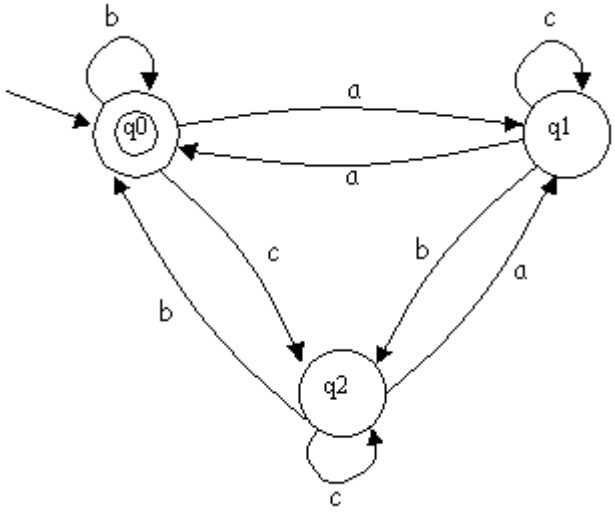
   Estado inicial q0

   La función de estado definida por la siguiente tabla:

	a	b	c
q0	q1	q3	q2
q1	q1	q3	q0
q2	q3	q0	q1
q3		q2	q1

5. A partir del siguiente diagrama determine:
   - Los símbolos de entrada.
   - Tabla de defunción de estado y de salida
   - Los estado
   - Los estados de aceptación.

- El estado inicial.



6. Construya un AF M con símbolos de entrada a y b que acepte solamente aquellas cadenas con a y b tales que el numero de b's es divisible por 3.
7. Sean a y b los símbolos de entrada, construya un autómata finito M que acepte precisamente aquellas cadenas en las que aparezcan a3 y b4.
8. Construya un AFM con símbolos de entrada a y b que acepte solamente aquellas cadenas con a y b tales que aabb aparezca como una sub-cadena. Ejemplo: baaabbba, babaabba serán aceptables, pero babbaa y aababaa no lo serán.

CAPÍTULO VIII

LENGUAJES FORMALES Y LENGUAJES NATURALES

Los lenguajes naturales y formales tienen puntos en común que nos pueden servir de inicio para una discusión. En principio se tiene la existencia de un conjunto finito llamado alfabeto, el cual esta constituido de símbolos simples llamados comúnmente letras. En los lenguajes naturales se tienen los alfabetos. En los formales la lógica, cálculo proposicional y de predicados. Mediante la concatenación de las letras del alfabeto formaremos: monemas, fonemas o palabras que determinan un conjunto extendido. El conjunto de palabras que tengan un significado constituirán el diccionario del lenguaje y, en lenguajes formales todas las palabras que puedan ser aceptadas por un cierto autómata.

A partir de lo anterior, tendremos que un lenguaje se considera como un conjunto, usualmente es infinito, de oraciones o enunciados que se forman con palabras del diccionario. En este punto, podemos distinguir entre dos clases de lenguajes; los ``*lenguajes naturales*'' como el francés, inglés, y castellano y, los ``*lenguajes formales*'' como el de las matemáticas y la lógica.

En resumen podemos decir que los lenguajes naturales y formales, difieren significativamente uno de otro por su origen y por su rea de aplicación.

LENGUAJES FORMALES

En matemáticas, lógica, y ciencias de la computación, un lenguaje formal es un conjunto de palabras (cadenas de caracteres) de longitud finita formadas a partir de un alfabeto (conjunto de caracteres) finito. El nombre lenguaje se justifica porque las estructuras que con este se forman tienen reglas de buena formación (gramática) e interpretación semántica (significado) en una forma muy similar a los lenguajes hablados.

Ejemplos de lenguajes formales

- El conjunto de todas las palabras sobre $\{a, b\}$.
- El conjunto $\{a^n : n$ es un número primo $\}$.
- El conjunto de todos los programas sintácticamente válidos en un determinado lenguaje de programación.

- El conjunto de entradas para las cuales una máquina de Turing concreta se detiene.
- El conjunto de sentencias bien formadas en lógica de predicados.

Especificación de lenguajes formales

Los lenguajes formales se pueden especificar de una amplia variedad de formas, como por ejemplo:

- Cadenas producidas por una gramática formal (véase Jerarquía de Chomsky).
- Cadenas producidas por una expresión regular.
- Cadenas aceptadas por un autómata, tal como una máquina de Turing.

Propiedades de los lenguajes formales

A la definición (es decir, axiomática) de una teoría a la cual se le asocia un lenguaje formal dado, la formación de oraciones (fórmulas) de este lenguaje; dando como consecuencia que el proceso de generación y desarrollo de un lenguaje formal es inverso al de los lenguajes naturales. En un lenguaje formal, las palabras y las oraciones están perfectamente definidas, una palabra mantiene el mismo significado prescindiendo del contexto o su uso. En principio, el significado de símbolos es determinado exclusivamente por la sintaxis, sin referencia a ningún contenido semántico, una función y una fórmula puede designar cualquier cosa, sin embargo:

- Las relaciones como: $=, \leq, \simeq, \in, \notin$, etc..
- Los conectivos lógicos como: $\forall, \wedge, \rightarrow$, etc..
- Los operadores algebraicos como: $\oplus, \times, \sqrt{}$, etc..

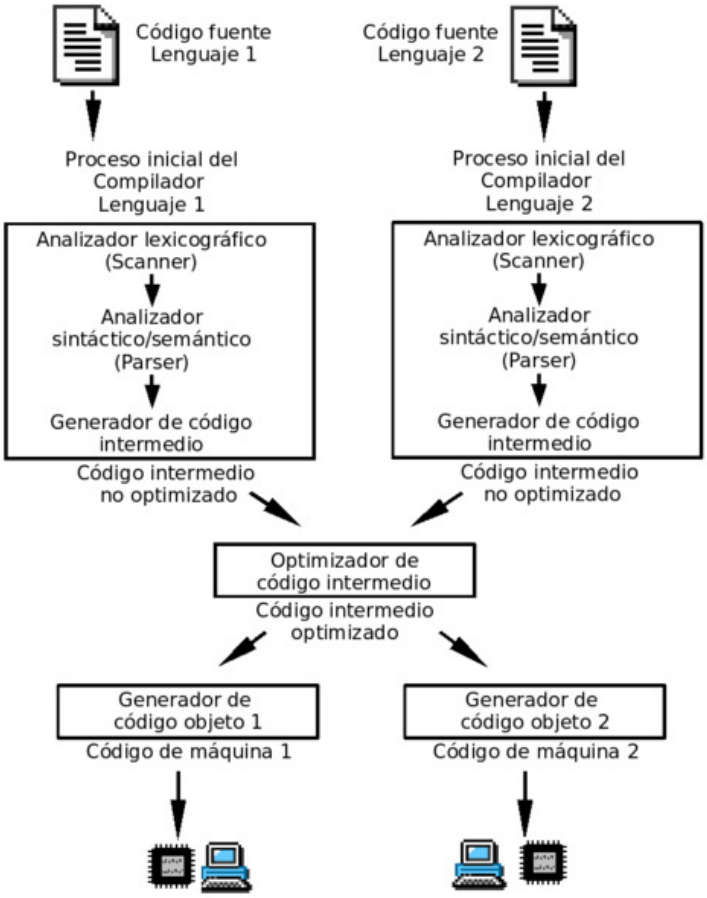
Resumiendo los lenguajes formales son caracterizados por las siguientes propiedades:

- Se desarrollan de una teoría preestablecida,
- Componente semántico mínimo.
- Posibilidad de incrementar el componente semántico de acuerdo con la teoría a formalizar.
- La sintaxis produce oraciones no ambiguas.
- La importancia del rol de los números.
- Completa formalización, caracterizadas por su construcción computacional

COMPILADOR

Un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es código máquina, pero también puede ser simplemente texto. Este proceso de traducción se conoce como compilación.

Un compilador es un programa que permite traducir el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior (típicamente lenguaje máquina). De esta manera un programador puede diseñar un programa en un lenguaje mucho más cercano a como piensa un ser humano, para luego compilarlo a un programa más manejable por una computadora.



LENGUAJE NATURAL

Es el lenguaje hablado y/o escrito por humanos para propósitos generales de comunicación, para distinguirlo de otros como puedan ser una lengua construida, los lenguajes de programación o los lenguajes usados en el estudio de la lógica formal, especialmente la lógica matemática.

El término lenguaje natural se refiere al estudio de las propiedades computacionales y de otro tipo implicadas en la comprensión, producción y uso de las lenguas naturales.

Propiedades de los lenguajes naturales

El origen de los lenguajes naturales sucede de manera oral hace unos 100,000 años aproximadamente. Las primeras formas de expresión gráfica no son por cierto tan antiguas, pero, se las puede hacer remontar a las etapas del periodo paleolítico cuando comienza a verse el empleo de sistemas gráficos. Esta función llevada a cabo, primeramente, por medio de señales y vocales (voz), culmina posteriormente con una escritura por signos escritos, para conformar en su totalidad el lenguaje natural.

Con respecto a nuestro mundo, el lenguaje nos permite designar las cosas reales y razonar acerca de ellas, así como también crear significados. Contrariamente a lo que ciertas teorías lingüísticas formales harían a uno creer, el lenguaje natural no fue fundamentado sobre una verdad racional a priori, sino que fue desarrollado y organizado a partir de la experiencia humana, en el mismo proceso en que la experiencia humana fue organizada. En su forma actual, los lenguajes naturales tienen un gran poder expresivo y pueden ser utilizados para analizar situaciones altamente complejas y llevarnos por razonamientos muy sutilmente.

Resumiendo los lenguajes naturales se distinguen por las siguientes propiedades:

- Desarrollados por enriquecimiento progresivo antes de cualquier intento de formación de una teoría.
- La importancia de su carácter expresivo debido grandemente a la riqueza de el componente semántico
- Dificultad o imposibilidad de una formalización completa.

PROCESAMIENTO DE LENGUAJE NATURAL (PLN)

Es una rama de la Inteligencia Artificial, que se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas o entre personas y máquinas por medio de programas que ejecuten o simulen la comunicación. Los modelos aplicados

se enfocan no sólo a la comprensión del lenguaje, sino a aspectos generales cognitivos humanos y a la organización de la memoria. El lenguaje natural sirve sólo de medio para estudiar estos fenómenos.

Las aplicaciones de Procesamiento de Lenguaje natural son: Síntesis del discurso, Análisis del lenguaje, Comprensión del lenguaje, Reconocimiento del habla, Síntesis de voz, Generación de lenguajes naturales, Traducción automática, Recuperación de la información, Dictado Automático ([1]). Teniendo múltiples aplicaciones el Procesamiento del Lenguaje Natural contempla elementos como: Análisis morfológico, análisis sintáctico, análisis semántico y análisis pragmático.

Los Procesos o componentes Lenguaje Natural

- Análisis morfológico. El análisis de las palabras para extraer raíces, rasgos flexivos, unidades léxicas compuestas y otros fenómenos. Ejm:
  Niñ + o = niño
  Niñ + a = niña
  Niñ + os = niños
  Niñ + as = niñas
  Niñ + ito = niñito
  Niñ + ita = niñita
- Análisis sintáctico. El análisis de la estructura sintáctica de la frase mediante una gramática de la lengua en cuestión.
- Análisis semántico. La extracción del significado de la frase, y la resolución de ambigüedades léxicas y estructurales.
- Análisis pragmático. Es el paso final hacia una comprensión eficaz consiste en decidir qué hacer como resultado. Una posibilidad es almacenar lo dicho como un hecho y hacerlo. Para algunas oraciones, en donde el efecto deseado es claramente declarativo, ésta precisamente la situación correcta, Ejemplo el hecho de que cuando el usuario reclama querer hacer algo que el sistema es capaz de realizar, entonces el sistema debería hacerlo directamente.

Aplicaciones

Las principales aplicaciones de trabajo en el PLN son:

- Análisis del lenguaje
- Comprensión del lenguaje
- Reconocimiento del habla

---

[1] Nilsson Nils “Inteligencia Artificial” Primera Edición Pág. 344.

- Síntesis de voz
- Generación de lenguajes naturales
- Traducción automática
- Recuperación de la información
- Extracción de la información

Lingüística Computacional

La lingüística computacional es un campo multidisciplinar de la lingüística y la informática que utiliza la informática para estudiar y tratar el lenguaje humano. Para lograrlo, intenta modelar de forma lógica el lenguaje natural desde un punto de vista computacional. Dicho modelado no se centra en ninguna de las áreas de la lingüística en particular, sino que es un campo interdisciplinar, en el que participan lingüistas, informáticos especializados en inteligencia artificial, psicólogos cognoscitivos y expertos en lógica, entre otros.

Algunas de las áreas de estudio de la lingüística computacional son: Corpus lingüístico asistido por ordenador, Diseño de analizadores sintácticos (en inglés: parser), para lenguajes naturales, Diseño de etiquetadores o lematizadores (en inglés: tagger), tales como el POS-tagger, Definición de lógicas especializadas que sirvan como fuente para el Procesamiento de Lenguajes Naturales, y Traducción automática.

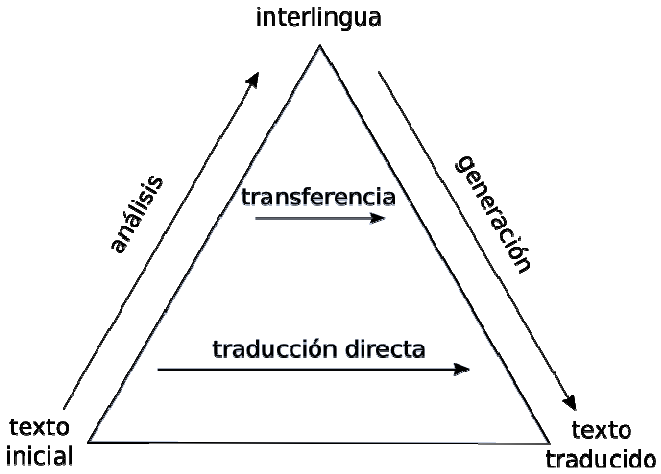
TRADUCTOR AUTOMÁTICO

Es una aplicación de Procesamiento de Lenguaje Natural, también considerada como área de la lingüística computacional que investiga el uso de software para traducir texto o habla de un lenguaje natural a otro. En un nivel básico, la traducción por computadora realiza una substitución simple de las palabras atómicas de un lenguaje natural por las de otro. Por medio del uso de corpus lingüísticos se pueden intentar traducciones más complejas, lo que permite un manejo más apropiado de las diferencias en la tipología lingüística, el reconocimiento de frases, la traducción de expresiones idiomáticas y el aislamiento de anomalías.

El traductor automático debe analizar el texto original, interrelacionar con la situación referida y como resultado debe encontrar el texto correspondiente en el lenguaje destino. ([2]). Los tipos de traducción automática son: Traducción automática basada en reglas, Traducción automática basada en corpus lingüístico y la traducción automática basado en contexto.

[2] Stuart Rusell y Meter Norvig, "Inteligencia Artificial un enfoque moderno", Segunda Edición, Editorial. Pearson Educación S.A. Madrid 2004 , Pág. 965.

La traducción automática basada en reglas, asume varios grados en su fundamento: a) traducción directa o por diccionario como modelo diccionarios bilingües. La traducción de un texto se obtiene a partir de la traducción palabra por palabra, sin tener en cuenta ni la relación entre ellas ni el contexto en que se encuentran; b) Traducción automática mediante transferencia en donde el análisis del texto original juega un papel más importante, y da paso a una representación interna que es la que se utiliza como enlace para traducir entre idiomas distintos finalmente recomponiendo para el idioma meta; y c) Traducción por Lenguaje Intermedio conocida también Traducción Automática Mediante Lengua Intermedia (interlingua), en donde el lenguaje original, por ejemplo un texto que debe ser traducido, es transformado a un lenguaje intermedio, cuya estructura es independiente a la del lenguaje original y a la del lenguaje final. El texto en el lenguaje final se obtiene a partir de la representación del texto en el lenguaje intermedio.



Traducción automática basada en corpus lingüísticos se basa en el análisis de muestras reales con sus respectivas traducciones, entre los mecanismos que utilizan corpus se incluyen los métodos estadísticos y los basados en ejemplo; finalmente *la traducción automática basado en contexto*, utiliza técnicas para hallar la mejor traducción, para una palabra fijándose en el resto de palabras que la rodean, básicamente este método se esmera en tratar el texto en unidades de entre 4 y 8 palabras, de manera que se traduce cada una de ellas por su traducción al idioma destino y se eliminan las traducciones que han generado una "frase" sin sentido.

Traductor automático por transferencia

En un sistema de traducción automático basado en el modelo de transferencia, el texto original se analiza primero morfológica y sintácticamente, obteniendo como resultado una representación sintáctica superficial. Esta representación se transforma a continuación en otra más abstracta que

hace especial énfasis en aspectos relevantes para el proceso de traducción e ignora otro tipo de información. El proceso de transferencia convierte esta última representación (ligada aún al idioma original) a una representación al mismo nivel de abstracción pero ligada al lenguaje objetivo. Estas dos representaciones son las llamadas normalizadas o intermedias. A partir de aquí el proceso se invierte: los componentes sintácticos generan una representación del texto y finalmente se genera la traducción en la lengua meta.

Análisis y transformación

Se pueden utilizar distintos mecanismos antes de llegar al resultado final. Dependiendo del diseño del traductor se hace énfasis en uno u otro, e incluso se pueden llegar a combinar con análisis estadísticos, formando auténticos híbridos. Pero, en los traductores por transferencia más clásicos, normalmente se pueden encontrar los siguientes etapas:

- Análisis morfológico. Consiste en identificar los elementos del texto y clasificarlos en función de lo que son: nombres, verbos, adjetivos, etc. Además, también deben reconocerse abreviaturas y otras expresiones o palabras compuestas.
- Categorización léxica. Algunas de las palabras que aparecen en un texto pueden tener más de un significado, causando así ambigüedad a la hora de hacer su análisis. La categorización léxica analiza el contexto, es decir, los elementos vecinos al actual, y escoge el significado que mejor encaja.
- Transferencia léxica. La transferencia léxica es equivalente a lo que comúnmente se conoce como traducción por diccionario. A partir de la forma léxica de la palabra original, se trata de derivar su equivalente en el nuevo idioma.
- Transferencia estructural. Una transferencia estructural analiza el texto desde un punto de vista más amplio. En vez de centrarse en palabras, amplia su objetivo a fragmentos mayores. De esta manera se pueden encontrar expresiones que puedan requerir un tratamiento especial (p.ej. refranes, dichos, etc.).
- Generador morfológico. En el momento en que ya se ha realizado el análisis, el generador morfológico es el encargado de asociar cada elemento identificado en fases anteriores con su equivalente en la lengua objetivo, procurando que la interpretación sea lo más fiel posible

Tipos de transferencia

La principal característica de los sistemas de transferencia es la existencia de una fase que proyecta representaciones intermedias del texto original sobre representaciones del texto objetivo. Éste componente puede trabajar en distintos niveles de análisis lingüístico, por lo que se pueden distinguir dos tipos de transferencia: Transferencia superficial y Transferencia profunda.

Transferencia superficial (sintáctica)

Se caracteriza por hacer un análisis sintáctico mediante el que se transfieren las estructuras sintácticas del lenguaje origen a las estructuras sintácticas del lenguaje objetivo. Este tipo de transferencia resulta muy apropiado para traducciones entre idiomas de una misma rama (p.ej. entre lenguas romance como el castellano, el catalán, el francés, el italiano, o el portugués).

Transferencia profunda (semántica)

Construye una representación semántica que es dependiente del lenguaje original. Esta representación puede consistir en una serie de estructuras que representen el significado. En estos sistemas la transferencia se realiza principalmente sobre predicados. La traducción de palabras normalmente también requiere una transferencia estructural previa. Este tipo es más común entre idiomas de ramas diferentes (p.ej. castellano-inglés, castellano-euskera, etc.).

.

GRAMÁTICAS

Una gramática formal es objeto o modelo matemático que permite especificar un lenguaje o lengua, es decir, es el conjunto de reglas capaces de generar todas las posibilidades combinatorias de ese lenguaje, ya sea éste un lenguaje formal o un lenguaje natural.

La expresión «gramática formal» tiene dos sentidos:

- Gramática de un *lenguaje formal*.
- *Descripción formal* de parte de la gramática de un lenguaje natural.

Cuando nos referimos a lenguaje natural estas reglas combinatorias reciben el nombre de sintaxis, y son inconscientes. Hay distintos tipos de gramáticas formales que generan lenguajes formales (Jerarquía de Chomsky).

Imaginemos una gramática con estas dos reglas:

1. $A \rightarrow bAc$ **2.** $A \rightarrow de$

La idea es sustituir el símbolo inicial de la izquierda por otros símbolos aplicando las reglas. El lenguaje al cual representa esta gramática es el conjunto de cadenas de símbolos que pueden ser generados de esta manera: en este caso, por ejemplo:

A → bAc → bbAcc → bbbAccc → bbbdeccc.

El elemento en mayúsculas es el símbolo inicial. Los elementos en minúsculas son símbolos terminales. Las cadenas de la lengua son aquellas que solo contienen elementos terminales, como por ejemplo: bbbdeccc, de, bdec, ... Estas serían tres posibles realizaciones del lenguaje cuya gramática hemos definido con dos reglas.

Para comprender mejor el concepto pondremos algunas reglas de la gramática castellana:

- Una FRASE se puede componer de SUJETO + PREDICADO
  O = SN + SV
- Un SUJETO se puede componer de un ARTÍCULO + NOMBRE o SUSTANTIVO (núcleo)
- Un PREDICADO se puede componer de un VERBO conjugado
- Un ARTICULO puede ser la palabra "el"
- Un NOMBRE o SUBSTANTIVO puede ser "niño"

Definición

Una gramática para estructura de expresiones G es una 4-ada **(V,S,vo,→)**

Donde:

- V es {S U N} un conjunto finito (alfabeto de símbolos no terminales llamadas variables)
  N es V-S es el conjunto de símbolos no terminales
- S es un subconjunto de V (alfabeto de símbolos terminales llamadas constantes) debe cumplir que V ∩S = ∅
- Vo Є V es el símbolo inicial o axioma de la gramática
- → es una relación finita en V* (conjunto de reglas de producción de la gramática)

La idea es que S es el conjunto de todas las "palabras" permitidas en el lenguaje, y V consta de S además de algunos otros símbolos. El elemento vo de V es un punto de partida para las sustituciones. Por último, la relación → sobre V* especifica los reemplazos permisibles, en el sentido de que, si w→w' se puede reemplazar w con w' son los lados izquierdo y derecho de la producción respectivamente se llama producción de G. entonces w y w'

Ejemplo

Sea:

- V = {S U N }
- S = {Juan, julia, maneja, corre, descuidadamente, rápido, frecuentemente}

N = {oración, sujeto, predicado, verbo, adverbio}

- Vo = Oración
- → = en V* queda descrita enumerando todas las producciones como sigue:
  - Oración → sujeto + predicado
  - Sujeto → Juan
  - Sujeto → Julia
  - Predicado → verbo + adverbio
  - Verbo → maneja
  - Verbo → corre
  - Adverbio → descuidadamente
  - Adverbio → rápido
  - Adverbio → frecuentemente

El conjunto S contiene todas las palabras permitidas en el lenguaje; N consta de las palabras que describen partes de la oración, pero que en realidad no están contenidas en el lenguaje. Se afirma que la oración "Julia maneja frecuentemente", que será denotada por w, es una oración permisible o con sintaxis correcta, de acuerdo con las reglas de este lenguaje. Para mostrar esto, considere la siguiente serie de cadenas.

Oración
Sujeto + predicado
Julia + predicado
Julia + verbo + adverbio
Julia + maneja + adverbio
Julia + maneja + frecuentemente

Cada una de las cadenas es consecuencia de la anterior, utilizando una producción para realizar una sustitución parcial o completa. En otras palabras, acada cadena está relacionada con la siguiente cadena por la relación → de modo que la oración $\rightarrow^{\infty}$ w. Por definición w tiene una sintaxis correcta ya que, en este ejemplo, vo es una oración.

BIBLIOGRAFIA

1. **BERNARD KOLMAN, ROBERT C. BYSBY, SHARON ROSS**. Estructuras de matemáticas discretas para la computación. Tercera Edición.
2. **KOLMAN**, Bernard (1986) Estructuras de matemáticas discretas para la computación. Editorial Prentice Hall Hispanoamericana, S.A. México.
3. **KENNETH**, **ROSS**. Matemáticas discretas. Editorial Prentice Hall Hispanoamericana. S.A. México.
4. **JOHNSONBAUGH**, **RICHARD**. Matemáticas discretas. Editorial. Iberoamericana, S.A. México.
5. **GRIMALDI, RALPH,** Matemáticas discretas y combinatorio. Edit. Addison Wesley Longman México.
6. **ROJO, Armando** (2002) Algebra I. Editorial Ateneo. Argentina
7. **JOHNSONBAUGH**, Richard (1986) Matemáticas discretas. Editorial. Iberoamericana, S.A. México.
8. **NILS, Nilsson**. Inteligencia Artificial. Madrid. McGraw Madrid. Hill/Interamericana S.A. 2004.
9. **RUSELL, Stuart y NORVIG**, Meter. Inteligencia Artificial un enfoque moderno. Segunda Edición. Madrid. Pearson Educación S.A. 2004.