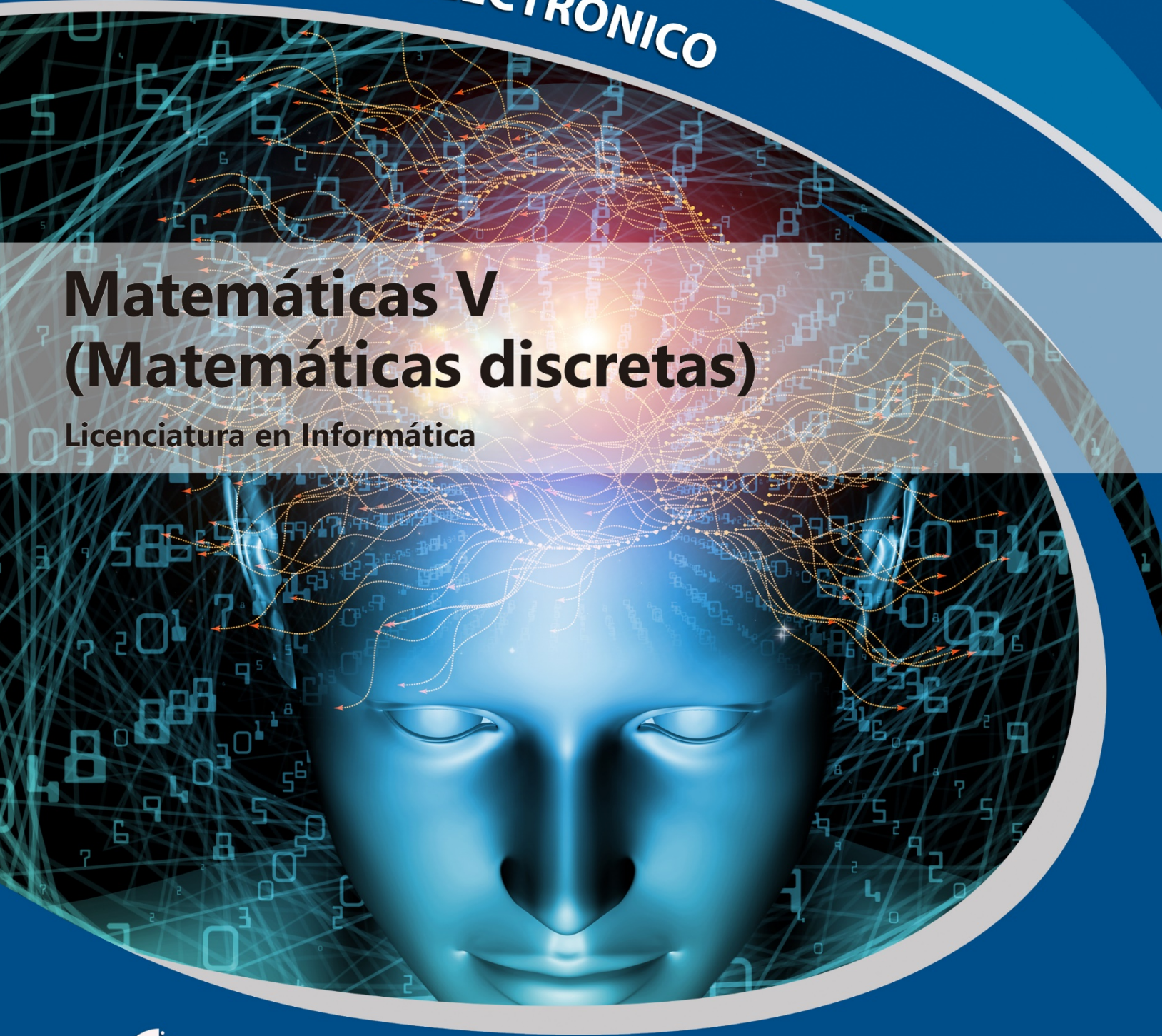
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
DIVISIÓN SISTEMA UNIVERSIDAD ABIERTA Y EDUCACIÓN A DISTANCIA

APUNTE ELECTRÓNICO

# Matemáticas V (Matemáticas discretas)

## Licenciatura en Informática



SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

# COLABORADORES

**DIRECTOR DE LA FCA**
Mtro. Tomás Humberto Rubio Pérez

**SECRETARIO GENERAL**
Dr. Armando Tomé González

– – – –

**COORDINACIÓN GENERAL**
Mtra. Gabriela Montero Montiel
Jefa del Centro de Educación a Distancia y Gestión del Conocimiento

**COORDINACIÓN ACADÉMICA**
Mtro. Francisco Hernández Mendoza
FCA-UNAM

– – – –

**AUTOR**
Mtro. Rene Montesano Brand

**REVISIÓN PEDAGÓGICA**
Mtra. Dayanira Granados Pérez

**CORRECCIÓN DE ESTILO**
Mtro. Carlos Rodolfo Rodríguez de Alba

**DISEÑO DE PORTADAS**
L.CG. Ricardo Alberto Báez Caballero
Mtra. Marlene Olga Ramírez Chavero

**DISEÑO EDITORIAL**
Mtra. Marlene Olga Ramírez Chavero

**Dr. Enrique Luis Graue Wiechers**
Rector

**Dr. Leonardo Lomelí Vanegas**
Secretario General

**Mtro. Tomás Humberto Rubio Pérez**
Director

**Dr. Armando Tomé González**
Secretario General

**Mtra. Gabriela Montero Montiel**
Jefa del Centro de Educación a Distancia
y Gestión del Conocimiento / FCA

# Matemáticas V (Matemáticas Discretas)
# Apunte electrónico

Edición: noviembre 2017



Facultad de Contaduría y Administración
Circuito Exterior s/n, Ciudad Universitaria
Delegación Coyoacán, C.P. 04510, México, Ciudad de México.

ISBN: En trámite
Plan de estudios 2012, actualizado 2016.



Hecho en México

# OBJETIVO GENERAL

El alumno conocerá y aplicará las diferentes herramientas correspondientes a las matemáticas discretas aplicadas en el desarrollo de la informática.

# TEMARIO OFICIAL
## (64 horas)

| | Horas |
|---|---|
| 1. Introducción. Unificación de conceptos | 6 |
| 2. Análisis de algoritmos | 12 |
| 3. Relaciones | 10 |
| 4. Teoría de grafos | 14 |
| 5. Árboles | 12 |
| 6. Prácticas en laboratorio | 10 |
| **Total** | **64** |

# INTRODUCCIÓN

Las matemáticas discretas, o más bien, las estructuras de matemáticas discretas, son abstracciones o modelos de representación matemática de la realidad, y en el caso de nuestra asignatura, de la representación de las relaciones entre cosas o personas.

A lo largo de la asignatura revisaremos conceptos que posiblemente ya abordaste con anterioridad en tus asignaturas de informática, donde revisaste las estructuras de datos; tales conceptos nos ayudarán a aterrizar los nuevos conceptos que empezaremos a revisar.

En la primera unidad, veremos algunos conceptos de matemáticas como son los conjuntos, sus operaciones, las series y sucesiones, los cuales nos serán de utilidad para comprender los conceptos de unidades más avanzadas.

En la segunda unidad repasaremos las funciones de tipo recursivo y de control de flujo que se emplean para mejorar el rendimiento de los programas de computación y que serán de gran utilidad en la aplicación de las estructuras discretas que revisaremos a continuación.

La tercera unidad nos habla de la estructura discreta conocida como relación, revisaremos sus características y propiedades, además de comprenderla como la base de las otras estructuras que veremos en el curso.

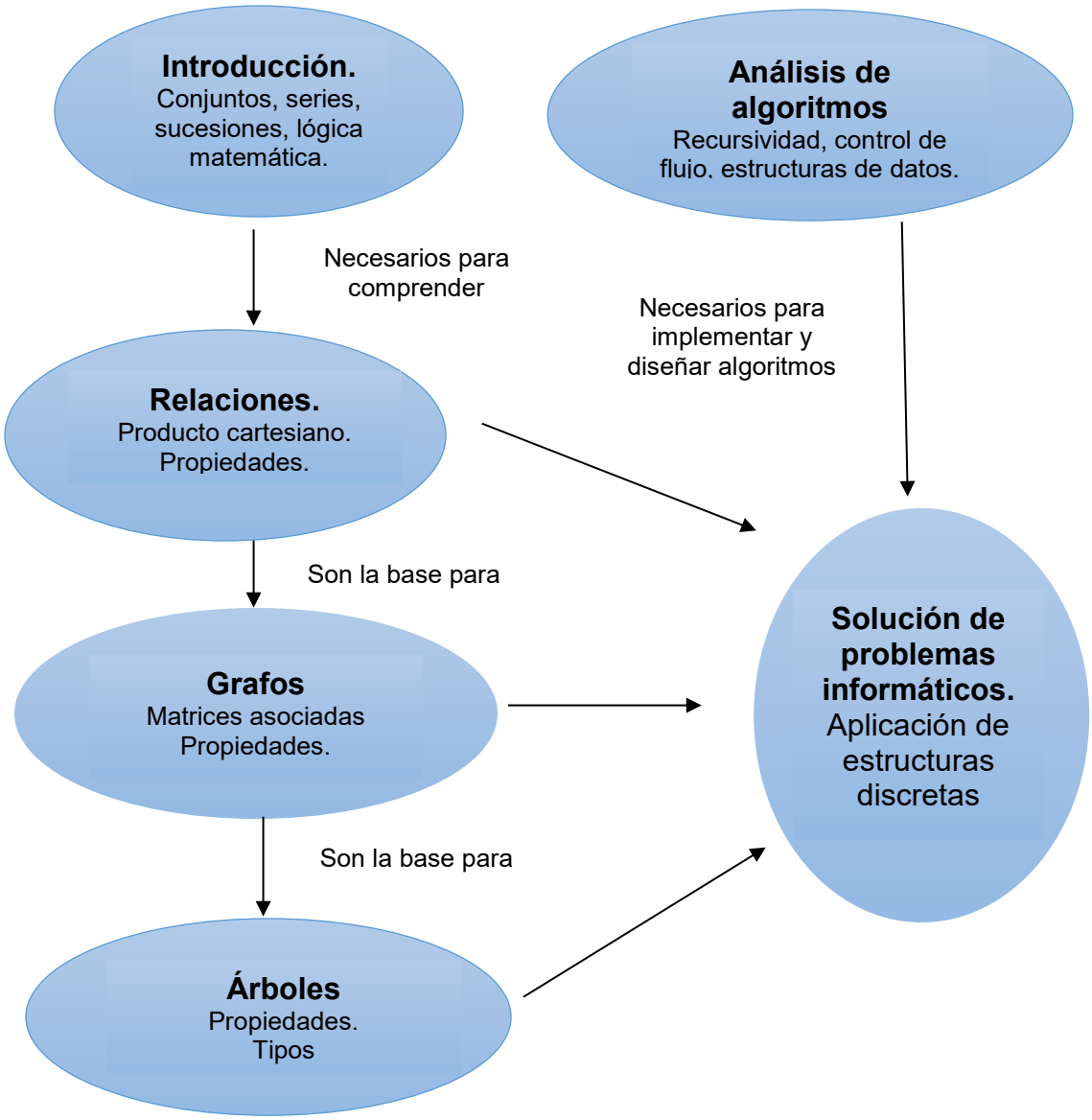
La cuarta unidad nos habla de los grafos dirigidos o dígrafos, que son una representación gráfica de las relaciones, pero que son de utilidad en el análisis de

las mismas desde un punto de vista más simple, también estudiaremos su matriz asociada y las operaciones que pueden realizarse con ella.

La unidad cinco, nos muestra la estructura discreta más compleja, también una de las más empleadas: los árboles, de los que revisaremos sus características, propiedades, operaciones y tipos, a fin de aplicarlos en los algoritmos de computación.

Finalmente, en la unidad 6, veremos algunos casos donde revisaremos la aplicación y la utilidad de las estructuras estudiadas.

# ESTRUCTURA CONCEPTUAL



Licenciatura: Informática

# UNIDAD 1

# Introducción. Unificación de conceptos

SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

# OBJETIVOS PARTICULAR

El alumno identificará los conceptos básicos relacionados con las matemáticas discretas.

# TEMARIO DETALLADO
## (6 horas)

1. Introducción. Unificación de conceptos
   - 1.1. Conjuntos, subconjuntos
   - 1.2. Sucesiones, listas, arreglos
   - 1.3. Índices, subíndices, ∑, ∏
   - 1.4. Operaciones (unión, intersección, complemento, diferencia, concatenación)
   - 1.5. Lógica matemática, proposiciones, tablas de verdad
   - 1.6. Pseudocódigo, algoritmos, diagramas de flujo

# INTRODUCCIÓN

Las matemáticas son una herramienta esencial para la informática, a partir de las operaciones, tanto aritméticas como de conjuntos, es posible manipular los datos que se procesan en las computadoras; en el caso de las bases de datos, es necesario emplear las operaciones con conjuntos para manipular los datos almacenados en las tablas que conforman la base de datos, a estas operaciones se les denomina álgebra relacional; en diseño digital, es necesario emplear de igual forma las operaciones con conjuntos junto con sus propiedades, además de la lógica matemática y el álgebra tradicional para diseñar circuitos lógicos, lo que da como resultado el álgebra booleana.

A lo largo de la presente unidad, repasaremos los conceptos matemáticos necesarios que nos facilitarán entender los conceptos de estructuras matemáticas más complejas, denominadas estructuras de matemáticas discretas o estructuras discretas, que nos permitirán establecer estructuras de datos más complejas, que a la postre serán de utilidad en la manipulación más eficiente de los datos.

# 1.1. Conjuntos, subconjuntos

Dentro del campo de la informática, el manejo de la teoría de conjuntos es necesario para comprender varios de sus campos, tales como las bases de datos, la minería de datos y algunas estructuras de datos complejas, entre otras cosas.

A continuación, repasaremos algunos de los conceptos principales de la teoría de conjuntos para reforzar los conocimientos de asignaturas previas y preparar el camino para el estudio de los conceptos que se abordarán en temas subsecuentes.

"Un conjunto, podemos definirlo como una colección de elementos que pueden ser del mismo tipo o diferentes, a estos elementos se les denomina elementos del conjunto."[1]

Algunos ejemplos de conjuntos son:

El conjunto de los números naturales (1, 2, 3, 4,...)
El conjunto de los números enteros (...-3, -2, -1. 0, 1, 2, 3)
El conjunto de las letras mayúsculas (A, B, C, D,…, Z)

[1] Fuente: Espinosa, E. e Canals, I. (2009). *Cálculo diferencial.* Hong Kong, China: Universidad Autónoma Metropolitana & Editorial Reverté, p. 66.

Podemos encontrar diversas formas de expresar los conjuntos:

1. Por medio de llaves:
   A = {1.2.3.4}
2. De forma descriptiva:
   A = {Todos los valores de x tal que x pertenece a los números naturales}
3. De forma descriptiva, pero empleando símbolos:
   A = {x/x ∈ $\boldsymbol{N}$} (se lee como en el ejemplo del punto 2)

Dentro de los conjuntos, es posible encontrar elementos desordenados, ordenados, repetidos, etc., que no alteran al conjunto en sí, ya que los elementos independientemente de lo anterior forman parte del conjunto.

También es posible encontrar conjuntos que no contienen elementos, a este tipo de conjuntos se les denomina conjunto vacío y se le denota mediante el símbolo ∅.

Ahora bien, los subconjuntos son conjuntos que pertenecen a un conjunto principal, es decir, son elementos de un conjunto grande con ciertas características particulares, por ejemplo:

Los números naturales son un subconjunto de los números enteros.

En el ejemplo anterior, entendemos a los números enteros como aquellos números que van desde el infinito negativo al infinito positivo, incluyendo el cero (...,-4, -3, -2, -1, 0, 1, 2, 3, 4,…), pero como los números naturales van del 1 al infinito positivo se dice que los números naturales forman parte de los enteros, formando de esta forma un subconjunto.

# 1.2. Sucesiones, listas, arreglos

Las sucesiones son una de las aplicaciones de los conjuntos y son elementos necesarios para explicar algunos problemas como la depreciación de algún objeto con el paso del tiempo.

Por ejemplo, si se compra un automóvil en $100,000 y se sabe que se deprecia 30% cada año, podemos deducir su valor a futuro de la siguiente manera:

100,000 x 0.7= 70,000, es decir, al final del primer año el auto vale 70,000.
100,000 x 0.7 x 0.7=100,000 x $(0.7)^2$ = 49,000, al final del segundo año, el valor del auto es de 49,000.

¿Qué pasa al año enésimo?, bueno, podemos establecer la función:

$$100{,}000 \times (0.7)^n$$

La función anterior nos da el valor de automóvil en el año que nosotros establezcamos como el valor de n. En otras palabras, a través de esa función nosotros podemos establecer la sucesión de valores del automóvil en un año determinado.

De esta forma entonces, podemos definir a las sucesiones de la siguiente manera:

Una sucesión es un conjunto ordenado

$$f(1), f(2), f(3), ..., f(n),...$$

formado a partir de una función ***f*** cuyo dominio es el conjunto de todos los números naturales, donde a f(n) se le denomina término enésimo de la sucesión[2].

Dentro de las sucesiones, podemos hablar de un límite de la sucesión, esto es, un número “L” el cual tiende el valor de la sucesión conforme va incrementándose el valor de f(n), de tal forma que dicho límite puede obtenerse mediante:

$$\lim_{n\to\infty} f(n)$$

Tomando la función del ejemplo de la depreciación del automóvil, $f(n) = 0.7^n$ por lo que $\lim_{n\to\infty}(0.7)^n = 0$, con lo que podemos concluir que el límite de la depreciación del automóvil es de cero, en otras palabras, a mayor tiempo el valor del automóvil será de cero.

Un ejemplo del empleo de las sucesiones en informática, es el empleo de funciones de tipo recursivo que permiten calcular valores de forma cíclica, repitiendo operaciones dentro de ese ciclo; este tema se abordará con mayor detalle en la siguiente unidad.

[2] Solar E., Speziale L. (1998). *Algebra I*. México: Limusa/ UNAM-Facultad de Ingeniería, p. 183.

Ejemplos de sucesiones:

$$f(n) = \frac{n+1}{3n} \rightarrow \frac{2}{3}, \frac{1}{2}, \ldots.$$

$$f(n) = \frac{1}{n} \rightarrow 1, \frac{1}{2}, \frac{1}{3}, \ldots.$$

Las listas o arreglos lineales como también se les conoce, son un "conjunto ordenado y finito de elementos homogéneos"[3]. Lo que nos indica que son elementos consecutivos del mismo tipo y con un número finito de elementos, es decir, es un elemento muy parecido a una sucesión, con la diferencia de que la sucesión no tiene elementos finitos, mientras que el arreglo o la lista sí lo tienen. Los ejemplos más comunes en matemáticas son los vectores y las matrices, que son conjuntos de elementos del mismo tipo con tamaños definidos.

Ejemplos:

Vector de posición: 3i+4j+5k = (3, 4, 5)

Matriz de 2 x 2: $\begin{matrix} 2 & 3 \\ 4 & 5 \end{matrix}$

[3] Tomado de: García, Gladys. *Estructuras de datos, estructuras lineales o arreglos*. Universidad Autónoma de Puebla. Consultado el 18/07/2014 de http://www.uap.edu.pe/pregrado1/02/trabajos/02119/PW_alas/cap4_t_arreglos.HTM

# 1.3. Índices, subíndices, ∑, ∏

Los índices y subíndices en matemáticas pueden tener varias interpretaciones, los índices, de forma general, nos pueden indicar pertenencia a un grupo, por ejemplo, $R^2$ indica un conjunto de elementos reales en arreglos de 2 elementos (1, 2), (3, 5), (-5, 3). Otra interpretación del índice es la elevación de un número a una potencia determinada, por ejemplo, $2^2=2*2=4$.

En el caso de los subíndices, nos indican, de forma general, un cierto número de elemento dentro de un arreglo, una serie o una sucesión, por ejemplo, $a_2$ nos indica el tercer elemento dentro de un arreglo, si el arreglo fuera (2, 4, 6, 8), $a_2$ corresponde al número 6.

Dentro del campo del álgebra, encontramos elementos que conocemos como series (estos elementos se representan como la sumatoria de elementos de un mismo tipo), las cuales pueden representar a una función, como es el caso de las series de Maclaurin y de Taylor, o que representan el área encerrada bajo una curva, como el caso de las series de Riemann.

La definición formal de una serie es la siguiente:

*Una serie, es una expresión de la forma:*

$$a_1 + a_2 + a_3 + \ldots + a_n +$$

*cuya forma compactada se representa por:*

$$\sum_{n=1}^{\infty} a_n$$

Donde $a_n$ se le conoce como término enésimo de la serie y son elementos que se obtienen a partir de una función con dominio en los números naturales.

Lo anterior puede resumirse en la siguiente definición:

Sea $\sum_{n=1}^{\infty} \boldsymbol{a_n}$ una serie y sea [$S_n$] una sucesión definida por:

$$\boldsymbol{Sn} = \sum_{\boldsymbol{n=1}}^{\infty} \boldsymbol{a_n}$$

i) Si existe un número S tal que $\lim_{n\to\infty} \boldsymbol{Sn} = \boldsymbol{S}$ se dice que la serie es convergente y tiene suma S, lo cual se denota como:

$$\sum_{\boldsymbol{n=1}}^{\infty} \boldsymbol{a_n} = \boldsymbol{S}$$

ii) Si por el contrario, el límite no existe, se dice que la serie es divergente y no tiene suma[4].

Como ya mencionamos, las series son empleadas para poder representar diferentes tipos de funciones, ya sea dentro del campo de los reales o de los complejos, lo anterior se logra al momento de descomponer dichas funciones en una serie de forma polinomio (denominada serie de potencias o de Taylor), lo que facilita su análisis y nos permite hacer aproximaciones, lo que en informática nos ayuda a convertir señales de tipo analógico en digitales, realizando dichas aproximaciones.

Ejemplos de sucesiones:

Serie convergente

$$\frac{\boldsymbol{3}}{\boldsymbol{10}} + \frac{\boldsymbol{3}}{\boldsymbol{10^2}} + \frac{\boldsymbol{3}}{\boldsymbol{10^3}} + \cdots + \frac{\boldsymbol{3}}{\boldsymbol{10^n}} = \frac{\boldsymbol{1}}{\boldsymbol{3}}\left(\boldsymbol{1} - \frac{\boldsymbol{1}}{\boldsymbol{10^n}}\right)$$

[4] Solar E., Speziale (1998). *Algebra I*. México: Limusa/ UNAM/ Facultad de Ingeniería pp.196 -197.

Donde $a_n = \frac{3}{10^n}$ y $Sn=\frac{1}{3}\left(1-\frac{1}{10^n}\right)$

Para saber si la serie converge:

$$\lim_{n\to\infty}\frac{1}{3}\left(1-\frac{1}{10^n}\right)=\frac{1}{3}$$

Por lo que decimos que la serie converge y su suma es $\frac{1}{3}$.

Una serie de tipo divergente es la siguiente:

$$\frac{n}{6}(2n^2+3n+1)=1+4+9+\cdots+n^2$$

Al tener una función polinomial donde no se tiene un cociente, el límite de n al tender a infinito es infinito, por lo que el límite de la función no existe y por tanto la serie es de tipo divergente.

Existen otro tipo de series que en lugar de emplear sumas infinitas emplean multiplicaciones infinitas, a las cuales se les denomina series de productos, al igual que las series de sumas éstas pueden converger o divergir, y en ambos casos todo depende del límite de la función asociada.

Ejemplos de serie de productos:

$$\prod_{n=2}^{\infty}(1-\frac{1}{n})=\left(\frac{1}{2}\right)\left(\frac{2}{3}\right)\left(\frac{3}{4}\right)....$$

$$\prod_{n=2}^{\infty}\left(1-\frac{1}{n^2}\right)=\left(\frac{3}{4}\right)\left(\frac{8}{9}\right)\left(\frac{15}{16}\right)....$$

En ambos casos, es posible validar ambos tipos de series a partir de la inducción matemática, que estudiaremos en la siguiente unidad.

# 1.4. Operaciones (unión, intersección, complemento, diferencia, concatenación)

Dentro de varios campos de la informática como, por ejemplo, las bases de datos y la arquitectura de computadoras (diseño digital), es necesario el empleo de operaciones matemáticas que nos permiten manipular datos, estos datos al final son conjuntos de diversas índoles, por lo que estas operaciones tienen su fundamento en las operaciones con conjuntos, las cuales repasaremos a continuación.

| Operador | Ejemplos |
|---|---|
| Unión. Permite unir un conjunto A con otro conjunto B, dando como resultado un conjunto más grande.<br>La operación se representa de la siguiente manera: A∪B | A = {2,4,6} B = {1,3,5}<br>$\boldsymbol{A} \cup \boldsymbol{B} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}\}$ |
| Intersección. Esta operación permite obtener aquellos elementos que son comunes en dos o más conjuntos y se representa de la siguiente manera: $\boldsymbol{A} \cap \boldsymbol{B}$ | A = {2,4,6} B = {1,2,3}<br>$\boldsymbol{A} \cap \boldsymbol{B} = \{\mathbf{2}\}$ |
| Complemento. El complemento o conjunto complemento, es un conjunto que contiene a todos los elementos que no se encuentran en un conjunto A. El | Sea A = {3,4,5}, suponiendo que el conjunto universo para A es U = {0,1,2,3,4,5,6,7,8,9} entonces:<br>$A^C$={0,1,2,6,7,8,9} |

| | |
|---|---|
| conjunto complemento se denomina regularme de la siguiente forma: $A^C$ | |
| Diferencia. Esta operación permite obtener los elementos de un conjunto A, que no se encuentran en un conjunto B. Generalmente se denota como A – B. | A = {2,4,6 } B = {1,2,3}<br>$\boldsymbol{A - B = \{4, 6\}}$<br>Para la diferencia se eliminan los elementos del conjunto A que se encuentran en el conjunto B. |
| Concatenación. La concatenación es la unión de dos conjuntos mostrando todos sus elementos sin importar que estos se repitan. El operador generalmente se denota como AB | A = {2,4,6} B = {1,2,3}<br>AB = {2,4,6,1,2,3}<br>Al realizar la concatenación toma precedencia el conjunto de la izquierda, por lo que sus elementos van primero. |

# 1.5. Lógica matemática, proposiciones, tablas de verdad

De acuerdo con el Dr. José Manuel Becerra:

> La lógica matemática es la disciplina que trata de métodos de razonamiento. En un nivel elemental, la lógica proporciona reglas y técnicas para determinar si es o no válido un argumento dado. El razonamiento lógico se emplea en matemáticas para demostrar teoremas, sin embargo, se usa en forma constante para realizar cualquier actividad en la vida.[5]

[5] Becerra E. José M. *Lógica matemática*. Consultado el 18/07/2014 de http://www.fca.unam.mx/docs/apuntes_matematicas/36.%20Logica%20Matematica.pdf.

La lógica define la forma en que asociamos cosas cuando razonamos, la aplicación de esta lógica en la solución de problemas matemáticos da como resultado la lógica matemática y ésta, a su vez, nos da las bases del procesamiento de datos binarios o la lógica binaria, que es la base de la computación.

Los elementos que se emplean para aplicar la lógica matemática se denominan preposiciones, que son enunciados que pueden tener un valor falso o verdadero, pero no ambos valores al mismo tiempo, estas preposiciones pueden ser lingüísticas o matemáticas.

Ejemplo de presuposiciones:

p: México es un país.
q: 3+5 = 7
r: $3x^2 + 3 > 5$

En los ejemplos anteriores, las preposiciones p y q son preposiciones que solo pueden ser falsas o verdaderas, mientras que la preposición r depende del valor de x para que su valor sea falso o verdadero.

De lo anterior, podernos definir diversos tipos de preposiciones:

| | |
|---|---|
| *Simples* | • Si sólo tienen un sujeto, un verbo y un complemento. En caso contrario, son proposiciones compuestas |
| *Cerradas* | • Si tienen determinado el sujeto. |
| *Abiertas* | • Si no lo tienen determinado. |
| *Afirmativas o Negativas* | • Según lo afirmen o nieguen. |
| *Verdaderas o Falsas* | • Según correspondan o no a la realidad. |

De esta forma:

p: México es un país. Es simple, cerrada y afirmativa.

q: 3+5 = 7 Es simple, cerrada, afirmativa y falsa.

r: $3x^2 +3 > 5$ Es simple, abierta y afirmativa.

Dentro de la lógica matemática es posible conjugar preposiciones y dar como resultado preposiciones compuestas con un valor verdadero o falso, para conjugarlas y determinar su valor de verdad tenemos los conectores lógicos que, a su vez, tienen asociadas tablas de verdad que nos ayudan a determinar el valor de la preposición compuesta resultante.

[6] Becerra E. José M. *Lógica matemática. (*p. 1). Consultado el 18/07/2014 de: http://www.fca.unam.mx/docs/apuntes_matematicas/36.%20Logica%20Matematica.pdf.

<table>
<tr><th>Operador</th><th>Ejemplo:</th></tr>
<tr><td>Conjunción (Operador and (y)).<br>La conjunción permite unir dos o más preposiciones y su valor de verdad dependerá del valor de cada preposición por separado de acuerdo a la siguiente tabla:<br><br><table><tr><td>P1</td><td>P2</td><td>R</td></tr><tr><td>V</td><td>V</td><td>V</td></tr><tr><td>F</td><td>V</td><td>F</td></tr><tr><td>V</td><td>F</td><td>F</td></tr><tr><td>F</td><td>F</td><td>F</td></tr></table><br>El operador suele representarse por el símboloΔ.</td><td>p: Voy a comer hamburguesas los martes<br>q: tengo dinero<br>r=pΔq<br>r= Voy a comer hamburguesas los martes y cuando tengo dinero.<br><br>La preposición resultante es condicional a q, p se asume verdadera mientras que q depende de si se tiene o no dinero. Si q es falsa la preposición compuesta será falsa, si es verdadera será verdadera acorde a la tabla.</td></tr>
<tr><td>Disyunción (Operador or (o)).<br>Este operador permite tener una preposición compuesta verdadera siempre y cuando cualquiera de las preposiciones que la componen sea verdadera.<br><br>Símbolo: ∇<br><br>Tabla de verdad:<br><table><tr><td>P1</td><td>P2</td><td>R</td></tr><tr><td>V</td><td>F</td><td>V</td></tr><tr><td>V</td><td>V</td><td>V</td></tr><tr><td>F</td><td>V</td><td>V</td></tr><tr><td>F</td><td>F</td><td>F</td></tr></table></td><td>Para ir a Europa puedo ir en avión o en barco.<br>p: Ir a Europa<br>q: ir en avión<br>r: ir en barco<br><br>El operador que da como resultado la preposición inicial es q∇r, y ambas son verdaderas, por lo que la preposición resultante es verdadera de acuerdo a la tabla de verdad.</td></tr>
</table>

<table>
<tr>
<td>Negación (Operador Not).<br>Este operador cambia el valor de verdad de una preposición al cambiar el sentido de la preposición original.<br><br>Símbolo: ‘Ejemplo p’<br><br>Tabla:<br><table><tr><td>P</td><td>P’</td></tr><tr><td>V</td><td>F</td></tr><tr><td>F</td><td>V</td></tr></table></td>
<td>p: Los gatos son animales<br>p’: los gatos no son animales.</td>
</tr>
</table>

# 1.6. Pseudocódigo, algoritmos, diagramas de flujo

Comencemos por recordar lo que es un algoritmo; por definición, se trata de una serie de pasos ordenados, finitos y precisos que llevan siempre a un mismo resultado.

Una de las principales características de los algoritmos es que deben ser escritos de tal forma que cualquier persona sea capaz de entender y seguir los pasos descritos en él, los algoritmos, por tanto, deben ser escritos en lenguaje natural y en forma precisa.

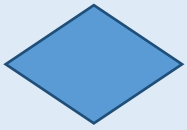
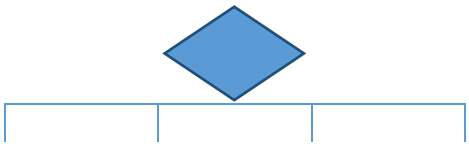
Los algoritmos son empleados de forma cotidiana en todas las actividades y disciplinas, desde recetas de cocina hasta rutinas muy complejas empleadas en computación.

Un ejemplo simple de un algoritmo es la receta para hacer agua de limón.

| |
|---|
| Inicio. |
| Toma una jarra. |
| Llénala a ¾ partes de su capacidad con agua. |
| Corta y exprime en el agua 5 limones. |
| Agrega 3 cucharadas soperas de azúcar en el agua. |
| Agita el agua por cerca de 5 minutos. |
| Agrega 6 hielos. |
| Fin. |

Los algoritmos son la base de los programas informáticos, por lo que después de desarrollarlos es posible representarlos de forma gráfica mediante lo que se denomina un diagrama de flujo.

Los diagramas de flujo, son representaciones gráficas de la secuencia de pasos de un algoritmo, emplea una serie de símbolos para representar cada paso, entradas, salidas, procesos y decisiones descritos en el algoritmo, su simbología es la siguiente:

| SIMBOLO | FUNCIÓN |
|---|---|
|  | Inicio/Fin, para indicar en dónde empieza y termina el diagrama. |
|  | Conector: Sirve para enlazar dos partes cualesquiera de un ordinograma a través de un conector en la entrada. Se refiere a la conexión en la misma página del diagrama. |
|  | Entrada/Salida, cualquier tipo de introducción de datos en la memoria desde periféricos “entrada” o registro de la información procesada en un periférico “salida”. |
|  | Proceso, operación para plantear instrucciones de asignación, tales como: desarrollar una expresión aritmética o mover un dato de un lado a otro. |
|  | Decisión para evaluar una condición y plantear la selección de una alternativa. Normalmente tiene dos salidas -respuestas SI o NO- pero pueden tener tres o más según los casos. |
|  | Selección múltiple: en función del resultado de la comparación se seguirá uno de los diferentes caminos de acuerdo con dicho resultado. |

| | |
|---|---|
|  | Líneas de Flujo, indican el sentido de ejecución de las operaciones. |
|  | Línea Conectora, sirve de unión entre dos símbolos. |
|  | Conector a otra Página, conexión entre dos puntos del organigrama situados en páginas diferentes. |
|  | Proceso Predefinido, es un módulo independiente del programa principal, que recibe una entrada procedente de dicho programa, realiza una tarea determinada y regresa, al terminar, al programa principal. |

Figura 1.1. Simbología de un diagrama de flujo[7].

Retomando el ejemplo del agua de limón, podemos desarrollar su diagrama de flujo correspondiente:

[7] Mercado, William. *Diagrama de flujo.* Universidad Experimental de Guyana. Consultado el 18/07/2014 de: http://macabremoon0.tripod.com/id6.html.

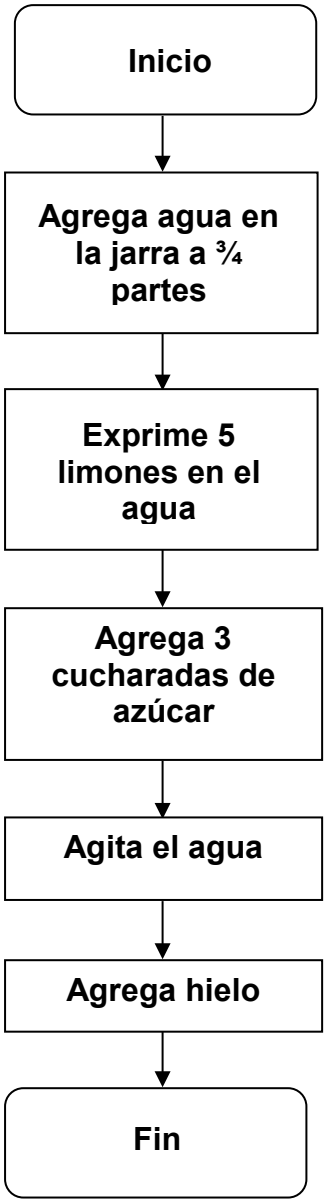



**Figura 1.2 Diagrama de flujo del procedimiento para preparar agua de limón**

Finalmente, otra forma de representar un algoritmo, ya de una forma más cercana a un programa de computadora es el pseudocódigo.

El pseudocódigo es un lenguaje de alto nivel más parecido a un lenguaje de programación, emplea palabras que ayudan a simplificar expresiones u operaciones

de los algoritmos, dentro del pseudocódigo es posible manejar estructuras de datos, funciones de control de flujo, subsunciones, etc.

El pseudocódigo no tiene una estructura estándar definida, ya que puede ser ajustado a la estructura de los diversos lenguajes de programación de alto nivel.

```
Programa Encender lámpara
si (lámpara enchufada) entonce
        si (lámpara encendida) entonces
                problema solucionado
        si no
                si (foco quemado) entonces
                        reemplazar foco
                si no
                        comprar nueva lámpara
                fin si
        fin si
si no
        enchufar lámpara
                si (lámpara encendida) entonces
                        problema solucionado
                si no
                        si (foco quemado) entonces
                                reemplazar foco
                        si no
                                comprar nueva lámpara
                        fin si
                fin si
fin si
Fin Programa
```

Ejemplo de pseudocódigo[8].

[8] Fuentes, Bessy. *Pseudocódigo*. Consultado el 18/07/2014 de: http://20111003091.blogspot.com/2011/07/pseudocodigo.html.

# RESUMEN

Las matemáticas son la base de muchas de las estructuras de datos empleados en el desarrollo de programas de cómputo, siendo la base los conceptos de conjuntos y la lógica matemática.

Los conjuntos tienen una serie de operaciones básicas como la unión, la intersección, diferencia, el complemento y la concatenación, estas operaciones son la base para manipular los elementos que conforman los conjuntos; nos permiten encontrar, agregar y eliminar elementos dentro de los conjuntos, lo que sirve como base para el procesamiento de datos. Las sucesiones, arreglos o listas, son estructuras matemáticas que permiten ordenar elementos de manera secuencial, lo cual se puede traducir en elementos tales como los arreglos de 1 a varias dimensiones dentro de los programas de cómputo.

Otro elemento derivado de las sucesiones, son las series, que ayudan a representar funciones en forma de sumatorias o productos sucesivos infinitos, lo que propicia realizar un análisis más profundo de dichas funciones y representarlas en forma digital.

Finalmente, la lógica matemática ayuda al procesamiento de la información al analizar el valor de verdad de las preposiciones de las sentencias que la componen, por medio de las operaciones lógicas y sus tablas de verdad asociadas.

# BIBLIOGRAFÍA

## SUGERIDA

| Autor | Capítulo | Páginas |
|---|---|---|
| Solar | IV | 182-260 |
| Espinosa | I | 66-69 |

Licenciatura: Informática

# UNIDAD 2

# Análisis de algoritmos

SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

# OBJETIVO PARTICULAR

El alumno aprenderá los conceptos básicos utilizados en el análisis y desarrollo de algoritmos.

# TEMARIO DETALLADO

## (12 horas)

2. Análisis de algoritmos
    - 2.1. Subrutina, función
    - 2.2. Principio de recursividad
        - 2.2.1. *Do – while*
        - 2.2.2. *Do – until*
        - 2.2.3. *For*
        - 2.2.4. *If – then – else*
        - 2.2.5. *Case*
    - 2.3. Desarrollo de algoritmos de modelos matemáticos recursivos.
        - 2.3.1. Factorial
        - 2.3.2. Números primos
        - 2.3.3. Cuadrado de número entero por sumas sucesivas
        - 2.3.4. Operaciones con *matrices*
        - 2.3.5. Solución de sistema de ecuaciones

# INTRODUCCIÓN

Las funciones recursivas se derivan de las sucesiones y series matemáticas, nos ayudan a resolver problemas que presentan rutinas repetidas y, en gran medida, a modelar funciones matemáticas.

En principio, la recursividad se emplea bajo el seguimiento de métodos matemáticos demostrativos, como el de inducción matemática que, por su naturaleza, pretende validar una expresión para cierto número “n” de términos, por lo que la función a emplear debe ser evaluada esas n-veces, es aquí, entonces, donde este tipo de funciones es necesaria.

La recursividad no solo es empleada en procedimientos demostrativos, también se presenta en el carácter incremental u oscilatorio de ciertas funciones como las circulares (seno, coseno, etc.), factoriales, etc.

A lo largo de la presente unidad revisaremos cada una de las diversas funciones recursivas que podemos encontrar y veremos su utilidad en la implementación de algoritmos de diversos modelos matemáticos.

# 2.1. Subrutina, función

Cuando se desarrollan programas de computadora, en muchas ocasiones las líneas de código empleadas son muy extensas, con el objetivo de reducir dicho tamaño se emplean funciones o subrutinas.

Las subrutinas o subprogramas "pueden definirse como un programa dentro de un programa[9]", en otras palabras, se tratan de líneas de código que pueden definirse como pequeños programas que realizan tareas determinadas y que pueden invocarse desde cualquier punto del cuerpo del programa principal, llamando a la subrutina por su nombre.

En el lenguaje C tenemos, por ejemplo, la función o subrutina printf

```
printf("cadena %s\n",$cadena)
```

Esta subrutina se invoca a través de su nombre y los valores o parámetros con los cuales queremos que trabaje.

Las subrutinas en los lenguajes de programación de alto nivel pueden ser creadas por el mismo programador o pueden ser parte del conjunto de funciones del mismo lenguaje.
Las funciones son similares a las subrutinas, con la principal diferencia de que las funciones siempre deben regresar un valor asociado a la misma función.

[9] Contreras Moreira, Bruno. *Subrutinas y pasos de parámetros.* Consultado el 20/07/2014 de: http://www.ccg.unam.mx/~contrera/bioinfoPerl/node42.html

```
[TIPO] FUNCTION NOMBRE(< lista de argumentos de entrada>1)
...
NOMBRE=….
RETURN
END
```

Ejemplo de estructura de una función[10].

## 2.2. Principio de recursividad

"Se dice que una función es recursiva cuando dicha función se define en términos de la misma función"[11]. Lo anterior nos quiere decir que las funciones de tipo recursivo, tienen la propiedad de auto invocarse, esto es, que se pueden llamar a sí mismas hasta completar una tarea.

La recursividad en informática se emplea para reducir problemas a su mínima expresión o para realizar tareas repetitivas, por lo general este tipo de funciones forman parte de los mismos lenguajes de programación, las cuales revisaremos a continuación.

[10]. Gómez, Rosa M. *Subprogramas: funciones y subrutinas.* Universidad de la Laguna, España. Consultado el 20/07/2014 de: http://rgomez.webs.ull.es/ARCHIVOSFMAT0708/clase3_0708.pdf
[11]. Serna Pérez, Eduardo. *Recursividad.* Universidad Autónoma de Aguascalientes. Consultado el 20/07/2014 de: http://www.paginasprodigy.com/edserna/cursos/estddatos/notas/Unidad2.%20Recursividad.pdf.

## a) Do – while

La sentencia *Do-While*, (hacer-mientras), es una función de tipo recursivo que ejecuta de forma repetitiva las instrucciones que contiene mientras la condición establecida dentro de ella sea verdadera.
La estructura general es:

do {

instrucciones;

instrucciones de rompimiento de ciclo;

} *while* (condición);

Ejemplo:

```
# include <stdio.h>
# include <conio.h>
# include <string.h>
void main ()
{
clrscr();
// deckaracion variables
int x=1;
// instrucción do while
do{
gotoxy(10,x+3);printf("%d GATO",x);
x++;} while(x<=10);
getchar();
}
```

Ejemplo de do-while en c[12].

[12] Tomado del sitio web *Programación fácil*. Consultado el 20/07/2014 de: https://www.programacionfacil.com/cpp/ciclo_do_while.html

## b) Do – until

La sentencia *Do-Until* (hacer-hasta) es similar a la sentencia *Do-While*, ya que repite las instrucciones dentro de la función con la diferencia de que se detendrá hasta que se cumpla la condición establecida en ella.

La estructura general es:

*do* {

Instrucciones;

} *until* (condición);

Ejemplo:

```
fib = ones (1,10)
i = 2
do
   i++;
   fib (i) = fib (i-1) + (i-2)
until (1 == 10)
```

Ejemplo de *Do-Until* en C[13].

## c) For

La sentencia *for,* es otra estructura recursiva que permite repetir un conjunto de instrucciones a partir de un valor inicial hasta un valor tope. La estructura general es:

*for* (valor inicial; condición; tipo de incremento)
{
Instrucciones

[13] Eaton, John. *GNU Octave*. Consultado el 20/07/2014 de: https://www.gnu.org/software/octave/doc/interpreter/The-do_002duntil-Statement.html.

}

Ejemplo:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int i; //Usaremos esta variable para contar el número de veces
   for (i=0:i<10;i++)
   {
        printf("%i.- Hola mundo\n", i+1);
   }
   Return 0;
}
```

Ejemplo de uso de for[14].

## d) If – then – else

La sentencia *if-then-else* (si-entonces-de lo contrario) no es en sí una sentencia recursiva, más bien de tipo condicional; como el nombre lo indica, para que las instrucciones que contenga la sentencia se ejecuten debe cumplirse la condición establecida en ella, si es empleada en conjunto la sentencia *else*, de no cumplirse la primera condición se ejecutarán las instrucciones contenidas en el *else*.

Estructura:

*if (*condición)
{
    Instrucciones.
}
*else*
{
    Instrucciones.
}
Ejemplo:

[14] Tomado de: *Manual de programación en C*: *Sentencias de control de flujo*. Grupo The neowriter. Consultado el 20/07/2014 de: http://whitehathacking.wordpress.com/2010/11/16/manual-de-programacion-en-c-sentencias-de-control-de-flujo.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int numero1;
   int numero2;
   printf("Introduce un número:  ");
   scanf("%i",&numero1);
   printf("\nIntroduce otro número:  ");
   scanf("%i",&numero2); printf("\n");

   if (numero1 > numero2)
   {
      printf("El primer número es mayor que el segundo\n");
   }
   else
   {
      printf("El segundo número es mayor que el primero\n");
   }
   return 0;
}
```

Ejemplo de sentencia *if-then-else*[15].

## e) Case

La sentencia *Case* o *Switch* (como se reconoce en lenguaje C), es otra función de tipo condicional, ya que permite colocar varias opciones o condiciones que pueden cumplirse dentro de su estructura y puede contener varias instrucciones diferentes.

[15] *Ídem.*

Su estructura es la siguiente:

| |
|---|
| *switch* (variable) |
| { |
| *case* opción 1: |
| Instrucciones |
| *break*; //Instrucción de salida |
| *case* opción 2: |
| Instrucciones |
| *break*; |
| |
| *case* opción N; |
| Instrucciones |
| *break*; |
| *default:* |
| Instrucciones que se ejecutan al no contener ninguna de las opciones. |
| *break*; |
| } |

Ejemplo:

```
A = 7;
switch (A)
  case {6,7}
    printf ("variable is either 6 or 7\n"
  otherwise
    printf ("variable is neither 6 nor 7\n"
endswitch
```

Ejemplo de uso de case (*switch*) en C[16].

[16] Eaton, John. *GNU Octave. Sentencia Switch.* Consultado el 20/07/2014 de: https://www.gnu.org/software/octave/doc/interpreter/The-switch-Statement.html#The-switch-Statement.

# 2.3. Desarrollo de algoritmos de modelos matemáticos recursivos

Una vez recordada la estructura básica de las diversas funciones recursivas que se pueden presentar dentro de los diversos lenguajes de programación, revisaremos algunos ejemplos donde pueden aplicarse al representar diversos modelos matemáticos.

## 2.3.1. Factorial

Los números factoriales son números que muestran el resultado de multiplicar todos los números naturales de manera sucesiva, es decir, si yo deseo encontrar el número 3! (el signo "!" indica que el número buscado es un factorial), lo que debo de hacer es 3!=1x2x3=6, debo de multiplicar todos los números naturales desde el 1 hasta el 3 que es el que deseo.

Podemos desarrollar un algoritmo sencillo que nos permita encontrar el factorial de cualquier número empleando una sentencia *for* o un *do while*.

Inicio.
Declaro x como entera
Declaro y como entera
Declaro z como entera
x=0
Imprimo mensaje "dame el número factorial que deseas obtener"

leer valor y almacenar en x.
si x < 0 entonces (sentencia if)
“No hay factoriales negativos”
Fin de programa
Fin del si
Si x=0 entonces
z=1
Imprime Z
Fin del programa
Fin del si
y=1
z=1
Desde y = 1 hasta y = x hacer (sentencia *for*)
z = z * y
y = y + 1
Fin desde.
Imprime z
Fin del programa

En este algoritmo podemos ver como las funciones recursivas nos ayudan a simplificar el proceso.

## 2.3.2. Números primos

Los números primos son aquellos enteros positivos, que solamente pueden dividirse entre sí mismos y entre el 1, algunos de ellos son 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, etc.

En la siguiente dirección electrónica encontrarás un algoritmo que permite obtener números primos de una base de datos, notarás que dentro del mismo se emplean varias funciones de recursividad.[17]

## 2.3.3. Cuadrado de número entero por sumas sucesivas

Elevar a una potencia un número no es otra cosa que multiplicar dicho número por el número de veces que indica la potencia, por ejemplo: $2^3=2*2*2 = 8$
También es posible obtener dicha potencia a través de sumas sucesivas:
$2^3= 2+ 2+2+ 2 = 8$.

En el caso del algoritmo que se solicita es simple, ya que solamente se pide el cuadrado de un número entero en forma de sumas,

$2^2 = 2+2=4$
$3^2=3+3+3=9$
$4^2=4+4+4+4=16$
$5^2=5+5+5+5+5=25$

Podemos detectar un patrón, cuando se nos solicita un número al cuadrado, debemos sumar dicho número las mismas veces, en el caso del 2 son 2, en el 3 son 3, en el 4 son 4 y así sucesivamente.

[17] Cid, Eva. Godino, Juan. Batanero, Carmen. *Sistemas numéricos y su didáctica para maestros.* Consultado el 27/03/2016 de: https://www.ugr.es/~jgodino/edumat-maestros/manual/2_Sistemas_numericos.pdf.

Podemos entonces escribir el siguiente algoritmo:

Inicio
Declara x como entera.
Declara y como entera.
Declara r como entera.
Pide el número a elevar al cuadrado y guárdalo en x.
y = 0
r = 0
Desde y=0 hasta y=x hacer
  r = r + x
  y = y+1
Fin desde
Imprime r
Fin del programa

## 2.3.4. Operaciones con matrices

Las matrices son arreglos de varias dimensiones o m – renglones y n – columnas, lo que se determina de una matriz de mxn elementos.

Ejemplo:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \boldsymbol{Matriz\ de\ 2\ x\ 2}$$

$$\begin{bmatrix} 1 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \boldsymbol{Matriz\ de\ 2x3}$$

Las operaciones básicas que podemos hacer con las matrices son la suma, multiplicación por un escalar y la multiplicación de matrices.

Para poder realizar la suma de matrices, la primera condición que debe de cumplirse es que las matrices deben ser del mismo tamaño, de lo contrario no será posible realizar la suma:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1+3 & 4+2 \\ 3+5 & 6+4 \\ 5+7 & 6+8 \end{bmatrix} = \begin{bmatrix} 4 & 6 \\ 8 & 10 \\ 12 & 14 \end{bmatrix}$$

Como podemos ver en el ejemplo anterior, las matrices se suman elemento por elemento de acuerdo a su posición, por lo que una matriz que no sea del mismo tamaño no permitiría realizar la suma.

La multiplicación por un escalar también es simple, para ello solamente se deben de multiplicar todos los elementos de la matriz por el número escalar.

$$(3)\begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \end{bmatrix} = \begin{bmatrix} 6 & 12 & 18 \\ 3 & 9 & 15 \end{bmatrix}$$

En este ejemplo, todos los elementos de la matriz se multiplican por el escalar 3, el resultado de la operación siempre será una matriz del mismo tamaño que la que se multiplicó.

La resta de matrices si bien no existe como tal, puede efectuarse mediante la suma de matrices, multiplicando la matriz que se desea restar por un escalar unitario negativo y realizando la suma.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} + (-1)\begin{bmatrix} 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1-3 & 4-2 \\ 3-5 & 6-4 \\ 5-7 & 6-8 \end{bmatrix} = \begin{bmatrix} -2 & 2 \\ -2 & 2 \\ -2 & -2 \end{bmatrix}$$

Finalmente, la multiplicación entre matrices, esta operación se puede realizar entre matrices cuadradas del mismo tamaño (las matrices cuadradas son aquellas que

tienen el mismo número de renglones que de columnas 2x2, 3x3, 4x4) o entre matrices conformables, es decir, matrices que tengan el mismo número de columnas de la primera matriz y de renglones en la segunda; por ejemplo, si la primera matriz es de 2x2 la segunda debe de ser de 2x3, el resultado será una matriz de 2x3, ejemplo:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} (1*1)+(2*4) & (1*2)+(2*5) & (1*3)+(2*6) \\ (3*1)+(4*4) & (3*2)+(4*5) & (3*3)+(4*6) \end{bmatrix}$$

$$= \begin{bmatrix} 9 & 12 & 15 \\ 19 & 26 & 33 \end{bmatrix}$$

La multiplicación, como podemos ver en el ejemplo anterior, para obtener el primer elemento del primer renglón se deben multiplicar los elementos del primer renglón de la primera matriz por los de la primer columna, el resultado de cada multiplicación se suma y se obtienen el elemento correspondiente, el segundo elemento del primer reglón se obtiene multiplicando los elementos del primer renglón de la primera matriz por la segunda columna de la segunda, y así sucesivamente hasta terminar con todas las columnas de la segunda matriz, después se toma el segundo renglón de la primer matriz y se multiplica por la primera columna de la segunda, se suman y se obtiene el elemento correspondiente, así se continúa hasta terminar nuevamente de multiplicar el segundo renglón por todas las columnas de la segunda matriz.

En la siguiente liga encontraras un ejemplo de un algoritmo que realiza suma de matrices:

*Aprender programación en C, C++ y Pascal. Suma de matrices.*

http://cypascal.blogspot.mx/2012/07/suma-de-matrices-en-c-arrays.html

Dentro del ejemplo podrás ver el uso de arreglos multidimensionales y algunas sentencias recursivas.

## 2.3.5. Solución de sistema de ecuaciones

Una ecuación lineal sobre el campo de los números complejos, es una expresión de la forma:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$$

donde $a_1$, $a_2$,…, $a_n$ y b pertenecen al campo de los números complejos[18].

Ejemplo: $3x + 4y + 6z = 1$

Los sistemas de ecuaciones nos ayudan a representar una serie de valores desconocidos (variables) y relacionarlos con valores conocidos (constantes) para posteriormente poder conocer esos valores ignorados.

Entonces, un sistema de ecuaciones lineales puede definirse como "un conjunto de ecuaciones lineales con las mismas incógnitas o variables[19]", la solución a dicho sistema será aquella que satisfaga al mismo tiempo las condiciones de cada una de las ecuaciones que integren al sistema.

De esta forma, podemos clasificar los sistemas de ecuaciones de la siguiente manera:

- Compatibles. Sistemas que tienen una o varias soluciones.
- Incompatibles. Sistemas sin solución.

Para poder encontrar la solución de un sistema de ecuaciones podemos emplear varias metodologías, a continuación revisaremos una de las más empleadas, la metodología de Gauss.

[18] Solar E. Speziale L. (1991). *Algebra Lineal* (Segunda edición). México: Limusa/ UNAM/ Facultad de Ingeniería, p.293.
[19] Ídem.

La metodología de Gauss busca reducir el sistema de ecuaciones de tal forma que su estructura quede escalonada, de esta forma cada ecuación puede ser multiplicada por un escalar, sumarse o restarse a otra ecuación para poder alcanzar dicho escalonamiento.

Ejemplo.

Tenemos el siguiente sistema de ecuaciones:

$$\begin{aligned} x_1 + x_2 + 2x_3 &= 3 \\ 3x_1 + 4x_2 + x_3 &= -1 \\ -2x_1 - 4x_2 - x_3 &= 0 \end{aligned}$$

El sistema de ecuaciones puede ser representado en forma matricial de la siguiente manera:

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{3} & \mathbf{4} & \mathbf{1} & \mathbf{-1} \\ \mathbf{-2} & \mathbf{-4} & \mathbf{-1} & \mathbf{0} \end{bmatrix}$$

Donde la primera columna representa a los coeficientes de $x_1$; la segunda, a los de $x_2$; la tercera, a los de $x_3$ y la cuarta, a los valores a los que están igualadas las ecuaciones.

Para solucionar el sistema podemos hacer lo siguiente:

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{3} & \mathbf{4} & \mathbf{1} & \mathbf{-1} \\ \mathbf{-2} & \mathbf{-4} & \mathbf{-1} & \mathbf{0} \end{bmatrix}$$

Multiplicamos el primer renglón por 2 y lo sumamos al tercero, de esta forma nos queda:

$$\begin{bmatrix} 2 & 2 & 4 & 6 \\ 3 & 4 & 1 & -1 \\ 0 & -2 & 3 & 6 \end{bmatrix}$$

Ahora multiplicamos el primer renglón por -3/2 y se lo sumamos al segundo.

$$\begin{bmatrix} -3 & -3 & -6 & -9 \\ 0 & 1 & -5 & -10 \\ 0 & -2 & 3 & 6 \end{bmatrix}$$

Dividimos el primer renglón entre -3 para regresarlo a su forma original.

$$\begin{bmatrix} 1 & 1 & 2 & 3 \\ 0 & 1 & -5 & -10 \\ 0 & -2 & 3 & 6 \end{bmatrix}$$

En la última matriz, podemos ver que todos los elementos de la primera columna, salvo el primero, son cero, con lo que el proceso de escalonamiento va por buen camino. En el segundo renglón también podemos notar que el segundo elemento es un 1, por lo que lo vamos a tomar como punto de referencia o pivote para realizar el siguiente escalonamiento.

Ahora, tomamos el segundo renglón y lo multiplicamos por 2 y se lo sumamos al tercer renglón.

$$\begin{bmatrix} 1 & 1 & 2 & 3 \\ 0 & 2 & -10 & -20 \\ 0 & 0 & -7 & -14 \end{bmatrix}$$

Ahora volvemos a multiplicar el primer renglón por ½ para dejarlo en su estado anterior y el tercer renglón lo multiplicamos por -1/7 para dejar el tercer elemento en valor unitario.

$$\begin{bmatrix} 1 & 1 & 2 & 3 \\ 0 & 1 & -5 & -10 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

Como podemos ver, el sistema de ecuaciones ya se encuentra en forma escalonada, por lo que podemos establecer la siguiente equivalencia:

$x_1 + x_2 + 2x_3 = 3$

$x_2 – 5x_3 = -10$

$x_3 = 2$

Como podemos observar, del escalonamiento ya tenemos el valor de la variable $x_3$, por lo que ahora solo queda sustituir dicho valor en la segunda ecuación para conocer $x_2$ y, finalmente, ambos valores en la primera ecuación para conocer $x_1$,, de esta forma nos queda:

$x_2 = -10 + (5)(2) = 0$

$x_1 = 3 – 0 – 2(4) = -1$

Con lo anterior queda resulto nuestro sistema de ecuaciones, por lo que decimos que es un sistema compatible[20].

Puedes consultar *Eliminación de Gauss-Jordan en C* publicado en la red por Harvey, Triana; el ejemplo de algoritmo que puede servir para resolver sistemas de ecuaciones.[21]

[20] Solar E. Speziale L. (1991). *Algebra Lineal* (Segunda edición). México: Limusa/UNAM/Facultad de Ingeniería, pp. 302-305.

[21] Harvey, Triana. *Eliminación de Gauss-Jordan en C.* Consultado el 20/07/2014 de: http://vexpert.mvps.org/articles/GJE.htm

# RESUMEN

Los métodos de recursivos son estructuras de datos esenciales para la programación que permiten resolver problemas haciéndolos cada vez más simples, esto es, permiten tomar decisiones y repetir instrucciones que nos ayudan a simplificar el desarrollo de nuestros algoritmos y, por ende, simplificar el problema.

Dentro de los diversos modelos matemáticos que empleamos para resolver diversos tipos de problemas, desde el manejo de números factoriales, primos, operaciones con matrices, sistemas de ecuaciones, etc., podemos encontrar diversos tipos de estructuras de datos y funciones recursivas que posibilitan su aplicación en una computadora.

Las funciones recursivas y de control de flujo son herramientas que nos abrirán el camino para poder implementar soluciones cada vez más complejas y, por ende, hacer eficientes nuestras tareas.

# BIBLIOGRAFÍA

## SUGERIDA

| Autor | Capítulo | Páginas |
|---|---|---|
| Solar | V, VI | 291 – 341 |
| Cornell | IV | 171 – 202 |
| Swokowski | 7 | 197-247 |

Licenciatura: Informática

# UNIDAD 3

# Relaciones

SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

# OBJETIVO PARTICULAR

El alumno conocerá los tipos y propiedades de las relaciones.

# TEMARIO DETALLADO

## (10 horas)

3. Relaciones
   - 3.1. Conjunto producto y particiones
   - 3.2. Caminos, recorridos, sucesiones
   - 3.3. Tipos de relaciones
   - 3.4. Propiedades de las relaciones
   - 3.5. Matriz asociada a una relación

# INTRODUCCIÓN

El principio de las matemáticas discretas y, por ende, de las estructuras discretas empleadas en informática son los conjuntos.

La primera estructura discreta que revisaremos son las relaciones, éstas son una forma de representación abstracta de las relaciones existentes en la vida cotidiana, las relaciones de cualquier tipo se establecen a partir de un conjunto de objetos o personas con ciertos gustos, preferencias o reglas.

En las matemáticas discretas, veremos que estas relaciones serán establecidas a través de reglas de correspondencia o asociación, que surgen a partir de operaciones con conjuntos.

A lo largo de la presente unidad, revisaremos los conceptos de las relaciones, partiendo de la operación básica que permite su creación, el producto cartesiano. Revisaremos sus propiedades, sus tipos y la estructura que al final del día nos permitirá implementarla en un programa de computadora, su matriz de relación.

# 3.1. Conjunto producto y particiones

Las relaciones son parte de nuestra vida cotidiana, siempre podemos encontrar relaciones entre varias cosas, como el agua y la vida, los números y las operaciones, las flores y las estaciones del año, etc.

Podemos ver una relación como el conjunto de dos conjuntos diferentes, como por ejemplo hombre-mujer, altitud-latitud, etc., es decir, una relación permite unir de alguna forma conjuntos de diversa índole o similares.

Para comenzar a entender las relaciones desde el punto de vista matemático, entendamos el concepto de producto cartesiano o conjunto producto como también se le denomina.

"Dados dos conjuntos A y B, se llama conjunto producto o producto cartesiano de A x B al conjunto de todos los pares ordenados cuyas primeras componentes son de A y las segundas de B.

$\boldsymbol{A\ x\ B} = \{(\boldsymbol{x}, \boldsymbol{y})/\boldsymbol{x} \in \boldsymbol{A}\ \Delta\ \boldsymbol{y} \in \boldsymbol{B}\}$"[22]

De manera general, si tenemos al conjunto A = {1, 2, 3} y B = {a, b, c}, el producto cartesiano de ambos se da como:

A x B= {(1, a), (1, b), (1, c), (2, a), (2, b), (2, c), (3, a), (3, b), (3, c)}

[22] Núñez, Ana M. *Relaciones y funciones*. Universidad de Mendoza, Argentina. Consultado el 21/07/2014 de: http://www.um.edu.ar/catedras/claroline/backends/download.php?url=L0FwdW50ZXMvUmVsYWNpb25lc195X2Z1bmNpb25lcy5wZGY%3D&cidReset=true&cidReq=TAYGAR

Podemos observar entonces que el producto cartesiano da como resultado los pares ordenados de todos los elementos de A con todos los elementos de B, teniendo prevalencia el orden de los elementos del conjunto A. Con el concepto de producto cartesiano establecido, podemos entonces definir el concepto de una relación.

Sean los conjuntos $A_1$, $A_2$, $A_3$,…., $A_n$. Una relación $R$ sobre $A_1$ x $A_2$ x $A_3$ x…x $A_n$ es cualquier subconjunto de este producto cartesiano, es decir:

$$R \subseteq A_1 \times A_2 \times A_3 \times \cdots \times A_n$$ [23]

O sea, una relación es un subconjunto del producto cartesiano de dos o más conjuntos. Por ejemplo:
El producto cartesiano de $R^2$ = R x R (R números reales), donde $R^2$ hace referencia al plano cartesiano x e y, de tal forma que el resultado del producto cartesiano del eje *x* con el eje *y* da como resultado todas las parejas ordenas en un plano (x, y), por tanto, una relación debe de ser un subconjunto que pertenezca a dicho plano, la función y=x es una relación, ya que sus pares ordenados siempre serán parte de $R^2$, si tabulamos tenemos:

| x | y |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |

Tenemos un conjunto de parejas ordenadas (1,1), (2,2), (3,3), etc., dichas parejas forman parte del plano cartesiano (x, y), por lo que son una relación.

[23] González Gutiérrez, Francisco J. (2004*). Apuntes de matemáticas discretas*. Universidad de Cádiz, España. Consultado el 22/07/2014 de: http://www2.uca.es/matematicas/Docencia/ESI/1710003/Apuntes/Leccion6.pdf.

Una partición de un conjunto A, es una colección P de subconjuntos no vacíos de A, tal que:

- Cada elemento de A pertenece a uno de los conjuntos en P
- Si $A_1$ y $A_2$ son dos elementos distintos de P, entonces $A_1 \cap A_2 = \phi$ [24]

Lo anterior nos indica, que la partición de un conjunto A es un subconjunto de los elementos únicos que pertenecen al conjunto A, de tal forma que este subconjunto, al realizar la intersección con otros subconjuntos de A, dé como resultado el conjunto vacío.

Por ejemplo:

Sean P= {1, 2, 3, 4, 5, 6, 7}, podemos "partir" el conjunto P de la siguiente manera:
$A_1$ = {1, 3}, $A_2$ = {2, 4}, $A_3$ = {5, 6, 7}
Como podemos observar, tanto $A_1$, $A_2$ y $A_3$ son subconjuntos de P, y a su vez son particiones de P ya que:

$$A_1 \cap A_2 = \phi, A_1 \cap A_3 = \phi \; y \; A_2 \cap A_3 = \phi$$

Por lo que podemos decir que la colección de elementos {$A_1$, $A_2$, $A_3$} es una partición del conjunto P.

[24] Kolman, B. *et al*. (1996). *Estructuras de matemáticas discretas para la computación* (3ª. edición). México: Prentices Hall, p. 103.

# 3.2. Caminos, recorridos, sucesiones

Se le denomina dominio o camino de una relación R al conjunto formado por todos los primeros elementos de los pares ordenados que pertenecen a R; e imagen, rango o recorrido al conjunto formado por todos los segundos elementos, es decir:

Si R es una relación de A a B, entonces:

$$Dom(R) = \{a \in A, \exists b : b \in B \wedge (a,b) \in R\}$$

$$img(R) = \{b \in B, \exists a : a \in A \wedge (a,b) \in R\}$$ [25]

Retomando el ejemplo de $R^2$

Si sabemos que A= {(1,3), (1,5), (2, 4), (2, 6)} es una relación en $R^2$ ya que A es un subconjunto de $R^2$, entonces:
Dom (A) = {1,2} ya que solamente 1 y 2 aparecen como elementos en x.
Img (A) = {3.4.5.6} ya que son todos los elementos que aparecen en y.

[25] González Gutiérrez, Francisco J. (2004). *Apuntes de matemáticas discretas*. Universidad de Cádiz, España. Consultado el 22/07/2014 de: http://www2.uca.es/matematicas/Docencia/ESI/1710003/Apuntes/Leccion6.pdf.

# 3.3. Tipos de relaciones

Existen diversos tipos de relaciones dependiendo de la forma en que se desarrollan:

- SI $R = \phi$ se dice que la relación es vacía.
- Si $R = A_1 \times A_2 \times A_3 \times \cdots \times A_n$ se dice que la relación es universal.
- Si $A_i = A, \forall i = 1,2,3,\cdots,n$ entonces se dice que es una relación n-aria sobre A.
- Si n = 2 diremos que R es una relación binaria y si n=3 diremos que es terciaria[26].

Dentro de las relaciones, las más importantes son las binarias, ya que son las que se producen con más frecuencia, como, por ejemplo, novio-novia, coordenadas (x, y), cliente-servicio, etc.

La notación más empleada para las relaciones binarias es la siguiente:

- Si una pareja ordenada (a, b) pertenece a una relación, escribimos $aRb$, lo que leemos como "a se relaciona con b".
- Si (a, b) no pertenecen a una relación, lo denotamos como $a\not{R}b$, lo que leemos como “a no se relaciona con b”.

La relación R normalmente se encuentra definida por una regla de correspondencia que permite asociar a los 2 elementos.

[26] González Gutiérrez, Francisco J. (2004). *Apuntes de matemáticas discretas*. Universidad de Cádiz, España. Consultado el 22/07/2014 de: http://www2.uca.es/matematicas/Docencia/ESI/1710003/Apuntes/Leccion6.pdf.

Retomando el ejemplo de $R^2$, una regla de correspondencia para una relación sería $y=x^2$ , donde algunos elementos de la relación resultante serían: R={(1,1), (2,4), (3,9), ...}, donde podemos observar que el dominio de la relación se denota por todos los números reales (R), mientras que la imagen se da por el cuadrado de los números reales positivos, así, un elemento que no pertenece a la relación puede ser (2, 3), ya que 3 no es el cuadrado de 2, por lo que no se cumple con la regla de correspondencia de la relación.

# 3.4. Propiedades de las relaciones

Las relaciones pueden tener varias propiedades como son:

***1. Reflexividad.***

Una relación binaria R sobre un conjunto A se dice que es reflexiva, cuando cada elemento de A se relaciona consigo mismo[27].

$$R\ es\ reflexiva \Leftrightarrow \forall a\left(a \in A \Rightarrow aRa\right)$$

Por ejemplo:

Supongamos que tenemos el conjunto A= {a, b, c, d}
Una relación reflexiva sería:

R= {(a, a), (a, b), (b, a), (b, b), (b, c), (c, c), (c, d), (d, d)}

Podemos ver que cada elemento de la relación es un par ordenado que contiene a los elementos del conjunto A relacionados entre sí y, principalmente, a los pares (a, a), (b, b), (c, c) y (d, d).

[27] Ídem.

***2. Simetría.***

Se dice que una relación R sobre un conjunto A es simétrica si cada que a se relaciona con b, se encuentra la relación de b con a[28].

$$R\ es\ simétrica \Leftrightarrow \forall a,b \in A\left(aRb \Rightarrow bRa\right)$$

Retomando el conjunto A del ejemplo anterior A= {a, b, c, d}.

Una relación simétrica puede ser:

R= {(a, a), (a, b), (b, a), (b, c), (c, b), (d, d)}

Podemos observar que efectivamente, a la relación (x, y) le sigue una relación (y, x).

***3. Asimetría.***

Se dice que una relación R sobre un conjunto A es asimétrica si cada que a se relaciona con b, no se encuentra la relación de b con a[29].

$$R\ es\ asimétrica \Leftrightarrow \forall a,b \in A\left(aRb \Rightarrow b\not Ra\right)$$

Retomando nuevamente el conjunto A del ejemplo anterior A= {a, b, c, d}.

Una relación asimétrica puede ser:

R= {(a, a), (a, d), (b, c), (b, d), (c, a), (d, c)}

En este ejemplo podemos ver que ninguna de las parejas ordenadas que conforman la relación tiene la secuencia (x, y), (y, x).

***4. Antisimetría.***

Se dice que una relación R sobre un conjunto A es antisimétrica si cuando (a, b) ∈ R y (b, a) ∈ R, entonces a = b[30].

[28] Ídem.
[29] Ídem.
[30] Ídem.

$$R\ es\ antisimétrica \Leftrightarrow \forall a,b \in A\left(aRb \wedge bRa \Rightarrow a = b\right)$$

Otra forma de expresarla es:

$$R\ es\ antisimétrica \Leftrightarrow \forall a,b \in A\left(aRb \wedge bRa \Rightarrow a \neq b\right)$$

Del ejemplo anterior A= {a, b, c, d}.

Una relación antisimétrica puede ser:

R= {(b, a), (c, a), (c, b), (d, a), (d, b), (d, c)}

En este ejemplo podemos ver que ninguna de las parejas ordenadas que conforman la relación cumple con la segunda forma de la antisimetría.

***5. Transitividad.***

Se dice que una relación R sobre un conjunto A es transitiva si cuando (a, b) ∈ R y (b, c) ∈ R, entonces (a, c) ∈ R[31].

$$R\ es\ transitiva \Leftrightarrow \forall a,b,c \in A\left(aRb \wedge bRc \Rightarrow aRc\right)$$

Un ejemplo de relación transitiva es:

A = {a, b, c, d}

R = {(a, b), (a, c), (a, d), (b, c)}

En la relación anterior se encuentran los pares (a, b), (b, c) y (a, c), por lo que cumple con la transitividad.

[31] Ídem.

# 3.5 Matriz asociada a una relación

Una forma de representar relaciones entre dos conjuntos es mediante una matriz booleana, es decir, una matriz con valores de 1 y 0 solamente, donde el 1 representa a los elementos relacionados y el 0 a los no relacionados, su definición formal es la siguiente:

Dados dos conjuntos finitos no vacíos,

$$A= \{a_1, a_2,..., a_n\} \text{ y } B= \{b_1, b_2,..., b_n\}$$

y una relación R cualquiera de A a B, llamaremos matriz de R a la matriz booleana siguiente:

$$M_R = (r_{i,j}); r_{i,j} \begin{cases} 1, si\ (a_i, b_j) \in R \\ 0, si (a_i, b_j) \notin R \end{cases}$$

Donde i=1, 2,3,..., n y j=1, 2,3,..., m[32].

Otra forma de ver la matriz, es viendo a la misma como una tabla de la forma siguiente:

| | a | b | c | d |
|---|---|---|---|---|
| a | 1 | 0 | 1 | 0 |
| b | 0 | 1 | 1 | 0 |
| c | 0 | 0 | 1 | 0 |
| d | 0 | 0 | 1 | 0 |

[32] González Gutiérrez, Francisco J. (2004). *Apuntes de matemáticas discretas*. Universidad de Cádiz, España. Consultado el 22/07/2014 de: http://www2.uca.es/matematicas/Docencia/ESI/1710003/Apuntes/Leccion6.pdf.

Con lo que es posible obtener la relación asociada a la matriz, en este caso R= {(a, a), (a, d), (b, b), (b, c), (c, c), (d, c)}.

Una forma más clara de poder determinar las propiedades de una relación es mediante la forma de sus matrices.

***1. Matriz reflexiva.***

R= {(a, a), (a, b), (b, a), (b, b), (b, c), (c, c), (c, d), (d, d)}

$M_r = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ Como podemos ver, cuando una relación es reflexiva, todos los elementos de su diagonal principal son 1.

***2. Matriz simétrica***

R= {(a, a), (a, b), (b, a), (b, c), (c, b), (c, c)}

$M_r = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ Como podemos ver, cuando una relación es simétrica, los elementos a los lados de la diagonal principal se reflejan el uno al otro.

***3. Matriz asimétrica.***

R= {(a, a), (a, d), (b, c), (b, d), (c, a), (d, c)}

$M_r = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ Como podemos ver, cuando una relación es asimétrica, los elementos a los lados de la diagonal principal de la matriz no se reflejan el uno al otro.

***4. Matriz antisimétrica.***

R= {(b, a), (c, a), (c, b), (d, a), (d, b), (d, c)}

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

Como podemos ver, cuando una relación es antisimétrica, los elementos del triángulo superior o inferior de la matriz deben ser o todos unos o todos ceros, no pudiendo ser, ambos, ceros o unos, de lo contrario se trataría de una matriz simétrica.

***5. Matriz transitiva.***

R = {(a, b), (a, c), (a, d), (b, c)}

$M_r = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ Como podemos ver, cuando una relación es transitiva, se forma una especie de T entre sus elementos, indicando con esto que a se relaciona con b, b con c y, adicionalmente, a con c.

Las matrices de las relaciones, al ser estructuras de tipo booleano, son las que nos permitirán establecer algoritmos para manejar esta estructura discreta en la computadora, pues al ser sus valores 1 y 0 es más fácil su programación.

# RESUMEN

Las relaciones es la primera estructura de matemáticas discretas que revisamos a detalle, hemos visto cómo parten de una operación con conjuntos como es el producto cartesiano, y es a partir de él como podemos establecer lo que es una relación, un subconjunto del producto cartesiano entre dos o más conjuntos.

Las relaciones pueden ser de diferentes tipos: vacías, terciarias, universales, etc.; las más comunes son las binarias.

Las relaciones binarias generan pares ordenados del tipo (x, y), pudiendo ser los elementos x e y de diversa naturaleza, pero que, a su vez, se encuentren enlazadas por una regla de correspondencia, una regla que permita establecer que x está relacionada con y.

La forma de relacionarse de los conjuntos ayudará a determinar el tipo de relación que se presenta, si es reflexiva, simétrica, antisimétrica, asimétrica o transitiva.

Otro elemento importante que facilita la identificación de las propiedades de una relación, y que, a su vez, permite establecer la forma como se asocian los elementos es la matriz de relación, que es una matriz de tipo booleano donde se representa qué elementos se encuentran asociados y cuáles no.

# BIBLIOGRAFÍA

## SUGERIDA

| Autor | Capítulo | Páginas |
|---|---|---|
| Kolman | 4 | 101 – 166 |
| González | NA | Todas |

Licenciatura: Informática

# UNIDAD 4

# Teoría de grafos

SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

# OBJETIVO PARTICULAR

El alumno aprenderá los conceptos básicos relacionados con la teoría de los grafos.

# TEMARIO DETALLADO

## (14 horas)

4. Teoría de grafos
    - 4.1. Grafos, dígrafos
    - 4.2. Vértices, aristas (entradas, salidas, paralelas), rizos, valencia, longitud
    - 4.3. Matrices
        - 4.3.1. Matriz asociada a un grafo
        - 4.3.2. Matriz booleana, operaciones booleanas
        - 4.3.3. Investigación / discusión sobre aplicaciones (sudoku)
    - 4.4. Representación por computadora de relaciones y grafos dirigidos

# INTRODUCCIÓN

En la unidad anterior, se señaló que existe una forma abstracta de representar las relaciones, principalmente las binarias, que permiten establecer una regla de correspondencia para relacionar elementos de diversos conjuntos.

También se indicó que existen matrices de tipo booleano asociadas a estas relaciones.

A lo largo de la presente unidad, se revisará una tercera forma de representación de las relaciones: los grafos dirigidos o dígrafos, que, al igual que las matrices de relación, permiten obtener el conjunto relación a partir del grafo.

Se estudiarán las propiedades y características de los grafos, así como la forma de interpretarlos y su asociación con la matriz booleana.

Finalmente, se revisarán algunos ejemplos de aplicación de los grafos y su forma de representación en la computadora.

# 4.1. Grafos, dígrafos

Al estudiar las relaciones binarias, éstas invariantemente nos llevarán a su asociación con los grafos; de manera formal podemos decir:

> Un grafo dirigido o un dígrafo es un par ordenado D = (A, R) donde A es un conjunto finito y R es una relación binaria definida sobre A. Al conjunto A lo llamaremos conjunto de nodos o vértices de D. A los elementos de R los llamaremos arcos o aristas de dígrafo[33] y lo representaremos como: $G_R$

En forma similar a las matrices de relación vistas en el tema anterior, un grafo puede ser generado a partir de una relación o una relación puede ser conocida a partir de su grafo.

En el caso del dominio y la imagen de una relación, en el grafo representarán al extremo inicial y final de un arco, respectivamente.

A continuación, revisaremos los elementos que contienen a un grafo y algunos ejemplos.

[33] González Gutiérrez, Francisco J. (2004). *Apuntes de matemáticas discretas*. Universidad de Cádiz, España. Consultado el 22/07/2014 de: http://www2.uca.es/matematicas/Docencia/ESI/1710003/Apuntes/Leccion6.pdf.

# 4.2. Vértices, aristas (entradas, salidas, paralelas), rizos, valencia, longitud

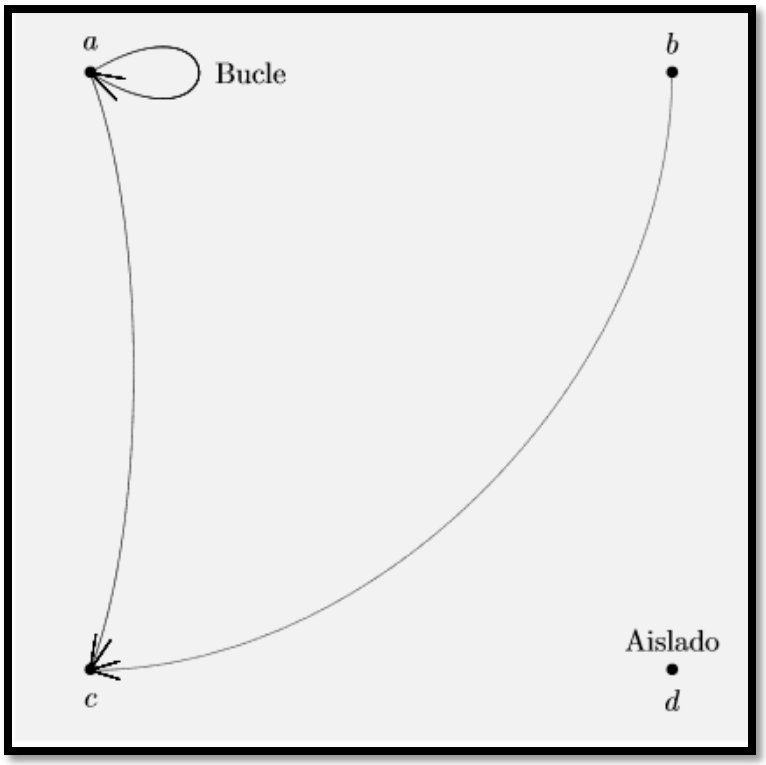
Como ya mencionamos en el punto anterior, los vértices representan a los elementos de un conjunto A asociado a una relación; pero vamos a profundizar un poco en este tema y los elementos generales que componen a un grafo.

Los elementos del conjunto A pueden ser tomados como puntos en un plano cartesiano, como en el caso de las funciones cuando vamos dando valores a la variable independiente x. Cuando encontremos una relación aRb realizaremos un trazo dirigido del punto a al punto b con las siguientes consideraciones:

- Al elemento *a* se le conocerá como el vértice inicial y al elemento b como vértice final de la arista (trazo).
- A la arista que une un punto consigo mismo se le conoce como bucle o rizo.
- A los elementos del conjunto A que no se encuentren asociados por medio de una relación se le denominará vértice asilado.
- Al número de aristas que llegan a un vértice se le denominará grado o valencia de entrada del vértice y se representará como $gr_e(a)$.
- Al número de aristas que salen de un vértice se le denominará grado o valencia de salida del vértice y se representará como $gr_s(a)$.
- La longitud o camino de un grafo, es el número de vértices que son recorridos mediante una secuencia de grafos.

Ejemplo:

Se tiene el conjunto A = {a, b, c, d} y una relación R = {(a, a), (a, c), (b, c)}, el grafo asociado a dicha relación es el siguiente[34]:




En el grafo anterior podemos ver que el elemento correspondiente a (a, a) tiene un bucle asociado a ese elemento, el elemento (a, c) tiene una arista que va de a hacia c, representado por una flecha con cuerpo de arco, el tercer elemento (b, c) también muestra una arista que sale de b y llega a c, mientras que el elemento d al no tener ninguna relación se etiqueta como aislado. Es importante resaltar que todos los elementos que componen al conjunto A deben de estar representados en el grafo, ya que son los vértices de donde parten las aristas.

[34] González Gutiérrez, Francisco J. (2004). *Apuntes de matemáticas discretas*. Universidad de Cádiz, España. Consultado el 22/07/2014 de: http://www2.uca.es/matematicas/Docencia/ESI/1710003/Apuntes/Leccion6.pdf.

Con respecto a los grados de los nodos:

El grado de salida de a es 2, $gr_s(a) = 2$, ya que podemos observar que del vértice a parten dos aristas.

El grado de entrada de a es de 1, $gr_e(a) = 1$, ya que podemos ver que solamente entra una arista en a (la del bucle).

Para el resto de los nodos:

$gr_e(b) = 0$

$gr_s(b) = 1$

$gr_e(c) = 2$

$gr_s(c) = 0$

$gr_e(d) = 0$

$gr_s(d) = 0$

La longitud o camino del grafo del ejemplo es de 2 de (a, c), ya que hay una arista de bucle en a y, posteriormente, otra de a hacia c.

# 4.3. Matrices

## 4.3.1. Matriz asociada a un grafo

La matriz asociada a un grafo es la misma que la matriz de relación vista en la unidad anterior, se trata de una matriz de tipo booleano donde se coloca un 1 cuando se encuentre una relación de un vértice a otro y un cero para los elementos aislados.

De nuestro ejemplo anterior, la matriz asociada al grafo es la siguiente:

$$Mr = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

## 4.3.2. Matriz booleana, operaciones booleanas

Una matriz booleana es una matriz que solamente puede contener dos valores, 0 y 1.

Al contener valores binarios, las operaciones que pueden realizarse con la matriz deben de considerar las propiedades de la suma y multiplicación booleana.

La suma booleana es representada mediante la tabla de verdad de la operación lógica "*or*" y la forma de operar es la siguiente:

| a | b | a + b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

La multiplicación booleana se realiza mediante la operación lógica "y" o "*and*" cuya equivalencia es la siguiente:

| a | b | a . b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Ahora bien, las operaciones que pueden realizarse con las matrices booleanas son la unión, la intersección y el producto booleano, considerando que las matrices surgen de las relaciones y éstas del producto cartesiano entre dos conjuntos.

La operación de unión se realiza básicamente como la suma de matrices; al igual que en la suma, se operan los elementos en cada posición de la matriz, sumando cada elemento de tal forma que el resultado será 1 si cualquiera de los 2 valores es 1, y será 0 solamente si ambos valores son cero.

Ejemplo:

Sean

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \end{bmatrix}$$

$$\boldsymbol{A} \cup \boldsymbol{B} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \end{bmatrix}$$

La segunda operación es la intersección o conjunción, y ésta se realiza de forma similar a la unión, con la diferencia que la operación que se realiza es la *and*, de tal forma que si alguno de los elementos de las matrices es 0 el resultado será 0, y 1 solamente cuando ambos elementos sean 1.

Ejemplo:

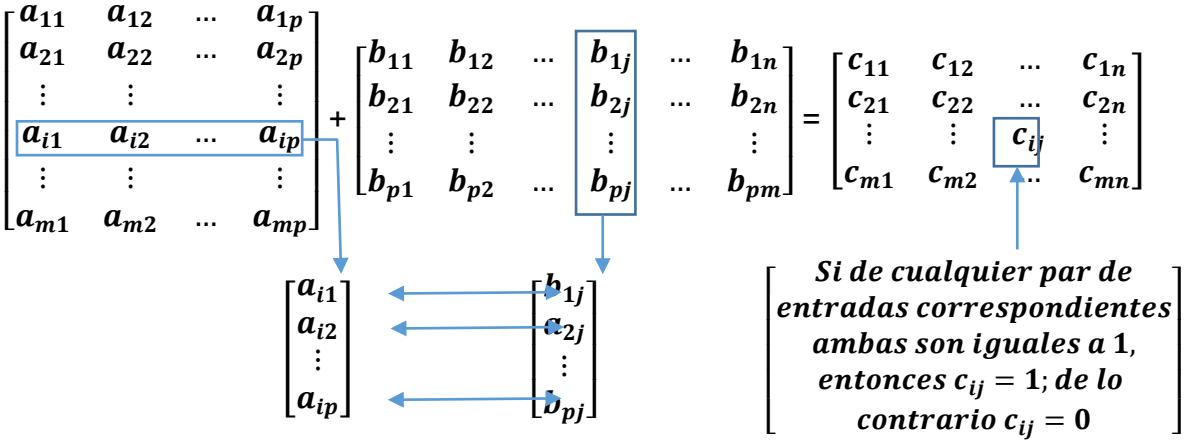
Sean

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

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

Finalmente, se tiene el producto booleano, representado por $\otimes$.

El producto booleano se realiza de la misma manera que la multiplicación entre matrices, donde ambas deben de ser conformables para poder realizar la operación. En el caso de la multiplicación tradicional, se obtenía el primer elemento de la matriz resultante al multiplicar el primer renglón de la matriz uno por la primer columna de la matriz dos, sumando el resultado de cada elemento; en este caso, las multiplicaciones seguirán a la operación "y" y las sumas a la operación "o".




**Representación gráfica de la multiplicación de matrices booleanas**[35]

[35] Ejemplos e imagen tomados de: Kolman, Bernard *et al.* (1996). *Estructuras de matemáticas discretas para la computación* (3ra edición). México: Prentice Hall, pp. 101- 104.

Ejemplo:

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

$$A \otimes B = \begin{bmatrix} (1 \wedge 1) \vee (0 \wedge 1) \vee (1 \wedge 0) \vee (1 \wedge 1) & (1 \wedge 1) \vee (0 \wedge 0) \vee (1 \wedge 0) \vee (1 \wedge 1) & (1 \wedge 0) \vee (0 \wedge 1) \vee (1 \wedge 1) \vee (1 \wedge 0) \\ (0 \wedge 1) \vee (1 \wedge 1) \vee (1 \wedge 0) \vee (0 \wedge 1) & (0 \wedge 1) \vee (1 \wedge 0) \vee (1 \wedge 0) \vee (0 \wedge 1) & (0 \wedge 0) \vee (1 \wedge 1) \vee (1 \wedge 1) \vee (0 \wedge 0) \\ (1 \wedge 1) \vee (0 \wedge 1) \vee (0 \wedge 0) \vee (1 \wedge 1) & (1 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 0) \vee (1 \wedge 1) & (1 \wedge 0) \vee (0 \wedge 1) \vee (0 \wedge 1) \vee (1 \wedge 0) \end{bmatrix}$$

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

## 4.3.3. Investigación / discusión sobre aplicaciones (sudoku)

Las aplicaciones de las matrices booleanas son muy amplias, por ello se introducen en esta unidad algunos anexos con ejemplos de aplicación tanto de los grafos como de las matrices booleanas en diversas áreas.

En el documento "Representación de redes sociales mediante matrices", el autor, Robert A. Hannemen, de la Universidad de California Riverside, describe la aplicación de matrices booleanas en el análisis de redes sociales. El documento lo puedes consultar al final de la unidad.

En el artículo "sudoku" se describe el empleo de las matrices booleanas para resolver el juego sudoku; dentro del mismo veremos la metodología y el pseudocódigo que resuelve el problema. El artículo pertenece al taller de computación de la Universidad Simón Bolívar, de Venezuela, del profesor Ernesto Hernández-Novich.

# 4.4. Representación por computadora de relaciones y grafos dirigidos

Los grafos y, por tanto, las relaciones, pueden representarse dentro de la computadora en formas distintas empleando varias estructuras de datos como las listas, los arreglos, las pilas y las colas.

A continuación, te presentaremos algunos artículos que muestran diversos ejemplos donde se representan los arreglos y los grafos.

El primer blog, dedicado a las matemáticas discretas, muestra diversos ejemplos de representación de grafos por computadora. El blog pertenece a Claudio Cifuentes y puedes consultarlo en el siguiente sitio: http://teoriadegrafos.blogspot.mx/2007/03/representaciones-de-grafos.html.

El anexo "grafos", nos da un repaso a mayor profundidad de los grafos y, adicionalmente, nos muestra diversos algoritmos para la representación de éstos en la computadora, su autora es Lipschutz, Seymour, del Instituto Tecnológico de Nuevo Laredo.

# RESUMEN

Los grafos dirigidos o dígrafos son la representación gráfica de las relaciones, se construyen a partir del conjunto principal y de la colección de pares ordenados en el conjunto relación, siendo los elementos del conjunto los vértices a partir de los cuales se realiza el trazo de los grafos.

Los grafos, al igual que las relaciones, tienen asociada una matriz de tipo booleana, con la cual se pueden representar los nodos y las aristas, que representan las relaciones entre vértices.

Las matrices booleanas, al igual que las matrices regulares, pueden ser manipuladas a través de diversas operaciones de tipo booleano, basadas principalmente en el álgebra booleana, tomando sus operaciones bases en los operadores “y” (*and*) como la base de la multiplicación y “o” *(or*) como base de la suma.

Las matrices booleanas, y por tanto las relaciones y grafos, tienen diversas aplicaciones en diversos campos, a lo largo de la unidad se vieron algunos ejemplos aplicados en el análisis de redes sociales y del juego sudoku.

La representación en computadora de los grafos, se basa principalmente en la implementación de su matriz asociada, empleando diversas estructuras de datos.

# BIBLIOGRAFÍA

## SUGERIDA

| Autor | Capítulo | Páginas |
|---|---|---|
| Kolman | 4 | 101 - 166 |
| González | NA | Todas |

Licenciatura: Informática

# UNIDAD 5

# Árboles

SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

# OBJETIVO PARTICULAR

El alumno identificará los tipos, características y recorridos de los árboles.

# TEMARIO DETALLADO

## (12 horas)

5. Árboles

- 5.1. Tipos, características, recorridos
- 5.2. Mínimos
- 5.3. Arraigados
- 5.4. Binarios
- 5.5. Representación por computadora de árboles

# INTRODUCCIÓN

Hasta el momento, se ha estudiado que a partir de los conjuntos podemos generar relaciones entre sus elementos, y que a partir de las relaciones se asocian matrices y grafos a ellas.

El siguiente paso, en el desarrollo de estructuras discretas asociadas a las relaciones, son los árboles, una nueva estructura discreta que permite realizar relaciones más complejas.

A lo largo de la presente unidad se estudiará el concepto de árboles, se revisarán sus características y terminología.

Posteriormente, se revisarán diversos tipos de árboles como el mínimo, el arraigado y el binario, que es el más empleado.

Finalmente, se estudiará las estructuras de datos y algunos algoritmos que ayuden a la representación de los árboles en la computadora.

# 5.1. Tipos, características, recorridos

Una de las estructuras discretas más completas y complejas que podemos encontrar son los árboles.

Un árbol es una estructura compuesta básicamente por relaciones, donde cada elemento de él se relaciona de alguna forma con el elemento principal o nodo raíz, como se le conoce, de tal forma que a partir de dicho nodo raíz pueden nacer diversos nodos denominados hojas.

Una definición formal de un árbol es la siguiente:

> Sea A un conjunto y T una relación en A. T es un árbol si existe un vértice en $v_0$ en A con la propiedad de que existe una única trayectoria en T de $v_0$ hacia cualquier otro vértice en A, pero no existe una trayectoria de $v_0$ a $v_0$.[36]

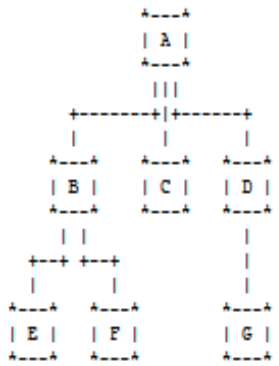
Los árboles de manera general, constan de un nodo o vértice inicial a partir del cual nacen varias relaciones con otros nodos o vértices, a las líneas o trazos que enlazan un nodo con otro se les denomina ramas y a los nodos que conectan con ellas, hojas.

[36] Kolman, Bernard *et al.* (1996). *Estructuras de matemáticas discretas para la computación* (3ra edición). México: Prentice Hall, p. 286.

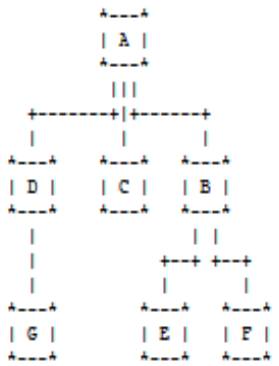
***Terminología:***

| | |
|---|---|
| ***Raíz.*** | Nodo o vértice principal a partir del cual nace el resto de los nodos. |
| ***Nodo padre***. | Se trata de un nodo a partir del cual nacen más nodos. |
| ***Nodo hijo***. | Nodo que nace a partir de un nodo padre. |
| ***Descendientes***. | Nodos hijos nacidos de otros nodos hijos y de un nodo padre. |
| ***Ascendientes***. | Nodos que son los padres de los nodos inferiores o hijos. |
| ***Hermanos***. | Nodos que se encuentran al mismo nivel y que nacen de un mismo nodo padre. |
| ***Hojas***. | Nodos que no tienen nodos descendientes o hijos. |
| ***Camino.*** | Se trata de la secuencia de nodos que parten desde el nodo raíz hasta un nodo hoja; generalmente, estos caminos son únicos. |
| ***Altura o profundidad***. | Se trata de la longitud del camino más largo desde el nodo raíz hasta el nodo hoja que se localiza en el nivel más bajo del árbol. |
| ***Subárbol***. | Todos los nodos que tienen descendencia forman estructuras de ramificación denominadas subárboles, cada nodo donde inicia el subárbol se denomina raíz del mismo. |
| ***Peso del árbol.*** | Es el número de nodos terminales. |
| ***Grado de un Nodo.*** | Es el número de subárboles que tiene un nodo. Los nodos hoja tienen grado cero. |
| ***Grado de un árbol.*** | El grado máximo de todos los nodos del árbol. |

Algunas formas de representación de los árboles son las siguientes:

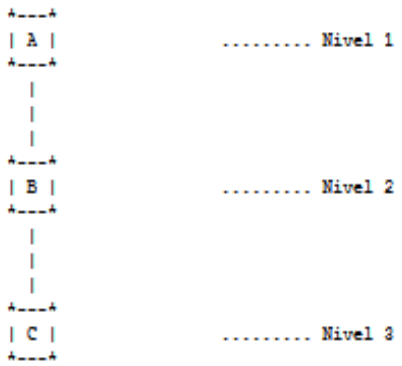



Arbol 1

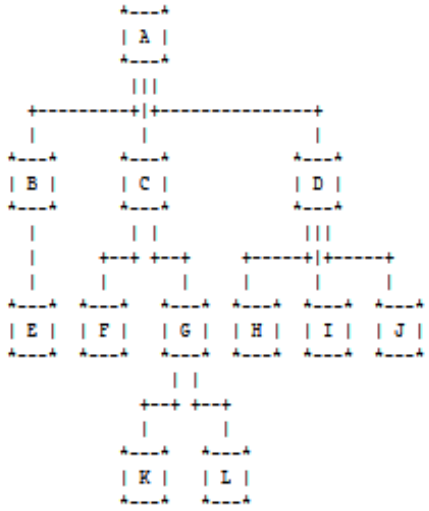



Arbol 2

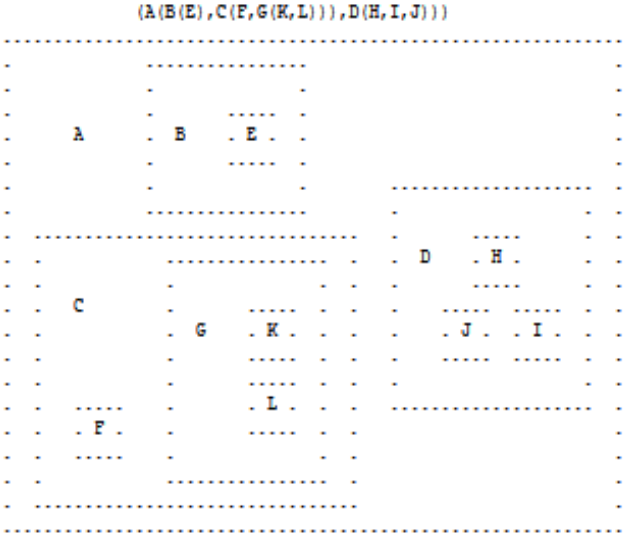

Arbol 3



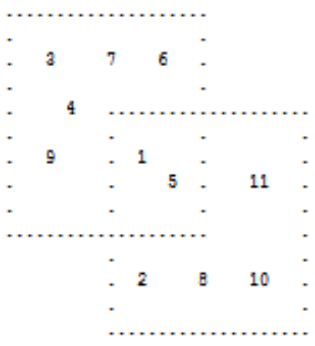

Arbol 4




Arbol 5




Arbol 5 - Conjuntos Disjuntos




Conjuntos No Disjuntos

## Ejemplos de representación de árboles[37]

Dentro de los tipos de árboles que podemos encontrar tenemos:

***Árbol lleno.***

- Se trata de un árbol donde casi todos sus nodos tienen descendencia y los nodos hojas tienen el mismo nivel o profundidad.

***Árbol vacío.***

- Es un árbol que no tiene ningún nodo descendiente, generalmente solo tienen el nodo raíz.

***Árbol completo***.

- Es un árbol donde todos sus nodos pueden numerarse y seguir dicha secuencia sin que falte o se salte algún número, por lógica, todo árbol completo es un árbol lleno.

***Árbol binario***.

- *Es un árbol cuyos nodos no pueden tener más de dos subárboles.* "En un árbol binario, cada nodo puede tener cero, uno o dos hijos. Se conocen como el nodo de la izquierda y el nodo de la derecha"[37].

Los árboles más comunes empleados en informática son los binarios, por lo que los retomaremos más a profundidad más adelante.

[37] Tomada de: Testa, Orestes. (2005). *Estructuras de datos. Árboles y fundamentos.* Universidad Nacional del Rosario, Argentina. Consultado el 24/07/2014 de: http://www.fceia.unr.edu.ar/estruc/2005/arbofund.htm.

[38] *Estructura de datos y árboles*. Universidad Nacional de Jujuy. Argentina. Consultado el 24/07/2014 de: http://www.fi.unju.edu.ar/materias/materia/IIEDD/document/Publicaciones/Estructura_de_Datos_y_Arboles.pdf?cidReq=IIEDD.

En cuanto a lo que a los recorridos se refiere, un recorrido es la forma en que procesamos o visitamos los nodos de un árbol, partiendo desde el nodo raíz.

En el caso de los árboles binarios tenemos 3 recorridos básicos:

Pre orden. Se comienza desde la raíz, visitando primero el subárbol izquierdo regresando a la raíz y luego el subárbol derecho.

En orden. Primero se visita el subárbol izquierdo, luego la raíz y el subárbol derecho.

Post orden. Se visita primero el subárbol izquierdo, después el derecho y finalmente la raíz.

## 5.2. Mínimos

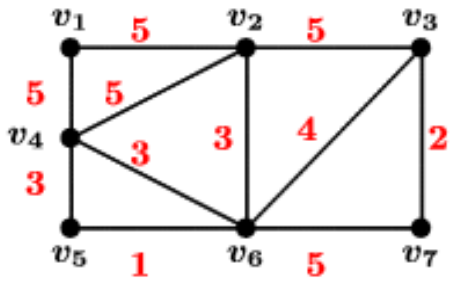
Para poder entender lo que es un árbol mínimo o de expansión mínima debemos entender primero qué es un árbol generador o de expansión.

"Llamaremos árbol generador o de expansión de un grafo conexo a cualquier subgrafo con el mismo número de vértices que sea árbol"[39].

Entendamos la definición, un grafo es la representación gráfica de una relación, y un árbol es un grafo de varias relaciones sobre un conjunto base, el árbol generador

[39] Abia Vian, José A. (2008). *Grafos no dirigidos acíclicos – Árboles.* Universidad de Valladolid, España. Consultado el 24/07/2014 de: http://www.ma.uva.es/~antonio/Industriales/Apuntes_07-08/LabM/Grafos_2008-4.pdf.

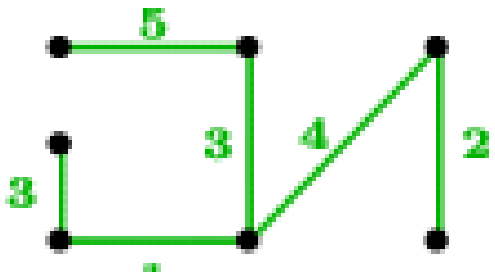
es un grafo que conecta a todos los nodos (elementos de la relación) sin excepción, a lo que se le denomina grafo conexo.




**Ejemplo de grafo conexo o árbol generador[40]**

Ahora bien, todos los árboles tienen un peso, el cual se da por el número de nodos terminales, pero también es posible darle peso a las aristas o conexiones entre nodos, como podemos ver en la figura anterior, lo que se conoce como grafica de peso o grafo de peso.

Un grafo generador mínimo, es un árbol o grafo conexo con peso, que mantiene todas las conexiones entre todos los nodos, pero procurando que el peso del árbol sea mínimo.




**Ejemplo del grafo conexo anterior pero con peso mínimo (árbol mínimo)[41].**

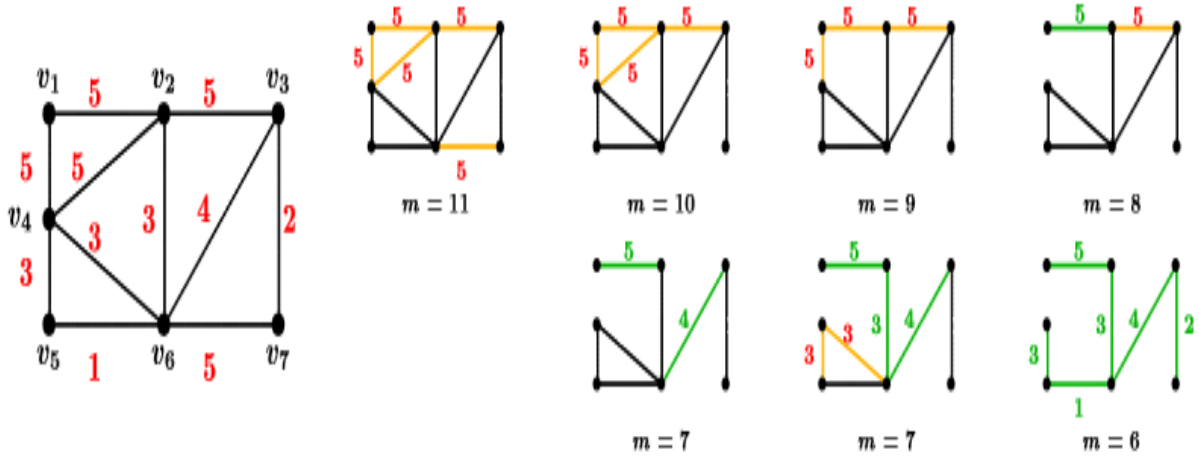
[40] Ídem.
[41] Ídem.

Para poder encontrar los árboles mínimos a partir de un árbol generador, se emplean algoritmos especializados como el caso de los algoritmos de Krustkal y Prim.

El algoritmo de Krustkal, es un algoritmo que elimina aristas del árbol generador o grafo conexo, hasta encontrar el árbol que conecte cada nodo con el peso mínimo.

El algoritmo de Krustkal toma en consideración los siguientes elementos:

- Si G es un grafo conexo de n vértices y m aristas, hay que quitar m-((n-1) aristas para que obtengamos las n-1 aristas del árbol mínimo.
- Se eliminan las aristas pertenecientes a ciclos o rizos, de tal forma que el grafo se mantenga conectado.
- Durante la búsqueda del árbol generador mínimo se deben eliminar aquellas aristas que tengan más peso[42].
-

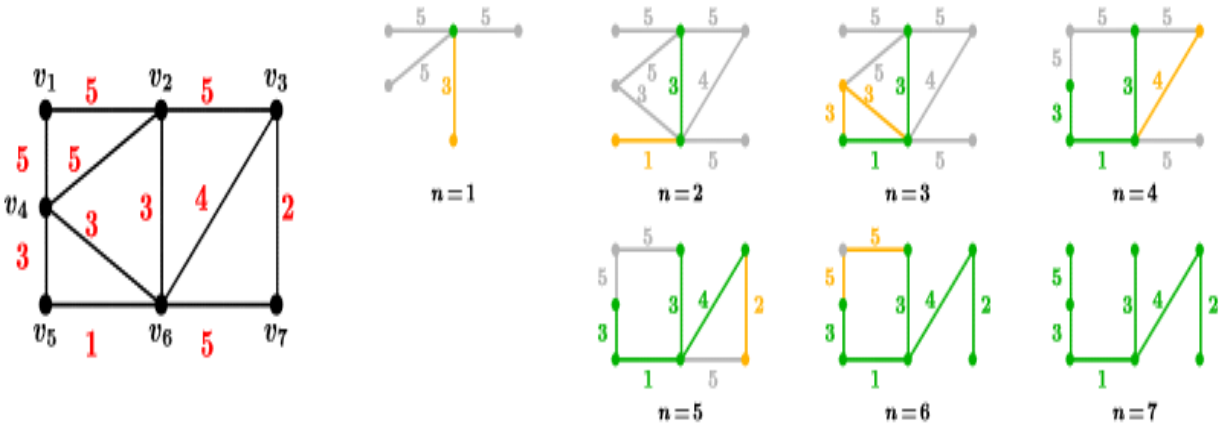



**Ejemplo de aplicación del algoritmo de Krustkal[43]**

[42] Ídem.
[43] Abia Vian, José A. (2008). *Grafos no dirigidos acíclicos – Árboles.* Universidad de Valladolid, España. Consultado el 24/07/2014 de: http://www.ma.uva.es/~antonio/Industriales/Apuntes_07-08/LabM/Grafos_2008-4.pdf.

El algoritmo de Prim parte de la generación de varios árboles generadores mínimos a partir de un nodo o vértice inicial.

El proceso del algoritmo va agregando hojas al nodo inicial hasta llegar al árbol generador mínimo deseado, al agregar las hojas de una en una mantiene la conexión entre nodos mínima, a su vez cada nueva conexión va seleccionando la de menor peso hasta construir el árbol final.

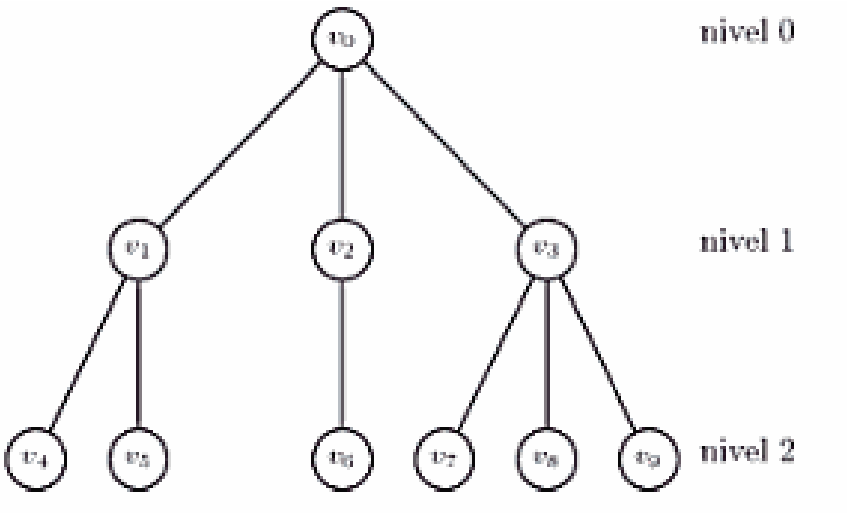



**Ejemplo de la aplicación del algoritmo de Prim[44].**

[44] Ídem.

# 5.3. Arraigados

Un árbol arraigado es un árbol cuya trayectoria entre el vértice o nodo raíz y cualquier otro nodo, es única y a su nivel de trayectoria se le denomina nivel del vértice enésimo[45].




Ejemplo de árbol arraigado[46].

En la figura anterior podemos ver que cada una de las trayectorias desde el nodo raíz hacia cualquiera de los nodos descendientes es única, es decir, no hay forma de acceder a ellos a través de ninguna otra trayectoria dentro del árbol. El nivel de cada trayectoria se determina mediante la profundidad de cada nodo; así, por ejemplo, la trayectoria desde $v_0$ hasta $v_4$ es de nivel 2, ya que el nodo $v_4$ se localiza en el nivel 2 del árbol.

[45] Caicedo Alfredo *et al.*, 2010: 85.
[46] Ídem.

# 5.4. Binarios

Como ya mencionamos, los árboles binarios solamente pueden tener como máximo 2 nodos a partir de un nodo raíz o de cada nodo padre y su recorrido puede darse en orden, postorden y preorden.

Dentro de los árboles binarios encontramos a los que son estrictamente binarios, éstos se caracterizan porque todos los nodos padres del árbol tienen un subárbol izquierdo y derecho.

En un árbol binario se pueden realizar las siguientes operaciones.

***Inserción***

En el caso de la inserción, es posible agregar nodos nuevos a un árbol siempre y cuando se observen las reglas de no tener más de dos nodos por cada nodo padre, de tal forma que el árbol mantenga su estructura binaria original.

***Borrado***

El borrado de árboles binarios presenta 3 casos.

1. Borrado de nodos hoja. En este caso solamente se realiza el borrado directo del nodo, al no contener hijos su eliminación es inmediata.
2. Borrado de nodos con un hijo. En este caso lo que se hace es simplemente realizar un enlace entre el nodo antecesor al nodo que se desea borrar con el nodo hijo, al contener solamente un hijo el enlace se puede hacer directamente sin problemas.

3. Borrado de un nodo con dos hijos. Este es el caso más complejo ya que tenemos dos trayectorias a partir del nodo que deseamos borrar; para poder borrar este nodo se puede sustituir el nodo por aquel de mayor peso entre los dos nodos hijo, con ello mantenemos el equilibrio del árbol y su forma binaria.

Otra forma de operación es el recorrido del árbol, que ya vimos con anterioridad.

## 5.5. Representación por computadora de árboles

La representación de los árboles en la computadora conlleva el empleo de varias estructuras de datos diferentes como listas, arreglos, pilas, colas, etc. A continuación, te presentamos "Un ejemplo de Árboles binarios y su programación", de la Universidad de las Palmas de Gran Canaria, España, donde podrás observar su implementación en un programa de computadora.

# RESUMEN

Los árboles son estructuras de matemáticas discretas que nacen a partir de las relaciones, siendo una representación de varias relaciones a diversos niveles con los elementos del conjunto base.

Los árboles, dependiendo de su estructura, pueden ir desde los árboles vacíos donde solamente se tiene el nodo inicial o raíz, hasta árboles que se consideran completos, donde todos sus nodos hoja están a un mismo nivel y los nodos padre todos tienen nodos hijos, que forman los subárboles izquierdo y derecho en los árboles binarios.

Existen árboles o grafos, denominados de expansión, que son aquellos donde todos sus nodos se encuentran interconectados, dentro de este tipo particular encontramos el árbol de expansión mínima, que es aquel árbol que mantiene conectados a todos sus nodos, pero con un peso mínimo.

Los árboles son estructuras a través de las cuales podemos incrementar o reducir, dependiendo de lo que se desee realizar.

En el ámbito informático, se requiere de combinar diferentes estructuras de datos y funciones recursivas para poder aplicarlas y darles utilidad.

# BIBLIOGRAFÍA

## SUGERIDA

| Autor | Capítulo | Páginas |
|---|---|---|
| Kolman | 8 | 286 - 328 |
| Caicedo | 7, 8 | 83-115 |

Licenciatura: Informática

# UNIDAD 6

# Prácticas en laboratorio

SUAyED UNAM
Sistema Universidad Abierta y Educación a Distancia

Plan 2012 2016 actualizado

# OBJETIVO PARTICULAR

El alumno resolverá problemas de matemáticas discretas utilizando software.

# TEMARIO DETALLADO

## (10 horas)

6. Prácticas en laboratorio

6.1. Prácticas en laboratorio

# INTRODUCCIÓN

Las estructuras de datos discretas han sido una gran herramienta para la simplificación de problemas en el ámbito de la informática, ayudan a reducir tiempo, reciclar código y a mejorar la eficiencia de los programas informáticos.

A lo largo de la presente unidad, veremos diversos casos en donde se emplean las estructuras discretas vistas en las unidades anteriores, desde las funciones recursivas hasta las estructuras de datos más complejas como son los árboles.

# 6.1. Prácticas en laboratorio

Las estructuras de matemáticas discretas que hemos visto hasta el momento tienen un sinfín de aplicaciones. En unidades anteriores vimos los casos de la aplicación de las matrices binarias en el análisis de redes sociales y de solución de procesos de razonamiento como en el caso del juego de sudoku.

La siguiente presentación ejemplifica "El empleo de la estructura de árbol en la solución de problemas asociados a la minería de datos", una de las disciplinas de la informática que han empezado a cobrar una alta importancia en el mundo empresarial debido al valor que tiene la información para estas empresas.

La presentación es de Juan A. Botía de la Universidad de Murcia, España, y muestra la forma en que las estructuras de árboles se emplean para el tratamiento de la información de forma inteligente.

El siguiente caso trata del "Empleo de las estructuras de los árboles en la toma de decisiones para tratar de predecir la deserción escolar". Fue desarrollado por Sergio Valero, Orea Alejandro Salvador Vargas y Marcela García Alonso de la Universidad Tecnológica de Izúcar de Matamoros, Puebla.

Revisa la presentación del "Empleo de los grafos en el desarrollo de automatizaciones industriales, mediante el empleo de las redes de Petri", que ayudan a relacionar lugares y transiciones en los procesos industriales y que son ampliamente empleadas en el área de telecomunicaciones. El caso es presentado por el departamento de Ingeniería Electrónica, de Sistemas Informáticos y Automática de la Universidad de Huelva, España.

# RESUMEN

Dentro del ámbito informático, el empleo de las estructuras discretas como grafos, relaciones, matrices binarias y árboles es muy amplio.

Disciplinas como la minería de datos, bases de datos, telecomunicaciones, ingeniería de software entre otras, emplean estas estructuras para poder tomar decisiones, optimizar procesos, diseñar sistemas expertos, etc. que ayudan a hacer más eficientes los sistemas de información.

En la minería de datos, el empleo de las estructuras de árboles y de las relaciones es esencial en el reconocimiento de patrones y predicciones; en base de datos, las operaciones con conjuntos y las funciones recursivas son esenciales para la programación de las bases de datos en sus procedimientos almacenados y funciones.

El empleo de matrices binarias para analizar las relaciones en las redes sociales es indispensable para el análisis de comportamiento y predicción dentro de ellas.

Como podemos ver, las aplicaciones son muy amplias, al igual que su potencial.

# BIBLIOGRAFÍA

## SUGERIDA

| Autor | Capítulo | Páginas |
|---|---|---|
| Kolaman | Todos | N/A |
| Caicedo | Todos | N/A |

# REFERENCIAS BIBLIOGRÁFICAS

## BIBLIOGRAFÍA SUGERIDA

Caicedo Barrero, A., Wagner de García, G. y Méndez Parra, R. (2010). *Introducción a la teoría de grafos.* Quindío: Universidad de Quindío.

Cornell, Gary (1994). *Manual de visual basic 3 para Windows*. Madrid: McGraw Hill.

Espinosa Herrera, E. y Navarrete, C. (2009). *Cálculo diferencial.* Hong Kong: Universidad Autónoma Metropolitana & Editorial Reverté.

González Gutiérrez, F. (2004). *Apuntes de matemáticas discretas*. Universidad de Cádiz, España. Consultado el 22/07/2014 de: http://www2.uca.es/matematicas/Docencia/ESI/1710003/Apuntes/Leccion6.pdf.

Kolman, B. *et al.* (1996). *Estructuras de matemáticas discretas para la computación.* (3ª edición). México: Prentice Hall.

Solar, E., Speziale, L. (1991). *Algebra lineal.* (2ª edición). México: Limusa/ UNAM/ Facultad de Ingeniería.

Solar, E, Speziale, L. (1998). *Algebra I.* (3ª edición). México: Limusa/ UNAM/ Facultad de Ingeniería.

Swokowski, E. (1971). *Algebra universitaria.* (3ª impresión). México: C.E.C.S.A.

# BIBLIOGRAFÍA BÁSICA

Cheney, W. (2011*). Métodos numéricos y computación.* (6ª edición) México: Cengage Learning.

Epp, S. (2012). *Matemáticas discretas con aplicaciones.* (4ª edición) México: Cengage Learning.

Jiménez, J. A. M. (2015). *Matemáticas para la computación.* (3ª edición) México: Alfaomega.

Rosen, K. H. (2012). *Matemáticas discretas y sus aplicaciones.* (7ª edición) Nueva York: McGraw Hill.

Villalpando, j. f. (2014). *Matemáticas discretas.* México: Alfaomega.

# BIBLIOGRAFÍA COMPLEMENTARIA

Espinosa, A. R. (2010*). Matemáticas discretas*. México: Alfaomega.

Johnsonbaugh, R. (2009). *Matemáticas discretas.* (7ª ed) México: Pearson Educación.

Lipschutz, S. (2009). *Matemáticas discretas.* (3ª ed) México: McGraw Hill.

Miranda, F. E. (2010). *Matemáticas discretas.* México: UNAM, Facultad de Ciencias.

# FUENTES ELECTRÓNICAS

Abia Vian, José A. (2008). *Grafos no dirigidos acíclicos – Árboles.* Universidad de Valladolid, España. Consultado el 24/07/2014 de: http://www.ma.uva.es/~antonio/Industriales/Apuntes_07-08/LabM/Grafos_2008-4.pdf.

Becerra E. José M. *Lógica matemática*. Consultado el 18/07/2014 de: http://www.fca.unam.mx/docs/apuntes_matematicas/36.%20logica%20matematica.pdf.

Bessy. *Pseudocódigo*. Consultado el 18/07/2014 de: http://20111003091.blogspot.com/2011/07/pseudocodigo.html.

Cid, Eva. Godino, Juan. Batanero, Carmen. *Sistemas numéricos y su didáctica para maestros.* Consultado el 27/03/2016 de: http://www.ugr.es/~jgodino/edumat-maestros/manual/2_Sistemas_numericos.pdf.

Contreras Moreira, Bruno. *Subrutinas y pasos de parámetros.* Consultado el 20/07/2014 de: http://www.ccg.unam.mx/~contrera/bioinfoPerl/node42.html. Actualmente disponible en: https://eead-csic-compbio.github.io/perl_bioinformatica/node41.html

Eaton, John. *GNU Octave*. Consultado el 20/07/2014 de: https://www.gnu.org/software/octave/doc/interpreter/The-do_002duntil-Statement.html.

Eaton, John. *GNU Octave. Sentencia Switch.* Consultado el 20/07/2014 de: https://www.gnu.org/software/octave/doc/interpreter/The-switch-Statement.html#The-switch-Statement.

*Estructura de datos y árboles***.** Universidad Nacional de Jujuy. Argentina. Consultado el 24/07/2014 de:

http://www.fi.unju.edu.ar/materias/materia/IIEDD/document/Publicaciones/Estructura_de_Datos_y_Arboles.pdf?cidReq=IIEDD.

García, Gladys. *Estructuras de datos, estructuras lineales o arreglos. Universidad Autónoma de puebla*. Consultado el 18/07/2014 de: http://www.uap.edu.pe/pregrado1/02/trabajos/02119/pw_alas/cap4_t_arreglos.htm.

Gómez, Rosa M. *Subprogramas: funciones y subrutinas.* Universidad de la Laguna, España. Consultado el 20/07/2014 de: http://rgomez.webs.ull.es/ARCHIVOSFMAT0708/clase3_0708.pdf.

González Gutiérrez, Francisco J. (2004*). Apuntes de matemáticas discretas*. Universidad de Cádiz, España. Consultado el 22/07/2014 de: http://www2.uca.es/matematicas/Docencia/ESI/1710003/Apuntes/Leccion6.pdf.

Harvey, Triana. *Eliminación de Gauss-Jordan en C.* Consultado el 20/07/2014 de: http://vexpert.mvps.org/articles/GJE.htm

*Manual de programación en C*: *Sentencias de control de flujo*. Grupo The neowriter. Consultado el 20/07/2014 de: http://whitehathacking.wordpress.com/2010/11/16/manual-de-programacion-en-c-sentencias-de-control-de-flujo.

Mercado, William. *Diagrama de flujo. Universidad experimental de Guyana.* Consultado el 18/07/2014 de: http://macabremoon0.tripod.com/id6.html.

Núñez, Ana M. *Relaciones y funciones*. Universidad de Mendoza, Argentina. Consultado el 21/07/2014 de: http://www.um.edu.ar/catedras/claroline/backends/download.php?url=L0FwdW50ZXMvUmVsYWNpb25lc195X2Z1bmNpb25lcy5wZGY%3D&cidReset=true&cidReq=TAYGAR.

*Programación fácil*. Consultado el 20/07/2014 de: https://www.programacionfacil.com/cpp/ciclo_do_while.html

Serna Pérez, Eduardo. *Recursividad.* Universidad Autónoma de Aguascalientes. Consultado el 20/07/2014 de: http://www.paginasprodigy.com/edserna/cursos/estddatos/notas/Unidad2.%20Recursividad.pdf.

Testa, Orestes. (2005). *Estructuras de datos. Árboles y fundamentos.* Universidad Nacional del Rosario, Argentina. Consultado el 24/07/2014 de: http://www.fceia.unr.edu.ar/estruc/2005/arbofund.htm.

Plan 2012

# 2016

actualizado