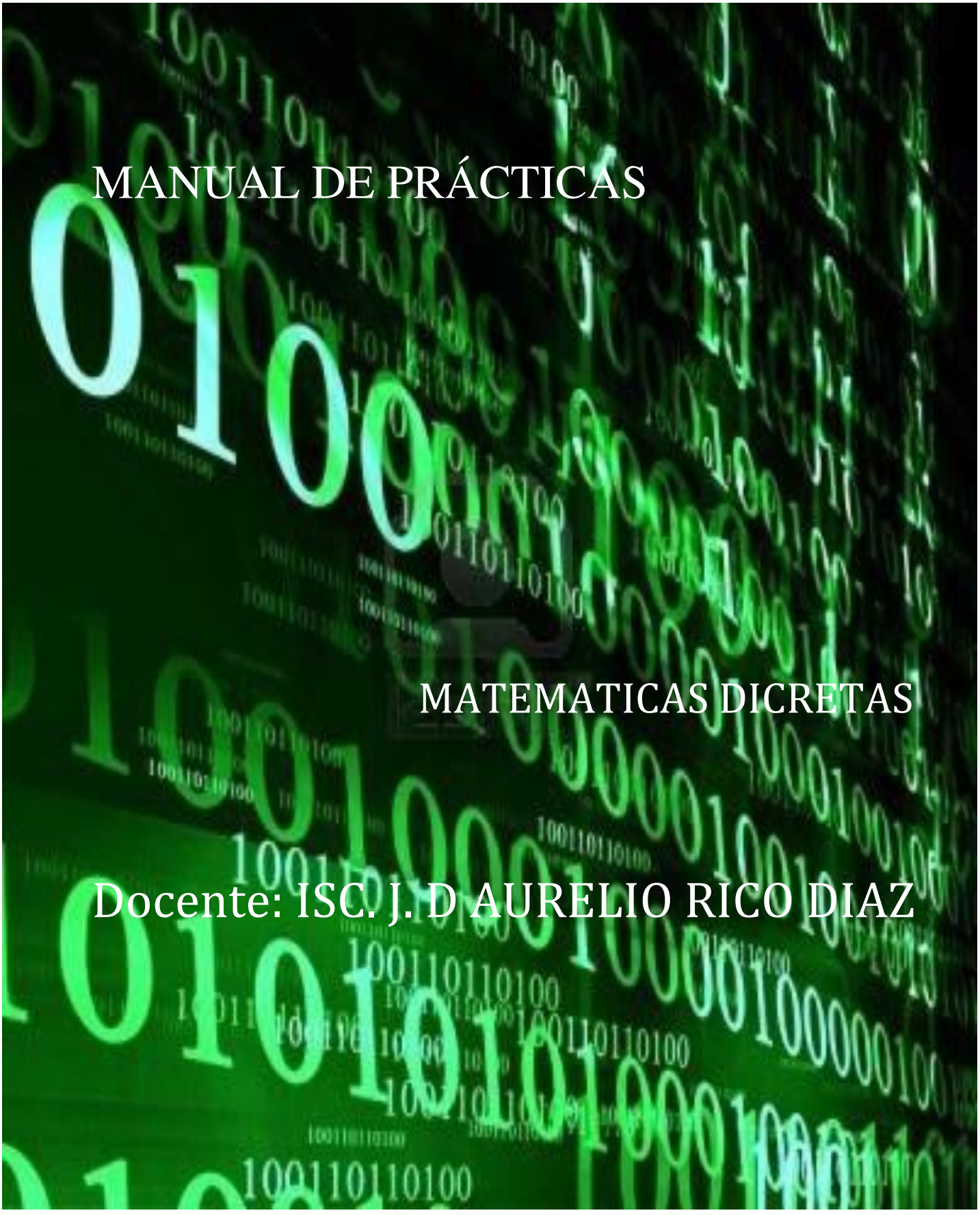
Manual DE Practicas MATE Discretas

Cálculo Diferencial (Instituto Tecnológico de Tijuana)



MANUAL DE PRÁCTICAS

MATEMATICAS DICRETAS

Docente: ISC. J. D AURELIO RICO DIAZ

Índice

INTRODUCCIÓN

En el presente manual de prácticas se describen una serie de 15 prácticas que ayudarán al estudiante a comprender mejor los conceptos vistos en clase y llevar la parte teórica a la práctica.

A lo largo de las seis unidades que tiene esta materia, se desarrollarán prácticas como es la conversión de sistemas de numeración, realización de tablas de verdad y análisis de proposiciones lógicas, las aplicaciones del álgebra booleana para el manejo de circuitos electrónicos integrados, el vínculo que hay en matemáticas discretas y las bases de datos, hasta llegar a la representación de grafos y árboles.

Práctica 1

CONVERSIONES NUMÉRICAS

Observaciones: Esta práctica incluye a la Práctica # 1 del temario de Matemáticas Discretas que dice "Elaborar a través de una hoja electrónica de cálculo un proceso para la conversión y realización de operaciones aritméticas básicas de cantidades en diferente base numérica".

1.- OBJETIVO

Conocer los diferentes algoritmos para convertir números entre diferentes sistemas de numeración.

2.- MARCO TEÓRICO

Sistema numérico.
Un sistema numérico está definido por la base que utiliza. La base es el número de símbolos diferentes necesarios para representar un número cualquiera, de los infinitos posibles en el sistema. Por ejemplo, el sistema decimal necesita diez símbolos diferentes o dígitos para representar un número y, es por tanto, un sistema numérico en base 10.

Los sistemas de numeración más comunes son:

Binario 0,1.
Octal 0, 1, 2, 3, 4, 5, 6, 7.
Decimal 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Hexadecimal 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Podemos cambiar de un sistema a otro siguiendo una serie de reglas (o algoritmos), las cuales se describirán a continuación:

Conversión de un numero decimal a binario
Para esta transformación es necesario tener en cuenta los pasos que mostraremos en el siguiente ejemplo: Transformemos el número 42 a número binario

1. Dividimos el número 42 entre 2
2. Dividimos el cociente obtenido por 2 y repetimos el mismo procedimiento hasta que el cociente sea 1.
3. El numero binario lo formamos tomando el primer dígito el ultimo cociente, seguidos por los residuos obtenidos en cada división, seleccionándolos de derecha a izquierda, como se muestra en el siguiente esquema.

Conversión de un número decimal fraccionario a binario
Para transformar un número decimal fraccionario a un numero binario debemos seguir los pasos que mostramos en el siguiente ejemplo: transformemos el numero 42,375.

La parte entera se transforma de igual forma que el ejemplo anterior.
La parte fraccionaria de la siguiente manera:

Multiplicamos por el numero 2 y tomamos la parte entera del producto que ira formando el numero binario correspondiente

Tomamos nuevamente la parte entera del producto, y la parte fraccionaria la multiplicamos sucesivamente por 2 hasta llegar a 0

Tomamos nuevamente la parte entera, y como la parte fraccionaria es 0, indica que se ha terminado el proceso. El numero binario correspondiente a la parte decimal será la unión de todas las partes enteras, tomadas de las multiplicaciones sucesivas realizadas durante el transcurso del proceso , en donde el primer dígito binario corresponde a la primera parte entera , el segundo dígito a la segunda parte entera, y así sucesivamente hasta llegar al último.

Luego tomamos el numero binario, correspondiente a la parte entera, y el numero binario, correspondiente a la parte fraccionaria y lo unimos en un solo numero binario correspondiente a el numero decimal.

Conversión de un número binario a un numero decimal
Para convertir un número binario a decimal, realizamos los siguientes pasos:

1. Tomamos los valores de posición correspondiente a las columnas donde aparezcan únicamente unos
2. Sumamos los valores de posición para identificar el número decimal equivalente

Conversión de un número decimal a octal
Para convertir un número en el sistema decimal al sistema de numeración Octal, debemos seguir los pasos que mostraremos en el siguiente ejemplo Convertir el número decimal 323.625 al sistema de numeración Octal

1. Se toma el numero entero y se divide entre 8 repetidamente hasta que el dividendo sea menor que el divisor, para colocar entonces el numero 0 y pasar el dividendo a formar el primer dígito del número equivalente en decimal
2. Se toma la parte fraccionaria del número decimal y la multiplicamos por 8 sucesivamente hasta que el producto no tenga números fraccionarios
3. Pasamos la parte entera del producto a formar el dígito correspondiente
4. Al igual que los demás sistemas, el número equivalente en el sistema decimal, está formado por la unión del número entero equivalente y el número fraccionario equivalente.

Conversión de un numero octal a binario
La ventaja principal del sistema de numeración Octal es la facilidad con que pueden realizarse la conversión entre un número binario y octal. A continuación mostraremos un ejercicio que ilustrará la teoría. Por medio de este tipo de conversiones, cualquier número Octal se convierte a binario de manera individual. En este ejemplo, mostramos claramente el equivalente 100 111 010 en binario de cada numero octal de forma individual

Conversión de un numero decimal a un numero hexadecimal
Convertir el número 250.25 a Hexadecimal

1. Se toma la parte entera y se divide sucesivamente por el número decimal 16 (base) hasta que el cociente sea 0
2. Los números enteros resultantes de los cocientes, pasarán a conformar el número hexadecimal correspondiente, teniendo en cuenta que el sistema de numeración hexadecimal posee solo 16 símbolos, donde los números del 10 hasta el 15 tienen símbolos alfabéticos que ya hemos explicado
3. La parte fraccionaria del número a convertir se multiplica por 16 (Base) sucesivamente hasta que el producto resultante no tenga parte fraccionaria
4. Al igual que en los sistemas anteriores, el número equivalente se forma, de la unión de los dos números equivalentes, tanto entero como fraccionario, separados por un punto que establece la diferencia entre ellos.

Conversión de un numero hexadecimal a un numero decimal
Como en los ejemplos anteriores este también nos ayudará a entender mejor este procedimiento: Convertir el numero hexadecimal 2B6 a su equivalente decimal.

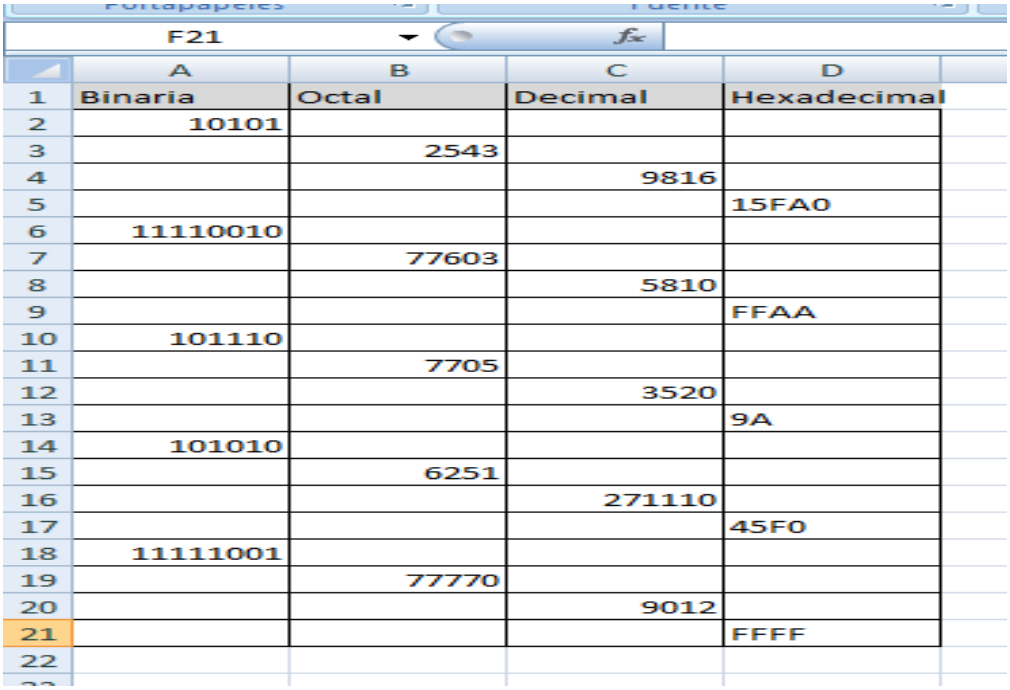
1. Multiplicamos el valor de posición de cada columna por el dígito hexadecimal correspondiente.
2. El resultado del número decimal equivalente se obtiene, sumando todos los productos obtenidos en el paso anterior.

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora
- Hoja de cálculo.
- Procesador de Palabras.

4.- COMPETENCIAS ESPECÍFICAS

En un archivo de hoja de cálculo electrónica (preferentemente Excel de Microsoft Office) colocar los siguientes datos y convertir las cantidades a las bases que hace falta:



F21

	A	B	C	D
1	Binaria	Octal	Decimal	Hexadecimal
2	10101			
3		2543		
4			9816	
5				15FA0
6	11110010			
7		77603		
8			5810	
9				FFAA
10	101110			
11		7705		
12			3520	
13				9A
14	101010			
15		6251		
16			271110	
17				45F0
18	11111001			
19		77770		
20			9012	
21				FFFF
22				

En Excel hay funciones para poder convertir de una base a otra base. Las funciones son las siguientes:

Función	Descripción
BIN.A.DEC(número)	Convierte el número binario a decimal
BIN.A.HEX(número)	Convierte el número binario a hexadecimal
BIN.A.OCT(número)	Convierte el número binario a octal
OCT.A.BIN(numero)	Convierte el número octal a binario
OCT.A.DEC(numero)	Convierte el número octal a decimal
OCT.A.HEX(numero)	Convierte el número octal a hexadecimal
DEC.A.BIN(numero)	Convierte el número decimal a binario
DEC.A.OCT(numero)	Convierte el número decimal a octal
DEC.A.HEX(numero)	Convierte el número decimal a hexadecimal
HEX.A.BIN(numero)	Convierte el número hexadecimal a binario
HEX.A.OCT(numero)	Convierte el número hexadecimal a octal
HEX.A.DEC(numero)	Convierte el número hexadecimal a decimal

Lo que debes hacer a continuación es colocar la función correspondiente a cada valor dado en la hoja de cálculo. Dando como resultado, la tabla llena con todas las conversiones entre los sistemas de numeración indicados.

Al finalizar, deberás realizar un reporte en un procesador de palabras sobre lo aprendido y los resultados obtenidos en ésta prácticas.

5. RESULTADOS

Al finalizar la práctica, el alumno debe ser capaz de convertir números de diferentes bases utilizando los algoritmos vistos en clases y con funciones ya predefinidas en alguna hoja de cálculo electrónica. El alumno deberá entregar en forma digital los resultados de convertir números entre diferentes bases numéricas con el uso de hojas electrónicas.

6. CONCLUSIONES

Como conclusión, ésta práctica les ayudará al alumno a resolver conversiones de una forma más rápida y sencilla, pero es muy importante que el alumno entienda primero cómo funcionan los algoritmos para la conversión de sistemas de numeración, para ello deberá hacer ejercicios para convertir de un sistema de numeración a otro, en hojas blancas o de su libreta. Todo lo que involucre esta práctica deberá ser reportado al docente

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 2

SOFTWARE DE CONVERSIÓN NUMÉRICA

Observaciones: Esta práctica incluye a la Práctica # 2 del temario de Matemáticas Discretas que dice "Buscar en Internet software que permita llevar a cabo ejercicios de conversión, operaciones matemáticas básicas (suma, resta, multiplicación y división) en diferentes sistemas numéricos, utilizarlo para resolver problemas planteados en clase".

1.- OBJETIVO

Conocer software especializado que permita realizar conversiones entre sistemas de numeración o que permita realizar las operaciones básicas en los sistemas numéricos.

2.- MARCO TEÓRICO

Sistema numérico.
Un sistema numérico está definido por la base que utiliza. La base es el número de símbolos diferentes necesarios para representar un número cualquiera, de los infinitos posibles en el sistema. Por ejemplo, el sistema decimal necesita diez símbolos diferentes o dígitos para representar un número y, es por tanto, un sistema numérico en base 10.

Los sistemas de numeración más comunes son:

Binario 0,1.
Octal 0, 1, 2, 3, 4, 5, 6, 7.
Decimal 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Hexadecimal 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Podemos cambiar de un sistema a otro siguiendo una serie de reglas (o algoritmos), las cuales se describirán a continuación:

Conversión de un numero decimal a binario
Para esta transformación es necesario tener en cuenta los pasos que mostraremos en el siguiente ejemplo: Transformemos el número 42 a número binario

1. Dividimos el número 42 entre 2
2. Dividimos el cociente obtenido por 2 y repetimos el mismo procedimiento hasta que el cociente sea 1.
3. El numero binario lo formamos tomando el primer dígito el ultimo cociente, seguidos por los residuos obtenidos en cada división, seleccionándolos de derecha a izquierda, como se muestra en el siguiente esquema.

Conversión de un número decimal fraccionario a binario
Para transformar un número decimal fraccionario a un numero binario debemos seguir los pasos que mostramos en el siguiente ejemplo: transformemos el numero 42,375.

La parte entera se transforma de igual forma que el ejemplo anterior.
La parte fraccionaria de la siguiente manera:

Multiplicamos por el numero 2 y tomamos la parte entera del producto que ira formando el numero binario correspondiente

Tomamos nuevamente la parte entera del producto, y la parte fraccionaria la multiplicamos sucesivamente por 2 hasta llegar a 0

Tomamos nuevamente la parte entera, y como la parte fraccionaria es 0, indica que se ha terminado el proceso. El numero binario correspondiente a la parte decimal será la unión de todas las partes enteras, tomadas de las multiplicaciones sucesivas realizadas durante el transcurso del proceso , en donde el primer dígito binario corresponde a la primera parte entera , el segundo dígito a la segunda parte entera, y así sucesivamente hasta llegar al último. Luego tomamos el numero binario, correspondiente a la parte entera, y el numero binario, correspondiente a la parte fraccionaria y lo unimos en un solo numero binario correspondiente a el numero decimal.

Conversión de un número binario a un numero decimal
Para convertir un número binario a decimal, realizamos los siguientes pasos:
1. Tomamos los valores de posición correspondiente a las columnas donde aparezcan únicamente unos
2. Sumamos los valores de posición para identificar el número decimal equivalente

Conversión de un número decimal a octal
Para convertir un número en el sistema decimal al sistema de numeración Octal, debemos seguir los pasos que mostraremos en el siguiente ejemplo Convertir el número decimal 323.625 al sistema de numeración Octal

1. Se toma el numero entero y se divide entre 8 repetidamente hasta que el dividendo sea menor que el divisor, para colocar entonces el numero 0 y pasar el dividendo a formar el primer dígito del número equivalente en decimal
2. Se toma la parte fraccionaria del número decimal y la multiplicamos por 8 sucesivamente hasta que el producto no tenga números fraccionarios
3. Pasamos la parte entera del producto a formar el dígito correspondiente
4. Al igual que los demás sistemas, el número equivalente en el sistema decimal, está formado por la unión del número entero equivalente y el número fraccionario equivalente.

Conversión de un numero octal a binario
La ventaja principal del sistema de numeración Octal es la facilidad con que pueden realizarse la conversión entre un número binario y octal. A continuación mostraremos un ejercicio que ilustrará la teoría. Por medio de este tipo de conversiones, cualquier número Octal se convierte a binario de manera individual. En este ejemplo, mostramos claramente el equivalente 100 111 010 en binario de cada numero octal de forma individual

Conversión de un numero decimal a un numero hexadecimal
Convertir el número 250.25 a Hexadecimal

1. Se toma la parte entera y se divide sucesivamente por el número decimal 16 (base) hasta que el cociente sea 0
2. Los números enteros resultantes de los cocientes, pasarán a conformar el número hexadecimal correspondiente, teniendo en cuenta que el sistema de numeración hexadecimal posee solo 16 símbolos, donde los números del 10 hasta el 15 tienen símbolos alfabéticos que ya hemos explicado
3. La parte fraccionaria del número a convertir se multiplica por 16 (Base) sucesivamente hasta que el producto resultante no tenga parte fraccionaria
4. Al igual que en los sistemas anteriores, el número equivalente se forma, de la unión de los dos números equivalentes, tanto entero como fraccionario, separados por un punto que establece la diferencia entre ellos.

Conversión de un numero hexadecimal a un numero decimal
Como en los ejemplos anteriores este también nos ayudará a entender mejor este procedimiento: Convertir el numero hexadecimal 2B6 a su equivalente decimal.
1. Multiplicamos el valor de posición de cada columna por el dígito hexadecimal correspondiente.
2. El resultado del número decimal equivalente se obtiene, sumando todos los productos obtenidos en el paso anterior.

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora
- Acceso a internet
- Procesador de palabras

4.- COMPETENCIAS ESPECÍFICAS

Abrir un navegador web, abrir diferentes buscadores y colocar las palabras a buscar, como puede ser: software de conversión numérica, operaciones con diferentes bases numéricas, software para bases numéricas, etc. Entrar a las páginas para ver si el software es libre o necesita licencia, anotar en un procesador de palabras las características de cada software.

Descargar un programa de libre licencia y realizar al menos 5 conversiones y 5 operaciones (suma, resta, multiplicación y/o división), En el procesador de palabras, se deberán anotar los resultados obtenidos por el software.

5. RESULTADOS

Al terminar ésta práctica, el alumno podrá identificar diferente software que hay en el mercado, el cual nos permita realizar diferentes operaciones con diferentes sistemas de numeración.

Cuando termine el alumno de realizar ésta práctica, deberá entregar en formato digital un reporte que contenga los datos generales de cada software, ventajas y desventajas que hay entre cada uno de ellos, además de los resultados de las operaciones realizadas.

6. CONCLUSIONES

Como conclusión, ésta práctica les ayudará al alumno a resolver conversiones y operaciones con diferentes bases numéricas de una forma más rápida y sencilla,

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 3

REPRESENTACIÓN DE CONJUNTOS EN DIAGRAMAS DE VENN

Observaciones: Esta práctica incluye a la Práctica # 3 del temario de Matemáticas Discretas que dice "Utilizando herramientas computacionales disponibles para el alumno, representar el comportamiento de las operaciones con conjuntos mediante diagramas de Venn".

1.- OBJETIVO

Conocer y aplicar los diagramas de Venn en teoría de conjuntos

2.- MARCO TEÓRICO

Un conjunto es un grupo de elementos u objetos especificados en tal forma que se puede afirmar con certeza si cualquier objeto dado pertenece o no a la agrupación. Para denotar a los conjuntos, se usan letras mayúsculas.

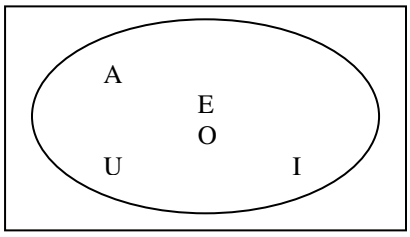
Existen cuatro formas de enunciar a los conjuntos:

a) Por *extensión o enumeración*: los elementos son encerrados entre llaves y separados por comas, es decir, el conjunto se describe listando todos sus elementos entre llaves.
b) Por *comprensión*: los elementos se determinan a través de una condición que se establece entre llaves. En este caso se emplea el símbolo | que significa: "tal que".
c) *Diagramas de Venn*: son regiones encerradas que sirven para visualizar el contenido de un conjunto o las relaciones entre conjuntos.
d) Por *descripción verbal*: es un enunciado que describe la característica que es común para los elementos.

Ejemplo:

Dada la expresión verbal "El conjunto de las vocales", expresarlo por extensión, comprensión y diagramas de Venn:

a) Por extensión: V = {a, e, i, o, u}
b) Por comprensión: V = {x | es un una vocal}
c) Por diagrama de Venn:

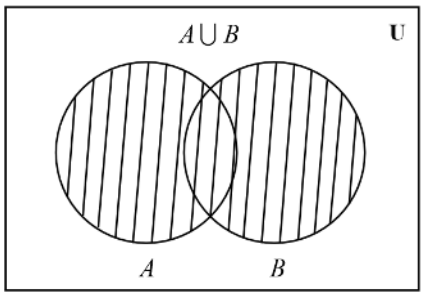


Operaciones con conjuntos.

La unión de los conjuntos A y B es el conjunto de todos los elementos de A con todos los elementos de B sin repetir ninguno y se denota como A ∪ B. Esto es

$$\boldsymbol{A} \cup \boldsymbol{B} = \{\boldsymbol{x} | \boldsymbol{x} \in \boldsymbol{A} \text{ ó } \boldsymbol{x} \in \boldsymbol{B}\}$$

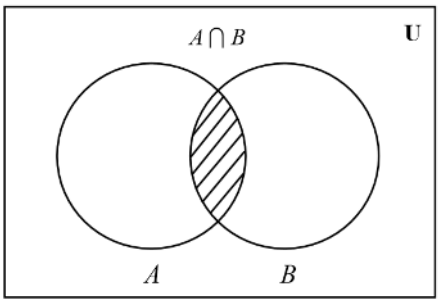
Gráficamente:



La intersección de los conjuntos A y B es el conjunto de los elementos de A que también pertenecen a B y se denota como $A \cap B$. Esto es:

$$\boldsymbol{A} \cap \boldsymbol{B} = \{\boldsymbol{x} | \boldsymbol{x} \in \boldsymbol{A} \, \boldsymbol{y} \, \boldsymbol{x} \in \boldsymbol{B}\}$$

Gráficamente:

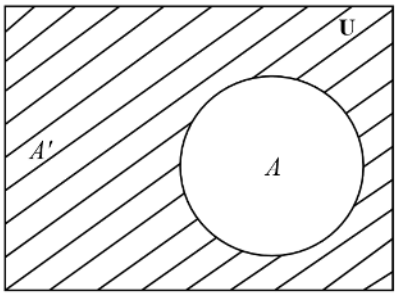


Dos conjuntos son ajenos o disjuntos cuando su intersección es el conjunto vacio, es decir, que no tienen nada en común.

El complemento del conjunto A con respecto al conjunto Universal U es el conjunto de todos los elementos de U que no están en A y se denota como A'. Esto es:

$$A' = \{x \in U | x \, no \, esta \, en \, A\}$$

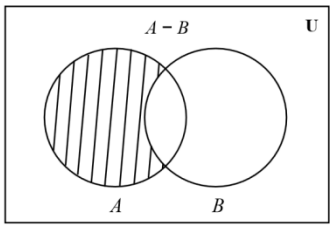
Gráficamente:



La diferencia de los conjuntos A y B (en ese orden) es el conjunto de los elementos que pertenecen a A y no pertenecen a B, y se denota como A – B. Esto es:

$$A - B = \{x | x \in A \; y \; x \; no \; pertenece \; a \; B\}$$

Gráficamente:



Los diagramas de Venn son ilustraciones usadas en la rama de la Matemática y Lógica de clases conocida como teoría de conjuntos. Estos diagramas se usan para mostrar gráficamente la agrupación de cosas elementos en conjuntos, representando cada conjunto mediante un círculo o un óvalo. La posición relativa en el plano de tales círculos muestra la relación entre los conjuntos. Por ejemplo, si los círculos de los conjuntos A y B se solapan, se muestra un área común a ambos conjuntos que contiene todos los elementos contenidos a la vez en A y en B. Si el círculo del conjunto A aparece dentro del círculo de otro B, es que todos los elementos de A también están contenidos en B.

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora
- Proyector digital
- Procesador de Palabras
- Software de presentaciones electrónicas
- Software para dibujo (Paint, Corel Draw o algún otro software)

4.- COMPETENCIAS ESPECÍFICAS

Sean los conjuntos:
U = {a, b, c, d, e, f, g, h, i, j, k, l, m, n}
A = {a, d, e, g, h, k, l, n}
B = {a, c, f, g, k, l, m}

Obtener:

1. $A \cup B$
2. B'
3. $A' \cup B$
4. $A' - B'$
5. $A \cap B$
6. $A - B$
7. $A \cap B'$
8. $(A \cup B)'$
9. A'
10. $B - A$
11. $A' \cap B'$
12. $(A \cap B)'$

Una vez obtenidos los resultados, en un software de dibujo, colocar la representación gráfica de los diagramas de Venn. Pasar los dibujos y resultados obtenidos a una presentación electrónica para que se visualicen los cambios que se obtienen de cada resultado.

5. RESULTADOS

Al terminar ésta práctica, el alumno podrá identificar las diferentes formas de representar los conjuntos y mostrar los resultados de una forma gráfica con los diagramas de Venn.

Cuando termine el alumno de realizar ésta práctica, deberá entregar en formato digital un reporte que contenga las diferentes operaciones que se pueden realizar entre conjuntos y la forma de representarlos. Podrá pasar a mostrar su presentación electrónica de los resultados de ejercicios ante los compañeros de clases.

6. CONCLUSIONES

Como conclusión, ésta práctica les ayudará al alumno a identificar de forma visual los resultados que se obtuvieron al aplicar diferentes operaciones entre conjuntos.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 4

EVALUACIÓN DE PROPOSICIONES MEDIANTE TABLAS DE VERDAD

Observaciones: Esta práctica incluye a la Práctica # 4 del temario de Matemáticas Discretas que dice "Elaborar, con ayuda de una hoja electrónica de cálculo, un proceso para llevar a cabo la evaluación de una proposición compuesta mediante tablas de verdad".

1.- OBJETIVO

Aprender a identificar todos los conectores lógicos, la diferencia que hay entre cada uno de ellos y poder aplicarlos para evaluar si una proposición lógica es una Veracidad, Falacia o Contradicción.

2.- MARCO TEÓRICO

La lógica proposicional o lógica de orden cero es un sistema formal cuyos elementos más simples representan proposiciones, y cuyas constantes lógicas, llamadas conectivas, representan operaciones sobre proposiciones, capaces de formar otras proposiciones de mayor complejidad.

La lógica proposicional trata con sistemas lógicos que carecen de cuantificadores, o variables interpretables como entidades. En lógica proposicional si bien no hay signos para variables de tipo entidad, sí existen signos para variables proposicionales (es decir, que pueden ser interpretadas como proposiciones con un valor de verdad de definido), de ahí el nombre proposicional.

La lógica proposicional incluye además de variables interpretables como proposiciones simples signos para conectivas lógicas, por lo que dentro de este tipo de lógica puede analizarse la inferencia lógica de proposiciones a partir de proposiciones, pero sin tener en cuenta la estructura interna de las proposiciones más simples.

Conectores lógicos.
A continuación hay una tabla que despliega todas las conectivas lógicas que ocupan a la lógica proposicional, incluyendo ejemplos de su uso en el lenguaje natural y los símbolos que se utilizan para representarlas en lenguaje formal.

Conectiva	Expresión en el lenguaje natural	Ejemplo	Símbolo en este artículo	Símbolos alternativos
Negación	no	**No** está lloviendo.	$\neg$	$\sim$
Conjunción	y	Está lloviendo **y** está nublado.	$\wedge$	$\&$,
Disyunción	o	Está lloviendo **o** está soleado.	$\vee$	
Condicional material	si... entonces	**Si** está soleado, **entonces** es de día.	$\rightarrow$	$\supset$
Bicondicional	si y sólo si	Está nublado **si y sólo si** hay nubes visibles.	$\leftrightarrow$	$\equiv$
Negación conjunta	ni... ni	**Ni** está soleado **ni** está nublado.	$\downarrow$	
Disyunción excluyente	o bien... o bien	**O bien** está soleado, **o bien** está nublado.	$\nleftrightarrow$	$\oplus, \not\equiv, W$

Tablas de verdad.
Una tabla de verdad, o tabla de valores de verdad, es una tabla que muestra el valor de verdad de una proposición compuesta, para cada combinación de valores de verdad que se pueda asignar a sus componentes

Conjunción:
La conjunción es un operador que opera sobre dos valores de verdad, típicamente los valores de verdad de dos proposiciones, devolviendo el valor de verdad verdadero cuando ambas proposiciones son verdaderas, y falso en cualquier otro caso. Es decir es verdadera cuando ambas son verdaderas

La tabla de verdad de la conjunción es la siguiente:

A	B	$A \wedge B$
V	V	V
V	F	F
F	V	F
F	F	F

Disyunción:
La disyunción es un operador que opera sobre dos valores de verdad, típicamente los valores de verdad de dos proposiciones, devolviendo el valor de verdad verdadero cuando una de las proposiciones es verdadera, o cuando ambas lo son, y falso cuando ambas son falsas.

La tabla de verdad de la disyunción es la siguiente:

A	B	$A \vee B$
V	V	V
V	F	V
F	V	V
F	F	F

Implicación o Condicional:

El condicional material es un operador que opera sobre dos valores de verdad, típicamente los valores de verdad de dos proposiciones, devolviendo el valor de falso sólo cuando la primera proposición es verdadera y la segunda falsa, y verdadero en cualquier otro caso.

La tabla de verdad del condicional material es la siguiente:

A	B	$A \rightarrow B$
V	V	V
V	F	F
F	V	V
F	F	V

Bicondicional:

El bicondicional o doble implicación es un operador que funciona sobre dos valores de verdad, típicamente los valores de verdad de dos proposiciones, devolviendo el valor de verdad verdadero cuando ambas proposiciones tienen el mismo valor de verdad, y falso cuando sus valores de verdad diferente.

La tabla de verdad del bicondicional es la siguiente:

A	B	$A \leftrightarrow B$
V	V	V
V	F	F
F	V	F
F	F	V

La evaluación de proposiciones compuestas consiste en hallar los valores del operador principal a partir de cada una de las componentes. Se pueden tener tres casos distintos:

- **Tautología:** una proposición es una tautología cuando es verdadera cualquiera sea el valor de verdad de las proposiciones componentes.
- **Contradicción:** una proposición es una contradicción cuando se da el caso opuesto a la tautología, es decir, cuando es falsa cualquiera sea el valor de verdad de las proposiciones componentes.
- **Contingencia:** una proposición es una contingencia si el valor de verdad de su operador contiene por lo menos un verdadero y un falso.

De aquí nace la definición de proposiciones lógicamente equivalentes, es decir, proposiciones cualesquiera (p y q) donde p↔q es una tautología, es decir, que tienen el mismo valor de verdad. La equivalencia se denota p≡q.

Funciones Lógicas en Excel:

Y → devuelve VERDADERO si todos los argumentos son VERDADERO; devuelve FALSO si uno o más argumentos son FALSO.

Sintaxis
Y(valor_lógico1;valor_lógico2; ...)

FALSO → Devuelve el valor lógico FALSO.
Sintaxis
FALSO( )

VERDADERO → Devuelve el valor lógico VERDADERO.
Sintaxis
VERDADERO ( )

SI → Devuelve un valor si la condición especificada es VERDADERO y otro valor si dicho argumento es FALSO. Utilice SI para realizar pruebas condicionales en valores y fórmulas.

Sintaxis
SI(prueba_lógica;valor_si_verdadero;valor_si_falso)

Prueba_lógica es cualquier valor o expresión que pueda evaluarse como VERDADERO o FALSO. Por ejemplo, A10=100 es una expresión lógica; si el valor de la celda A10 es igual a 100, la expresión se evalúa como VERDADERO. De lo contrario, la expresión se evalúa como FALSO. Este argumento puede utilizar cualquier operador de comparación.

Valor_si_verdadero es el valor que se devuelve si el argumento prueba_lógica es VERDADERO. Por ejemplo, si este argumento es la cadena de texto "Dentro de presupuesto" y el argumento prueba_lógica se evalúa como VERDADERO, la función SI muestra el texto "Dentro de presupuesto". Si el argumento prueba_lógica es VERDADERO y el argumento valor_si_verdadero está en blanco, este argumento devuelve 0 (cero). Para mostrar la palabra VERDADERO, utilice el valor lógico VERDADERO para este argumento. Valor_si_verdadero puede ser otra fórmula.

Valor_si_falso es el valor que se devuelve si el argumento prueba_lógica es FALSO. Por ejemplo, si este argumento es la cadena de texto "Presupuesto excedido" y el argumento prueba_lógica se evalúa como FALSO, la función SI muestra el texto "Presupuesto excedido". Si el argumento prueba_lógica es FALSO y se omite valor_si_falso, (es decir, después de valor_si_verdadero no hay ninguna coma), se devuelve el valor lógico FALSO. Si prueba_lógica es FALSO y valor_si_falso está en blanco (es decir, después de valor_si_verdadero hay una coma seguida por el paréntesis de cierre), se devuelve el valor 0 (cero). Valor_si_falso puede ser otra fórmula.

NO → invierte el valor lógico del argumento. Use NO cuando desee asegurarse de que un valor no sea igual a otro valor específico.

Sintaxis
NO(valor_lógico)

Valor_lógico es un valor o expresión que puede evaluarse como VERDADERO o FALSO.

O → Devolverá VERDADERO si alguno de los argumentos es VERDADERO; devolverá FALSO si todos los argumentos son FALSO.

Sintaxis
O(valor_lógico1;valor_lógico2; ...)

Valor_lógico1;valor_lógico2,... son entre 1 y 30 condiciones que desea comprobar y que pueden ser VERDADERO o FALSO.

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.
- Hoja de cálculo.

4.- COMPETENCIAS ESPECÍFICAS

En un archivo de hoja electrónica de cálculo (puede ser Excel), formaliza siguiente argumento, una vez formalizado, realiza su tabla de verdad e indica si es válidos (tautologías) o no. Una vez que termines de analizar la tabla, escribe en un archivo de texto las herramientas que utilizaste y cómo desarrollaste la práctica, también una conclusión de para qué nos sirven las tablas de verdad y los razonamientos lógicos.

Jaime se come el helado o se le derretirá, no se derrite el helado, por lo tanto, Jaime se come el helado.
P: Jaime se come el helado
Q: El helado se derrite

((p V q) ^ ¬q) → p

P	Q	¬Q	P OR Q	(p or q) and not q	((p or q) and not q)-->p
V	V				
V	F				
F	V				
F	F				

Las funciones que necesitas para desarrolla esta práctica se encuentran en el marco teórico.

5. RESULTADOS

Al terminar ésta práctica, el alumno podrá aplicar las tablas de verdad para analizar si una proposición lógica es válida o no lo es. Cuando termine el alumno de realizar ésta práctica, deberá entregar en formato digital un reporte que contenga los resultados obtenidos en la práctica (puede obtener una tautología, contradicción o contingencia.

6. CONCLUSIONES

Como conclusión, ésta práctica les ayudará al los alumnos a comprender la importancia de analizar proposiciones con tablas de verdad.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 5

COMPORTAMIENTO DE UAN EXPRESIÓN PROPOSICIONAL

Observaciones: Esta práctica incluye a la Práctica # 5 del temario de Matemáticas Discretas que dice "Utilizando un simulador, verificar el comportamiento de una expresión proposicional".

1.- OBJETIVO

Observar el comportamiento de las expresiones proposicionales mediante el uso de software de simulación especializado en el área de lógica matemática.

2.- MARCO TEÓRICO

Dentro de la *lógica simbólica*, que se vale de símbolos para analizar razonamientos y sus partes, se encuentra la **lógica proposicional**. La lógica proposicional simboliza, generalmente con letras minúsculas del alfabeto (p, q, r, s, por ejemplo) las denominadas proposiciones simples o atómicas que constituyen las partes de ciertas oraciones más complejas.

Por ejemplo, en la oración **"llueve y hace frío"** nos encontramos con dos proposiciones simples ("llueve" es una y la otra "hace frío") que se encuentran unidas por una *conectiva lógica llamada* **conjunción**.En este caso, al simbolizar la oración a partir de sus proposiciones componentes y el modo cómo se unen obtenemos una expresión que se denomina **forma proposicional**, y a veces algo equívocamente *forma de enunciado*.

En la nomenclatura habitual de la lógica proposicional se simbolizaría

p . q (que se lee " p y q"), siendo el puntito el símbolo de la mencionada **conjunción**.

Naturalmente, una forma proposicional puede ser más compleja, como la que resultaría de simbolizar las proposiciones atómicas de "*si no llueve ni hace frío voy a tu casa o al cine*", por ejemplo. Pero dejaremos para otra ocasión la simbolización para intentar clarificar el **significado y alcance de una forma proposicional.**

Se dice que éstas son expresiones "**veritativo funcionales**", o que una forma proposicional es *funcion de verdad de las proposiciones simples o atómicas que contiene y del significado de las conectivas lógicas que las vinculan*.

Esto puede parecer complejo, pero no lo es realmente. Que una cierta forma proposicional es tal mencionada cosa significa que **su valor de verdad** (*que "llueve y hace frío" sea una expresión verdadera o*

falsa **dependerá de dos cosas**: *si sus elementos componentes son verdaderos o falsos* (p y q respectivamente) y del *significado de las conectivas que las vinculan*.

El **significado de la conjunción** es que es **verdadera únicamente cuando las dos partes que une son verdaderas.**

Por ejemplo, la oración "La Luna es un satélite y gira alrededor de marte" es una oración falsa, porque "la Luna es un satélite" es verdadera (sería p) pero "la Luna gira alrededor de Marte" es una proposición simple falsa, por lo que la conjunción es falsa. "una silla es un mueble y tiene patas" es una oración verdadera.

Entonces dada cualquier forma proposicional de la forma "p . q", ella sólo será verdadera cuando p, sea lo que fuere que simbolice, es **verdadera y q también**. En cualquier otro caso será falsa (Verdadero/Falso; Falso/Verdadero; Falso/Falso).

Diferentes formas proposicionales, con *diferente estructura interna y diferentes conectivas* tendrán distintos modos de poder ser verdaderas o falsas, dependiendo de los dos aspectos mencionados, pero una vez determinado si p, q, r, etc. son verdaderas o falsas y una vez advertidas las conectivas que las unen (cuyo significados los lógicos conocen y los estudiantes deben asimilar), **queda determinado por completo el valor de verdad de la expresión total.**

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.
- Hoja de cálculo.
- Acceso a Internet

4.- COMPETENCIAS ESPECÍFICAS

Mediante el uso de un software para verificar el comportamiento de una expresión proposicional, evaluar las siguientes afirmaciones:

Si se tiene: "Gana o Pierde o Empata" y "Si Gana entonces da una Fiesta o Va de Viaje". Se puede deducir que: "O Pierde o Empata o da una Fiesta o va de Viaje".

Sea C el siguiente conjunto de cláusulas $\{p,\ \neg p \vee q,\ \neg r,\ \neg p \vee \neg q \vee r\}$, demostrar que C es insatisfacible por resolución.

Puedes analizar las proposiciones en hojas de papel y posteriormente verificar si tus resultados coinciden con los que arroje el simulador de expresiones proposicionales.

5. RESULTADOS

El alumno ´deberá entregar un reporte digital con los resultados obtenidos al evaluar cada una de las expresiones proposicionales y determinar su comportamiento.

6. CONCLUSIONES

Al finalizar ésta práctica, el alumno podrá analizar dada una expresión proposicional, el grado de veracidad o falsedad de dicha expresión. Y tendrá un mejor razonamiento lógico.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 6

DETERMINACIÓN DE RAZONAMINETO VÁLIDO

Observaciones: Esta práctica incluye a la Práctica # 6 del temario de Matemáticas Discretas que dice "Utilización de diagramas de Venn para la determinación de razonamiento".

1.- OBJETIVO

Dado un razonamiento lógico, analizar su grado de veracidad o falsedad mediante el uso de diagramas de Venn

2.- MARCO TEÓRICO

Algoritmo de resolución proposicional.

El algoritmo se basa en una regla de inferencia sencilla y, a la vez de gran potencia: la regla de resolución.
Puesto que se utiliza una sola regla, el algoritmo es fácil de analizar e implementar.

La idea del principio de resolución es simple: Si se sabe que se cumple: "P ó Q" y también se sabe que se cumple "no P ó R" entonces se puede deducir que se cumplirá "Q ó R".

Definición: Dadas dos cláusulas C_1 y C_2 tales que exista un literal l de forma que $l \in C_1$ y $\neg l \in C_2$, se denomina resolvente de C_1 y C_2 respecto a l a la cláusula:

$$R_i(C_1 C_2) = (C_1 - \{l\}) \cup (C_2 - \{\sim l\})$$

Se dice que *C*1 y *C*2 son cláusulas resolubles.

Teorema (Consistencia de la regla de resolución): El resolvente de dos cláusulas es consecuencia lógica de ellas. Es decir $\{C_1, C_2\} \Rightarrow R(C_1, C_2)$

Teorema: Dadas dos cláusulas C_1 y C_2 pertenecientes a un conjunto C y resolubles respecto un literal l, entonces: $C \equiv C \cup R_l(C_1, C_2)$.

Teorema: Si el resolvente de dos cláusulas C_1 y C_2 pertenecientes a un conjunto C es la cláusula vacía, entonces C es insatisfacible.

A partir de los teoremas anteriores, se define el algoritmo de resolución que chequeará si un conjunto de cláusulas es insatisfacible.

Algoritmo de resolución insatisfacible.

Entrada: un conjunto cláusulas C
Salida: detecta si C es insatisfacible

1. Buscar dos cláusulas $C_1, C_2 \in C$ tales que exista una literal l que cumple que $l \in C_1$ y $\sim l \in C_2$
2. Si se encuentran:
   a. Calcular $Rl\ (C1\ ,C2\ )$ y añadirlo al conjunto C
   b. Si $Rl\ (C1\ ,C2\ ) = \square$ entonces **SALIR** indicando que C es **insatisfacible.**
   c. **Si no, volver a 1**
3. Si no se encuentran: **SALIR** indicando que C **no es insatisfacible**

ESTRATEGIAS DE RESOLUCIÓN:

El método de resolución es un algoritmo no determinista ya que pueden encontrarse múltiples formas de alcanzar la cláusula vacía en un conjunto insatisfacible. Muchas veces, siguiendo un determinado camino se alcanzará la cláusula vacía con muchos menos pasos de resolución que por otro camino.

Durante el desarrollo del algoritmo es necesario responder las siguientes preguntas: ¿Qué dos cláusulas se seleccionan? y ¿sobre qué literales se realiza la resolución?.

Las distintas estrategias de resolución tratan de responder a ambas preguntas de forma que se mantenga la completud (si el conjunto es insatisfacible, alcanzar la cláusula vacía) y que se obtenga un comportamiento eficiente.

Una de las desventajas de la utilización de la reglas de resolución sin ninguna restricción consiste en que se pueden seleccionar cláusulas cuyo resolvente no sea útil en el camino de búsqueda de la cláusula vacía. Se observa que muchas veces los resolventes son redundantes o no aportan ninguna utilidad para la búsqueda. A continuación se mencionan una serie de estrategias que servirán para eliminar el trabajo inútil.

Eliminación de cláusulas con literales puros

Definición: Un literal es puro si y sólo si no existe un literal complementario a él en el conjunto de cláusulas. Una cláusula que contenga un literal puro es inútil en la búsqueda de la cláusula vacía, puesto que el literal puro no podrá ser eliminado nunca mediante resolución. Por tanto, una estrategia de borrado consiste en la eliminación de cláusulas con literales puros.

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.
- Hoja de cálculo.
- Software para dibujo (Paint, Corel Draw o algún otro software)

4.- COMPETENCIAS ESPECÍFICAS

Mediante el uso de diagramas de Venn, demuestra los siguientes razonamientos lógicos.

Determinar si el razonamiento $\{p \wedge q \rightarrow r \wedge s, p \rightarrow \neg s\} \Rightarrow \neg p \vee \neg q$ es correcto por resolución y por medio de un diagrama de Venn.

Sea el conjunto de cláusulas C = $\{p, \neg p \vee q, \neg r, \neg p \vee \neg q \vee r\}$, construir el árbol semántico para C y dibujar el diagrama de Venn para analizar si el razonamiento es válido o no

5. RESULTADOS

El alumno ´deberá entregar un reporte digital con los resultados obtenidos al evaluar cada una de las expresiones proposicionales y determinar su grado de veracidad.

6. CONCLUSIONES

Al finalizar ésta práctica, el alumno podrá determinar dada una expresión proposicional, la veracidad o falsedad de dicha expresión. Consiguiendo un mejor razonamiento lógico.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 7

COMPUERTAS LÓGICAS

Observaciones: Esta práctica incluye a la Práctica # 7 del temario de Matemáticas Discretas que dice "Construir un circuito usando compuertas lógicas, implementarlas utilizando software para la construcción de circuitos electrónicos".

1.- OBJETIVO

Utilizar las diferentes compuertas lógicas aplicando diferentes operaciones entre ellas para diseñar un diagrama que nos permita construir un circuito electrónico.

2.- MARCO TEÓRICO

George Boole (1815-1864) presentó el primer tratamiento sistemático de la lógica y para ello, desarrolló un sistema algebraico, conocido ahora como **Álgebra de Boole**. Además de sus aplicaciones al campo de la lógica, el álgebra de Boole ha tenido dos aplicaciones importantes: el tratamiento de conjuntos mediante las operaciones de unión e intersección que ha servido de base a la teoría de la probabilidad y el diseño de **circuitos digitales combinacionales**.

Un álgebra de Boole es una estructura de la forma **{A, +, ´ , -, 0,1}** siendo A un conjunto en el que se definen las siguientes operaciones:

+ y × son leyes de composición binaria sobre A: $a + b \in A$ y $a \times b \in A$ $\forall a,b \in$ A
- es una ley de composición unaria sobre A: $a \in A$ $\forall$ a $\in$ A

Verificándose los postulados:

1. Conmutativa:	a + b = b + a	∀a,b ∈ A	{conmutativa +}
	a × b = b × a	∀a,b ∈ A	{conmutativa ×}
2. Distributiva:	a + (b × c) = (a + b) × (a + c)	∀a,b,c ∈ A	{distributiva +}
	a × (b + c) = (a × b) + (a × c)	∀a,b,c ∈ A	{distributiva ×}
3. Elemento neutro:	a + 0=a	∀ a ∈ A	{neutro +}
	a × 1=a	∀ a ∈ A	{neutro ×}
4. Elemento inverso:	$a + a = 1$	∀ a ∈ A	{inverso +}
	$a * a = 0$	∀ a ∈ A	{inverso ×}

En el caso más sencillo, el conjunto A tiene como únicos elementos a los neutros de las operaciones, A={ 0 , 1 }. Esto quiere decir que las variables sólo pueden tomar los valores 0 o 1. En este caso, el álgebra de Boole se dice que es bivaluada.

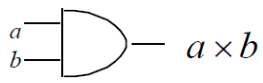
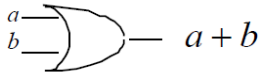
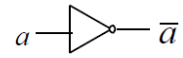
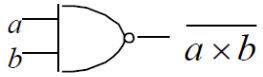
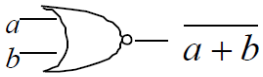
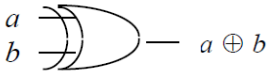
Teorema (Principio de dualidad): Cada identidad deducida de los postulados del álgebra de Boole permanece válida si se intercambian las operaciones + y ×, y los valores 0 y 1.

De manera informal, este teorema puede demostrarse indicando que, puesto que los postulados son todos simétricos y cumplen la propiedad de dualidad, todo lo que se deduzca de ellos, cumplirá también dicha propiedad.

Compuertas lógicas

Un *circuito digital* es un circuito electrónico cuyas entradas y salidas sólo pueden tomar dos niveles distintos de tensión. Desde el punto de vista del diseño, estos niveles de tensión se representan como 1 (verdadero) ó 0 (falso). Un *circuito combinacional* se caracteriza por ser un sistema sin *memoria:* el valor de las salidas en cada instante depende sólo del valor de las entradas en ese momento. Un circuito de estas características puede representarse analíticamente, mediante una *función booleana*, o gráficamente, mediante un *diagrama de puertas lógicas*. En estos diagramas se representan las entradas, las salidas, las operaciones o puertas lógicas y sus conexiones.

Las diferentes conectivas pueden representarse mediante las siguientes puertas lógicas.

Puerta AND 	Puerta OR 	Puerta NO (Inversor) 
Puerta NAND 	Puerta NOR 	Puerta XOR (O-Exclusiva) 

Funciones Booleanas

Definición. Una **variable booleana** es una variable que toma únicamente dos valores 0 ó 1.
Definición. Una **función Booleana** es una expresión algebraica que relaciona **variables Booleanas** por medio de las operaciones +, ×, y –.
Ejemplo: $f(a,b,c) = (a+b)\times c + a \times (b+c)$

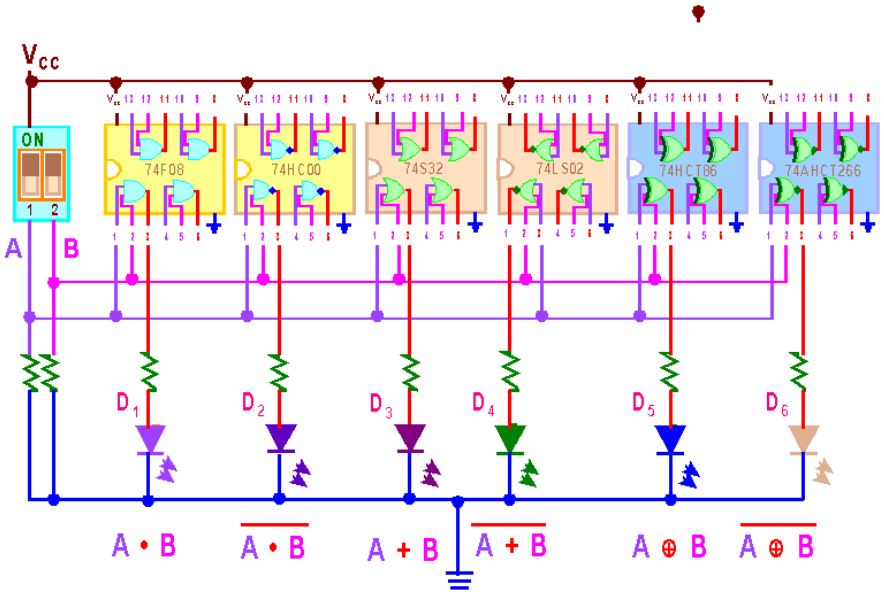
3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora
- Simulador de compuerta lógicas
- Procesador de Palabras

4.- COMPETENCIAS ESPECÍFICAS

Mediante el uso de un simulador de compuertas lógicas, **Comprobar** las **tablas funcionales** o de **verdad** de los componentes básicos **Y** (*AND*), **O** (*OR*), **NO** (*NOT*), **NO-Y** (*NAND*)**, NO-O** (*NOR*), **O-EXCLUSIVA** (*OREX*) y **NO-O-EXCLUSIVA** (*NOREX*), utilizando circuitos integrados.

Armar el siguiente **circuito topológico** para comprobar las **tablas de verdad**.



5. RESULTADOS

Al término de ésta práctica, el alumno deberá entregar un reporte electrónico de cómo se realizó la construcción del circuito integrado. El reporte lo deberá entregar al profesor de forma elecrónica.

6. CONCLUSIONES

Se espera que el alumno interactúe por primera vez con las compuertas lógicas utilizando circuitos integrados para que vea la aplicación de la teoría de álgebra booleana.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 8

RELACIONES Y EL MODELO RELACIONAL

Observaciones: Esta práctica incluye a la Práctica # 8 del temario de Matemáticas Discretas que dice "Ejemplificar el modelo relacional utilizado en las bases de datos".

1.- OBJETIVO

Comprender la importancia de utilizar las relaciones matemáticas para modelar relaciones en la creación de bases de datos.

2.- MARCO TEÓRICO

La clase más importante de relaciones es la de las relaciones binarias. Debido a que este tipo de relaciones son las más frecuentes, el término "relación" denota generalmente una relación binaria; adoptaremos este criterio cuando no haya confusión y especificaremos las que no sean binarias con términos tales como "ternaria" o "n-aria".

Si (a, b) $\in R$ diremos que a está relacionado con b y lo notaremos por aRb.
Si (a, b) $\notin R$, escribiremos a$\not R$b y diremos que a no está relacionado con b.

Dominio o imagen

Llamaremos dominio de una relación R al conjunto formado por todos los primeros elementos de los pares ordenados que pertenecen a R, e imagen o rango al conjunto formado por los segundos elementos.

Es decir, si R es una relación de A a B, entonces

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

Matriz de una relación.

Dados los conjuntos finitos, no vacios,

A = {a_1, a_2, …, a_m } 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:

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

Donde i = 1, 2,….m; j = 1,2,…n

Directamente de la definición dada se deduce que la matriz de una relación binaria es cuadrada.

El modelo relacional

El modelo relacional constituye una alternativa para la organización y representación de la información que se pretende almacenar en una base de datos. Se trata de un modelo teórico matemático que, además de proporcionarnos los elementos básicos de modelado (las relaciones), incluye un conjunto de operadores (definidos en forma de un álgebra relacional) para su manipulación, sin ambigüedad posible.

El carácter formal del modelo relacional hace relativamente sencilla su representación y gestión por medio de herramientas informáticas. No es casual, pues, que haya sido elegido como referencia para la construcción de la gran mayoría de los Sistemas de Gestión de Bases de Datos comerciales disponibles en el mercado; ni tampoco que sea también habitualmente seleccionado como modelo de referencia para la elaboración del esquema lógico de una base de datos, como tercer paso de la habitual metodología de diseño de BDs (después del análisis de requerimientos y la elaboración del esquema conceptual).

En el modelo relacional se basa en el concepto matemático de relación. En este modelo, la información se representa en forma de “tablas” o relaciones, donde cada fila de la tabla se interpreta como una relación ordenada de valores (un conjunto de valores relacionados entre sí). El siguiente ejemplo presenta una relación que representa al conjunto de los departamentos de una determinada empresa, y que recoge información sobre los mismos.

Número	Nombre	Localidad
D-01	Ventas	A. Coruña
D-02	I+D	Ferrol

Tabla: Relación “Departamentos”

Formalmente, una relación se define como un conjunto de n–tuplas; donde una n–tupla se define a su vez como un conjunto ordenado de valores atómicos (esto es, no divisibles ni descomponibles en valores más “pequeños”.

En el ejemplo 1, la relación mostrada incluye dos 3–tuplas: (‘D01’, ‘Ventas’, ‘A Coruña’) y (‘D-02’, ‘I+D’, ‘Ferrol’). Cada tupla incluye información sobre los departamentos de una determinada empresa con sede en Galicia: el identificador del departamento dentro de la empresa, su nombre, y la localidad donde tiene su sede. En cada tupla, los tres valores están relacionados por el hecho de describir todos ellos al mismo departamento.

Cada relación, vista como una tabla, consta de un conjunto de columnas; cada una de esas columnas recibe el nombre de atributo. A cada atributo de una relación le corresponde un nombre, que debe ser

único dentro de la relación, y un dominio: el conjunto de valores válidos para un atributo; o, dicho de otra manera, el conjunto de valores que cada tupla de la relación puede tomar para ese atributo.

En el caso de la relación de nuestro ejemplo, los atributos de la misma serían Num, Nombre y Localidad. Cada uno de ellos tendrá un dominio asociado: el conjunto de los identificadores válidos de departamento (una cadena alfanumérica con formato „D-xx"), el conjunto de todos los nombres de departamento válidos (cadenas de texto de cualquier longitud), y el conjunto de todas los nombres de localidades gallegas (ídem), respectivamente

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.

4.- COMPETENCIAS ESPECÍFICAS

Dado el esquema relacional siguiente:

Cliente(dni, nombre-cli, direccion-cli, tarifa)
Automóvil(matricula, marca, modelo, pagado, dni, #clase)
Categoría(#clase, tasa)
Taller(cif, nombre-tal, direccion-tal)
Accidente(#accidente, dni, matricula, cif, fecha, coste)

Expresar de forma matemática las relaciones que existen en dicho esquema, también se deberá indicar el tipo de relación que hay.

5. RESULTADOS

El resultado que deberá entregar el alumno es un reporte en formato electrónico de cómo identificó los tipos de relaciones que hay y la expresión en forma matemática de las relaciones que encontró.

6. CONCLUSIONES

Ésta práctica ayudará al alumno a entender los fundamentos matemáticos de la creación de relaciones en una base de datos.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. "Matemáticas discreta y combinatoria" 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. "Matemáticas para la computación". Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. "Matemáticas para la Computación". Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 9

RELACIONES DE ORDEN PARCIAL

Observaciones: Esta práctica incluye a la Práctica # 9 del temario de Matemáticas Discretas que dice "Utilizando software disponible para el alumno, determinar las propiedades de una relación, aplicar cerraduras para lograr que una relación sea de equivalencia y determinar el diagrama de Hasse de relaciones de orden parcial".

1.- OBJETIVO

Identificar las propiedades de las relaciones de orden parcial y generar su diagrama Hasse

2.- MARCO TEÓRICO

Relación 'R': Es cualquier subconjunto de A×B que cumpla la propiedad en concreto.

Es decir, siendo $x \in A, y \in A$ y grafo(gráfica) $R / R \subseteq A \times A$, decimos que xRy si $(x,y) \in R$

El nº de relaciones ó subconjuntos de A×B será $2^{|A| \cdot |B|}$

Relación n-aria: Cualquier subconjunto del producto cartesiano de $A_1 \times A_2 \times ... \times A_n$ (Una relación binaria sería una relación de $A_1 \times A_2$)

Propiedades que puede cumplir una relación:

1) Reflexiva, si $\forall\ a \in A \Rightarrow aRa$
2) Simétrica si $\forall\ a,b\ /\ aRb \Rightarrow bRa$
3) Transitiva, si $\forall\ a,b,c\ /\ (aRb \text{ y } bRc) \Rightarrow aRc$
4) Antisimetrica, si $\forall\ a,b\ /\ (aRb \text{ y } bRa) \Rightarrow a=b$

Todo elemento cumple las tres primeras consigo mismo. Cuidado con la 4º: no simetrica≠antisimetrica

Matriz de una relación A×B: - filas = elementos de A, columnas = elementos de B

1 si $(a_i,b_j) \in R$, 0 si $(a_i,b_j) \notin R$

Relación equivalente '~': Es la relación binaria que verifica las propiedades reflexiva, simétrica y transitiva.

Clase de equivalencia '[x]': Dada una relación de equivalencia en un conjunto A, se define clase de equivalencia de un $a \in A$, como el conjunto de elementos de A equivalentes al elemento dado. Se denota como $[a]=\{a' \in A\ /\ a' \sim a\}$

El representante de la clase de equivalencia puede ser cualquier elemento del conjunto. Asi, se cumple:

- a'~a $\Leftrightarrow$ [a']=[a] (prop. transitiva)
- y por tanto, x no~a $\Leftrightarrow$ $[x]\cap[a] = \varnothing$

Por transitividad de ~ es imposible que $[x]\cup[a] \neq [x]\cap[a]$ porque las clases de equivalencia son idénticas o disjuntas La clase de equivalencia de cualquier elemento x cumple $[x]\neq\varnothing$ porque $x\in[x]$

Las clases de equivalencia forman una familia de subconjuntos$\neq\varnothing$ y disjuntos entre si, (porque por transitiva si tuvieran un elemento en común serían iguales), cuya unión es A.

Relación de orden '≤' en un conjunto dado:
Es una relación binaria que cumple propiedades reflexiva, antisimetrica y transitiva.
Se dice que (A,≤) es un conjunto parcialmente ordenado '**poset'** si verifica una relación de orden.
Un poset es además un **orden total** si $\forall$ x, y $\in$ A se cumple x*R*y ó y*R*x.
En caso contrario sera un **orden parcial**.

Diagramas de Hasse
Es la representación de una relación de orden, mediante aristas no dirigidas entre 2 elementos x, y si y solo si y cubre a x. Se dice que **y cubre a x** cuando se cumplen los dos siguientes enunciados:

- $x \leq y$
- $x\leq z\leq y \Rightarrow y=z$ o x=z (no hay ningún elemento entre los dos)

Las aristas se leen de abajo arriba por convención (al $\exists$ una dirección de lectura no hacen falta aristas dirigidas).

Si R es una relación de orden en A, se elabora un diagrama de Hasse para R en A trazando segmentos de recta no dirigida de *x* a *y*, si x,y∈A, con x*R*y, pero solo si no hay otro elemento z∈A tal que x*R*z, z*R*y. En el grafo de una relación de orden son superfluos los lazos y aristas multiples (se sobreentiende su existencia por las propiedades reflexiva y transitiva)

Isomorfos
Sean (P,≤) y (Q,≤) (Q c. imagen de P) conjuntos parcialmente ordenados. Se dice que son 'Isómorfos' si
$\exists$ f:P$\longrightarrow$Q biyectiva que mantiene el orden para a,b∈P: $a\leq b \Leftrightarrow f(a) \leq f(b)$

Sea (A,≤) un conjunto ordenado y $C\subseteq A / C\neq\varnothing$:

- k∈A es "Cota superior" de C si $x\leq k$, $\forall x\in C$,"Supremo" será la menor de las cotas superiores.
- k∈A es "Cota inferior" de C si $k\leq x$ $\forall x\in C$, "Ínfimo" será la mayor de las cotas inferiores.
- Un elemento k de A
  $\forall$ x ∈ C, $x\leq k \Rightarrow$ k es máximo
  $\forall$ x ∈ C, $k\leq x \Rightarrow$ k es mínimo
- x∈C es *maximal/minimal* de C si ningún elemento de C es >/< que x.

Todo conjunto poset finito tiene al menos 1 maximal y 1 minimal.

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.

4.- COMPETENCIAS ESPECÍFICAS

En una hoja de papel, dado el conjunto A = {a, b, c} dibujar el diagrama de Hasse del conjunto parcialmente ordenado $\{\mathcal{P}(A), \subseteq\}$

Determinar el diagrama Hasse de las relaciones siguientes:

a) A = {1, 2, 3, 4} y

R = {(1, 1); (1, 2); (2, 2); (2, 4); (1, 3); (3, 3); (3, 4); (1, 4); (4, 4)}

b) A = {a, b, c, d, e} y

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

Al terminar, busca software que permita determinar las propiedades de las relaciónes, aplicar cerraduras para lograr que una relación sea de equivalencia y determinar el diagrama de Hasse de relaciones de orden parcial.

5. RESULTADOS

Al finalizar la práctica, el alumno debe entregar como resultado un reporte en el que se muestre los diagramas de Hasse resultantes de las dos asignaciones previas.

6. CONCLUSIONES

Los diagramas hasse ayudarán al alumno a eliminar la necesidad de representar ciclos en los grafos puesto que se entiende que una relación parcialmente ordenada es reflexiva. Puesto que la transitividad también está implicada, se puede prescindir de mostrar las líneas entre cada elemento.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 10

CÁLCULO DE CAMINOS DE UN GRAFO

Observaciones: Esta práctica incluye a la Práctica # 10 del temario de Matemáticas Discretas que dice "Representar un grafo utilizando una hoja electrónica de cálculo, y obtener el número de caminos de longitud n mediante el cálculo correspondiente".

1.- OBJETIVO

A través de una representación de grafos, obtener el número de caminos de longitud n entre los nodos de los grafos.

2.- MARCO TEÓRICO

Un grafo G es el par (V,A) que representa una relación entre un conjunto de Vertices y otro de Aristas.
Representaremos cada elemento arista como un par de elementos de V.
Gráficamente representaremos los vértices por puntos y las aristas por líneas que los unen.
Un vértice puede tener 0 o más aristas, pero toda arista debe unir exactamente 2 vértices.
Orden de un grafo: es su nº de vértices = |V|. Si |V| es finito se dice que el grafo es finito. En este curso estudiaremos los grafos finitos.

Aristas

Si la arista carece de dirección se denota indistintamente {a,b} o {b,a}, siendo a y b los vértices que une.
Lazo: arista que une un vértice con si mismo
Arista incidente: Se dice que e es "incidente" en v si v esta en uno de los vertces de la arista
Arista múltiple: Aquella que une los mismos vértices que alguna otra.

Vértices

Vértices adyacentes: Se dice que 'v,w son adyacentes' si $\exists$ e={v,w}$\in$E (o sea, existe una arista entre los 2)
Un vértice es adyacente a si mismo si tiene lazo.
Grado de un vértice '∂': Es el nº de aristas que inciden en él. Por ejemplo, un lazo aumenta el grado en 2.
Depende solo de la estructura matemática, (los isomorfos tienen el mismo).
Vértice de aristas múltiples: Es aquel que tiene más de un arista.
Se dice que un vértice es '*par*' o '*impar*' según lo sea su grado.

Camino (o trayectoria)

Para x,y$\in$V, se dice que hay un camino en G de *x* a *y* si existe una sucesión finita no vacía de aristas distintas que contengan a vx y vy en su primer y último termino. Así: {vx,v1},{v2,v3},...,{vn,vy}

- El nº de aristas de un camino se llama *longitud* del camino.
- Si los vértices no se repiten es un camino *propio* o *simple*.
- Si hay un camino no simple entre 2 vertices, también habrá un camino simple entre ellos.
- Cuando vértice de llegada=vértice de salida, el camino se llama *circuito*, *ciclo*, o *camino cerrado*.
- Un circuito es *propio* o *simple* si solo se repiten el primer y último vértice. En estos apuntes los circuitos seran simples si no se indica lo contrario
- *Vértices accesibles:* son aquellos entre los que existe un camino. Todo vértice es accesible respecto a si mismo. La accesibilidad entre vértices es una relación de equivalencia cuyas clases son las componentes conexas de G.

Si el grado de cualquier vértice de un grafo $\geq 2 \Rightarrow$ el grafo tiene un circuito.

Grafo simple: Aquel que no tiene lazos ni aristas múltiples
Propiedades de un grafo G(V,E):

- Como cada arista incide en 2 vertices o 2 veces en el mismo vertice si es un lazo, tenemos que: Suma de los grados de todos los vertices es = doble de las aristas: $\sum_{v \in V} \partial v = 2|E|$
- Demostración: Al realizar la suma de los grados de todos los vertices, ya que cada arista tiene 2 extremos se cuenta exactamente 2 veces.

- En un grafo finito existe un nº par (o cero) de vértices de grado impar.

- En general V dividido en: $V_1=\{v' \in V / \partial' v=\text{impar}\}$, $V_2=\{v'' \in V / \partial v''=\text{par}\}$, $V_1 \cup V_2=V$; $V_1 \cap V_2=\varnothing$

- Demostración: Sabemos que $\sum_{i=1}^{p} \sigma v_i = 2|E|$ para $V=\{v_1, ..., v_p\}$. Sean $v_1, ..., v_t$ los vertices de grado impar y $v_{t+1}, ..., v_p$ los de grado par

- $\sum_{i=1}^{t} \sigma v_i + \sum_{i=t+1}^{p} \sigma v_i = 2|E|$ par+impar=impar, asi que debe ser nº de vertices impares=0

- Sabemos que σv_i es impar para i=t+1, ..., p, por lo que podemos expresarlo como $2n_i+1$ para algun n_i

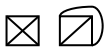
Grafo regular: Aquel con el mismo grado en todos los vértices. Si ese grado es k se, llamara *k-regular.*

Grafo bipartito: Es aquel con cuyos vértices pueden formarse dos conjuntos disjuntos de modo que no haya adyacencias entre vértices pertenecientes al mismo conjunto

Grafo completo o conexo: Aquel con una arista entre cada par de vértices, (todos estan conectados con todos).

Dos grafos completos con mismo |v| son isomorfos. Un grafo completo con n vértices se denota K_n.
Todo grafo completo es regular pq. cada vértice tiene grado |v|-1 al estar conectado con todos los otros vértices.
Un grafo regular no tiene porque ser completo.
Un grafo bipartito regular se denota $K_{m,n}$ donde m, n es el grado de cada conjunto disjunto de vértices.

Complementario de un grafo G:
Es el grafo G´que tiene conectados los vértices no conectados de G y desconectados los conectados.
Si dos grafos son complementarios, sus isomorfos también. Un grafo+su complementario = grafo completo.

Grafo plano: Aquel que admite una representación bidimensional sin que se crucen sus aristas.
En este ejemplo, vemos un grafo plano con su representación plana: 

Grafo pesado o *grafo etiquetado* - Aquel grafo cuyas aristas tienen todas un nº real positivo que sera su *peso* o *longitud*. El peso del grafo sera el sumatorio de los pesos de las aristas.
Si todas las etiquetas valen 1, la definición de longitud del camino de un grafo pesado coincide con la definición de longitud del camino a un grafo.

Grafo conexo
Grafo conexo: Aquel en el ∃ un cámino entre cualquier par de vertices.
Componente conexa de G:
Def.: Un subgrafo conexo de G que no es subgrafo propio? de ninguna componente conexa de G.
Otra def.: Subgrafo de G de forma que ningún otro vértice∈G esta conectado con vértice alguno de G´
Otra def.: Son las clases de equivalencia de estar conectado.

Subgrafo de G=(V,E) es $G'(V',E') / V' \subset V$ y $E' \subset E$ (el grafo que se obtiene borrando alguna arista o vértice de G)

Multigrafo: Grafo que tiene alguna arista múltiple. Un multigrafo se transforma en grafo añadiendo un vertice en mitad de cada arista multiple.

Pseudografo: Grafo con algún lazo.

Digrafo: Grafo con todas sus aristas dirigidas. Por tanto, los pares de vértices que definen las aristas, son pares ordenados.

Isomorfismo de grafos:
Dados G=(V,E) y G´=(V´,E´), se denomina ‘ isomorfismo de G a G´ ‘ a la aplicación biyectiva f tal que para a,b∈V, {a,b}∈E ⇔ se cumple {f(a),f(b)}∈E´. Es decir, la aplicación que relaciona biyectivamente pares de vertices de E con pares de vertices de E´, de modo que los vértices conectados siguen estandolo.

Se cumple que σa=σf(a)

Isomorfismo es la biyección que mantiene la adyacencia de vertices

- G y G´ se denominan isomorfos, y son matemáticamente iguales, solo varia la apariencia, o sea, que se mantienen las adyacencias, estructura, caminos, ciclos, nº de vértices, nº de aristas, etc.
- Si dos grafos son isomorfos, sus complementarios también.
- Se llama *automorfismo* al isomorfismo de un grafo en si mismo. Un conjunto de automorfismos, será por tanto, un conjunto de grafos isomorfos.

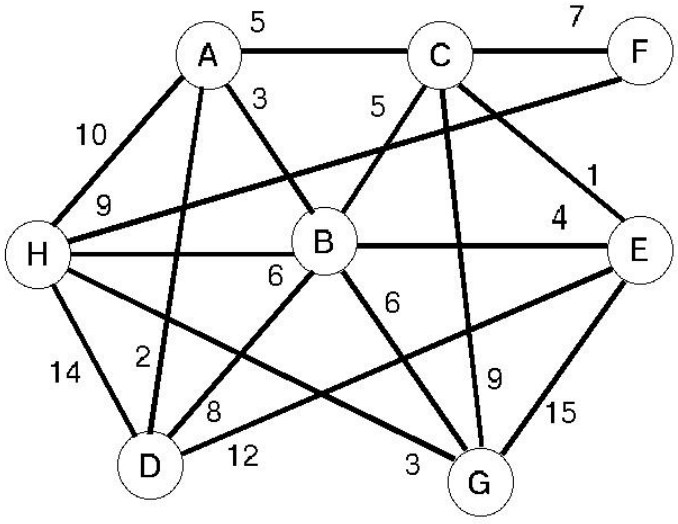
Dos grafos son isomorfos ⇔ tienen mismo número de vertices y el número de vertices con un grado dado es el mismo en los dos grafos.

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.

4.- COMPETENCIAS ESPECÍFICAS

Considerando la siguiente figura de un grafo no dirigido, en una hoja de cálculo electrónica, calcular la matriz de adyacencia y la matriz de caminos de longitud 4. ¿Cuántos caminos de longitud 5 tengo para ir del nodo H al nodo F, cuáles son esos caminos?



5. RESULTADOS

Representar un grafo utilizando una hoja electrónica de cálculo, y obtener el número de caminos de longitud n mediante el cálculo correspondiente. Después de obtener los datos de la hoja de cálculo, realizar un reporte con las operaciones que se tuvieron que realizar, muestra los caminos en una matriz indicando qué nodo se conecta con cual.

6. CONCLUSIONES

Con ésta práctica, el alumno entenderá mejor las aplicaciones que se le pueden dar a los grafos, como lo es encontrar el número máximo o mínimo de caminos que nos pueden llevar de una ciudad a otra ciudad.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 11

GRAFOS Y SUS OPERACIONES

Observaciones: Esta práctica incluye a la Práctica # 11 del temario de Matemáticas Discretas que dice "Mediante software disponible para el alumno, determinar características, propiedades y recorridos importantes en un grafo".

1.- OBJETIVO

Realizar la inserción de un vértice o arista y la eliminación de un vértice o arista en los grafos para identificar las características de los grafos..

2.- MARCO TEÓRICO

Operaciones básicas de los grafos

En los grafos, como en todas las estructuras de datos, las dos operaciones básicas son insertar y borrar. En este caso, cada una de ellas se desdobla en dos, para insertar/eliminar vértices e insertar/eliminar aristas.

Insertar vértice

La operación de inserción de un nuevo vértice es una operación muy sencilla, únicamente consiste en añadir una nueva entrada en la tabla de vértices (estructura de datos que almacena los vértices) para el nuevo nodo. A partir de ese momento el grafo tendrá un vértice más, inicialmente aislado, ya que ninguna arista llegará a él.

Insertar arista

Esta operación es también muy sencilla. Cuando se inserte una nueva arista en el grafo, habrá que añadir un nuevo nodo a la lista de adyacencia (lista que almacena los nodos a los que un vértice puede acceder mediante una arista) del nodo origen, así si se añade la arista (A,C), se deberá incluir en la lista de adyacencia de A el vértice C como nuevo destino.

Eliminar vértice

Esta operación es inversa a la inserción de vértice. En este caso el procedimiento a realizar es la eliminación de la tabla de vértices del vértice en sí. A continuación habrá que eliminar las aristas que tuviesen al vértice borrado como origen o destino.

Eliminar arista

Mediante esta operación se borra un arco del grafo. Para llevar a cabo esta acción es necesario eliminar de la lista de adyacencia del nodo origen el nodo correspondiente al nodo destino.

Otras operaciones

Las operaciones adicionales que puede incluir un grafo son muy variadas. Además de las clásicas de búsqueda de un elemento o recorrido del grafo, también podemos encontrarnos con ejecución de algoritmos que busquen caminos más cortos entre dos vértices, o recorridos del grafo que ejecuten alguna operación sobre todos los vértices visitados, por citar algunas operaciones de las más usuales.

Matriz de adyacencia*:*

Muestra adyacencias de vertices.

Se define como A=(aij)n×n (n=|V|) donde aij=1 si {vi,vj}∈E ; en caso contrario aij=0.

La matriz de adyacencia siempre es simétrica (y por tanto, no se modifica haciendo la traspuesta), porque $a_{ij} = a_{ji}$.

Para cualquier k≤n se cumple que $\sum_{i=1..n} a_{ki} = \partial v_k$ (grado de un vértice=sumatorio de la columna o fila de ese vértice).

Para un grafo G de n vértices con n>1, con A=matriz de adyacencia se cumple:

"El valor del coeficiente a_{ij}^k de la matriz A^k, es el nº de caminos de longitud k con extremos v_i y v_j"

($A^k = A \cdot A \cdot ...k \text{ veces}... \cdot A$)

Dado M= $\sum_{i=1..n} A^i$, se cumple que:
- el grafo sera conexo, si y solo si, todos los elementos de M son distintos de 0
- la diagonal de la matriz nos indica el grado de los vértices

M=Suma de matrices de adyacencia.

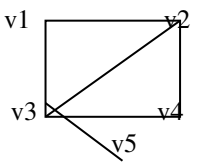
Teorema:

Sea G=(V,E), A=matriz adyacencia de G.

Si ∃ un camino de longitud m (m≥n) entre 2 vértices cualquiera, entonces ∃ un camino de longitud ≤n-1 entre esos dos vértices.

Ejemplo:

	v1	*v2*	*v3*	*v4*	*v5*
v1	0	1	1	0	0
v2	1	0	1	1	0
v3	1	1	0	1	1
v4	0	1	1	0	0
v5	0	0	1	0	0



Para comprobar si dos grafos son isomorfos, comprobamos si sus matrices quedan iguales al permutar su orden.

Ejemplo:

? Sea un grafo con matriz de adyacencia $A=\begin{pmatrix}0&3&0\\3&2&1\\0&1&0\end{pmatrix}_{3\times3}$, habra que llegar a $A^{n-1}=A^2$

$$A+A^2=\begin{pmatrix}0&3&0\\3&2&1\\0&1&0\end{pmatrix}+\begin{pmatrix}9&6&3\\6&14&2\\3&2&1\end{pmatrix}=\begin{pmatrix}9&9&3\\9&16&3\\3&3&1\end{pmatrix}$$, como $\forall\ b_{ij}\neq\varnothing$ el grafo es conexo

Matriz de incidencia:
Muestra adyacencias de aristas en vertices.
Es la matriz M de |V| filas y |E| columnas, donde mij=1 si vi es vértice de la arista ej, en caso contrario es 0.
Solo puede definirse para grados simples.

Para comprobar si un grafo es conexo:
- Se halla la matriz adyacencia de orden n×n y se eleva a la n-1 potencia
- Si todos sus elementos son ≠ 0, el grafo es conexo.

Arista de separación o puente: Aquella que al ser suprimida deja desconectados sus dos vértices.
Si e=(u,v), e∈G es un puente y G tiene k componentes conexas, G-{e} tendría k+1 componentes conexas
Punto de corte: es un vértice de un grafo conexo G que una vez suprimido convierte a G en disconexo.

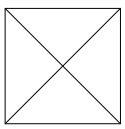
Grafo euleriano

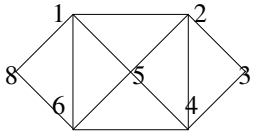
Camino euleriano es el camino que contiene a todas las aristas, apareciendo cada una exactamente una vez.
Circuito euleriano es un camino euleriano que comienza y acaba en el mismo vertice.
El grafo que admite algun circuito euleriano se llama grafo euleriano.

Grafos eulerianos

Grafo eulerianos: grafo con un circuito que contiene todas las aristas sin que se repitan. El grafo será *semieuleriano* si la trayectoria no es cerrada. Las trayectorias correspondientes se llaman eulerianas y semiulerianas.

Ejemplo:





El primero no es euleriano ni semieuleriano,
El segundo es euleriano

Lema: Si el grado de cualquier vértice de un grafo ≥2 ⇒ el grafo tiene un circuito.
Demostración:
Pueden darse 2 casos:
a) *G conexo*. Si G no tuviera circuitos ⇒ G sería un árbol ⇒ |V|=|E|+1
Pero $\sum\partial(v\in V)\geq2|V|$ ⇒ |V|≠|E|+1 ⇒ no es un árbol ⇒ tiene algun circuito.
b) *G no conexo*. Aplicamos a) a sus componentes conexas.

Teorema: Un grafo conexo G=(U,E) es euleriano ⇔ todo vértice tiene grado par.
Demostración:

"⇐" (por inducción en |E|=m)

a) Base de inducción |E|=1. Al ser |E|=1 el grafo es euleriano

b) Suponemos que el teorema es cierto para grados en las mismas condiciones y con menos de m aristas. Tenemos grafo G con todos los vértices de grado par≠0, es decir ≥2. Dado que G es conexo ⇒ ∀v ∈ V, ∂v>1 (porque existe un circuito euleriano). En cualquier caso ∂v≥2 ⇒ ∃ x circuitos en G. Suponiendo

b_1) En x están todas las aristas de G una vez → circuito euleriano →G euleriano

b_2) En x no están todas las aristas → ??

Los grafos bipartitos completos son eulerianos si son pares los bipartitos m,n.

Teorema de Euler

Si un grafo admite un camino euleriano, o todos sus vértices son pares (camino cerrado) o 2 de ellos son impares (camino abierto)

Hamilton

Camino Hamiltoniano: Es aquel que recorre todos los vértices sin pasar 2 veces por la misma arista. Solo puede existir en grafos simples donde no existan vértices impares.

Grafo Hamiltoniano - Aquel que admite un camino hamiltoniano.

Es *Semihamiltoniano* si tiene una trayectoria abierta y pasa una sola vez por cada uno de los vértices

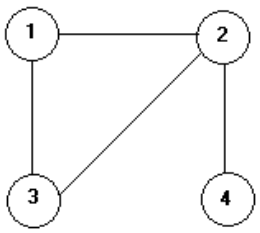
Todos los hamiltonianos son eulerianos y todos los semihamiltonianos son semieulerianos.

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.

4.- COMPETENCIAS ESPECÍFICAS

Dado el siguiente grafo, se desea eliminar la arista etiquetada por el número 2, pero debe mantenerse el nodo 4 conectado al grafo, ¿Cuál es la mejor opción para conectar al nodo 4 con el resto del grafo, justifica tu respuesta?



5. RESULTADOS

El alumno deberá entregar un nuevo grafo justificando los cambios que se han realizado o si no es posible conectar el nodo 4 al resto del grafo. La justificación será entregada en formato digital.

6. CONCLUSIONES

Al concluir ésta práctica, el alumno debe ser capaz de poder aplicar diferentes operaciones de inserción o eliminación de aristas a los grafos que se le presenten.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 12

ALGORITMO DE DIJKSTRA

Observaciones: Esta práctica incluye a la Práctica # 12 del temario de Matemáticas Discretas que dice "Desarrollar el algoritmo del camino más corto".

1.- OBJETIVO

Encontrar el camino más corto de un nodo a otro nodo mediante el algoritmo de Dijkstra

2.- MARCO TEÓRICO

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista.

La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene. El algoritmo es una especialización de la búsqueda de costo uniforme, y como tal, no funciona en grafos con aristas de coste negativo (al elegir siempre el nodo con distancia menor, pueden quedar excluidos de la búsqueda nodos que en próximas iteraciones bajarían el costo general del camino al pasar por una arista con costo negativo).

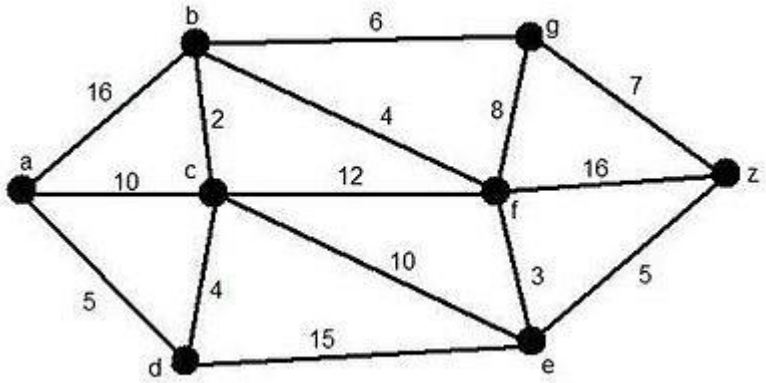
Algoritmo:
Teniendo un grafo dirigido ponderado de N nodos no aislados, sea x el nodo inicial, un vector D de tamaño N guardará al final del algoritmo las distancias desde x al resto de los nodos.

1. Inicializar todas las distancias en D con un valor infinito relativo ya que son desconocidas al principio, exceptuando la de x que se debe colocar en 0 debido a que la distancia de x a x sería 0.
2. Sea a = x (tomamos a como nodo actual).
3. Recorremos todos los nodos adyacentes de a, excepto los nodos marcados, llamaremos a estos nodos no marcados vi.
4. Si la distancia desde x hasta vi guardada en D es mayor que la distancia desde x hasta a, sumada a la distancia desde a hasta vi; esta se sustituye con la segunda nombrada, esto es:si (Di > Da + d(a, vi)) entonces Di = Da + d(a, vi)
5. Marcamos como completo el nodo a.
6. Tomamos como próximo nodo actual el de menor valor en D (puede hacerse almacenando los valores en una cola de prioridad) y volvemos al paso 3 mientras existan nodos no marcados.

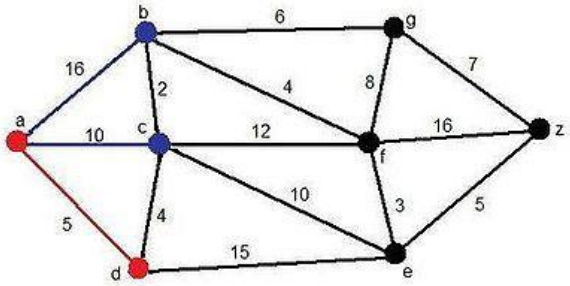
Una vez terminado al algoritmo, D estará completamente lleno.

Ejemplo:

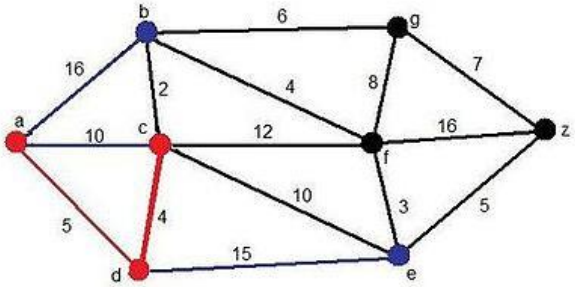
El siguiente ejemplo se desarrollará con el fin de encontrar el camino más corto desde a hasta z:



Leyenda:
Rojo: Aristas y vértices pertenecientes a la solución momentánea.
Azul: Aristas y vértices candidatos.
Paso 1

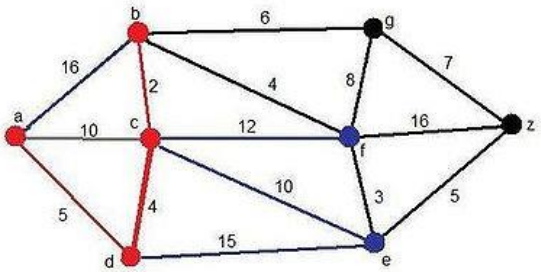


En d
Distancia:5
Paso 2

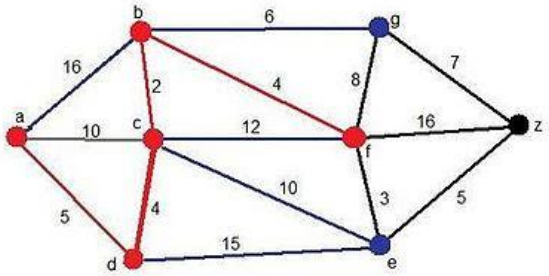


Ahora, vemos que se añade un nuevo candidato, el vértice e, y el vértice c, pero esta vez a través del d.
Pero el camino mínimo surge al añadir el vértice c.
Solución momentánea:

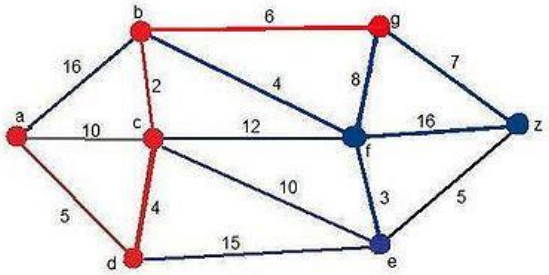
Camino: ADC
Distancia:9
Paso 3



En este paso se añade como candidato el vértice f. En este caso el camino mínimo hallado es el siguiente:
Solución momentánea:
Camino: ADCB
Distancia:11
Paso 4

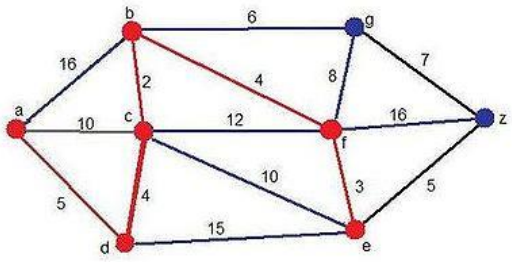


Como podemos comprobar, se han añadido un candidato nuevo, el vértice g, a través del vértice b. El mínimo camino hallado en todo el grafo hasta ahora es el siguiente:
Solución momentánea:
Camino: ADCBF
Distancia:15
Paso 5

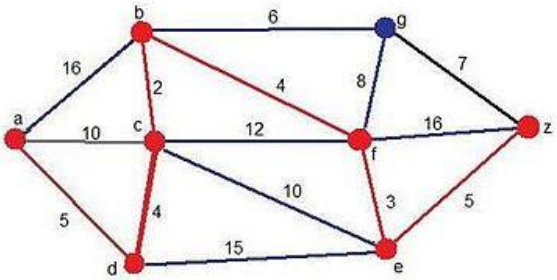


En este antepenúltimo paso, se añaden tres vértices candidatos, los vértices g, z y e. Este último ya estaba pero en esta ocasión aparece a través del vértice f. En este caso el camino mínimo, que cambia un poco con respecto al anterior, es:

Solución momentánea:
Camino: ADCBG
Distancia:17
Paso 6



En el penúltimo paso, vuelve a aparecer otro candidato: el vértice e, pero esta vez a través del vértice f. De todas formas, el camino mínimo vuelve a cambiar para retomar el camino que venía siguiendo en los pasos anteriores:
Solución momentánea:
Camino: ADCBFE
Distancia:18
Paso 7



Por fin, llegamos al último paso, en el que sólo se añade un candidato, el vértice z a través del e. El camino mínimo y final obtenido es nada mas y nada menos k:
Solución Final:
Camino: ADCBFEZ
Distancia:23

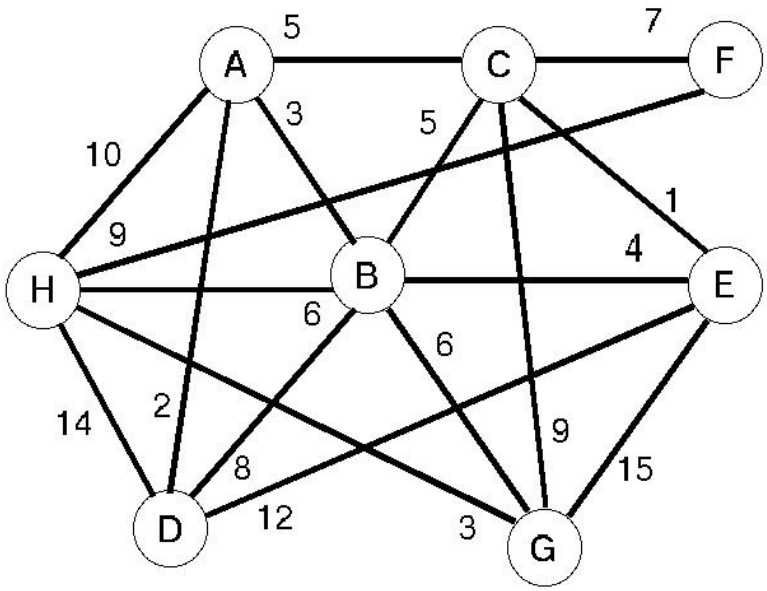
3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.

4.- COMPETENCIAS ESPECÍFICAS

Dada la siguiente figura, aplicando el algoritmo de Dijkstra, encuentra el camino más corto para llegar de:

A) H → C
B) F → B



5. RESULTADOS

Deberá mostrar los pasos que se fueron siguiendo para encontrar el camino más corto de los nodos propuestos. Los pasos se irán almacenando en un procesador de palabras así como el trazado de los caminos por los cuales se fue pasando. Una vez terminados los caminos, se deberá entregar el reporte de forma electrónica.

6. CONCLUSIONES

Cuando termine ésta práctica, el alumno aprenderá a encontrar caminos más cortos de un nodo a otro nodo con el algoritmo de Dijkstra.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 13

RECORRIDO DE ÁRBOLES

Observaciones: Esta práctica incluye a la Práctica # 13 del temario de Matemáticas Discretas que dice "Realizar el recorrido de un árbol que represente una expresión matemática y obtener su valor usando para ello el concepto de pila para almacenar resultados".

1.- OBJETIVO

Identificar la diferencia de hacer un recorrido preorder, inorder y posorder aplicados a los grafos

2.- MARCO TEÓRICO

RECORRIDO EN ARBOLES BINARIOS

Una de las operaciones mas importantes a realizar en un árbol binario es el recorrido de los mismos, recorrer significa visitar los nodos del árbol en forma sistemática, de tal manera que todos los nodos del mismo sean visitados una sola vez.

Existen 3 formas diferentes de efectuar el recorrido y todas ellas de naturaleza recursiva, estas son:

RECORRIDO PREORDEN: En el que se procesa el nodo y después se procesan recursivamente sus hijos.

RECORRIDO POSTORDEN: Donde el nodo dado se procesa después de haber procesado recursivamente a sus hijos.

RECORRIDO INORDEN: En este se procesa recursivamente el hijo izquierdo, luego se procesa el nodo actual y finalmente se procesa recursivamente el hijo derecho.

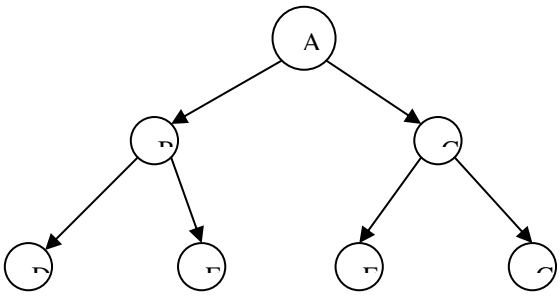
Hay un último recorrido que implementa a estos 3.

RECORRIDI POR NIVELES: Este recorrido procesa los nodos comenzando en la raíz y avanzando de forma descendente y de izquierda a derecha.

RECORRIDO PREORDEN

- VISITAR LA RAIZ
- RECORRER EL SUBARBOL IZQUIERDO
- RECORRER EL SUBARBOL DERECHO

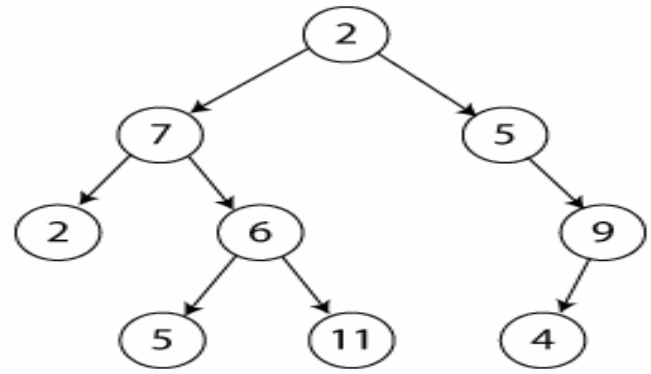
Recorrido en preorden: consiste en visitar el nodo actual (visitar puede ser simplemente mostrar la clave del nodo por pantalla), y después visitar el subárbol izquierdo y una vez visitado, visitar el subárbol derecho. Es un proceso recursivo por naturaleza.



PREORDEN: A-B-D-E-C-F-G

RECORRIDO POSTORDEN

En este caso se trata primero el subárbol izquierdo, después el derecho y por último el nodo actual. En el árbol de la figura el recorrido en postorden sería: 2, 5, 11, 6, 7, 4, 9, 5 y 2.



RECORRIDO INORDEN

En este caso se trata primero el subárbol izquierdo, después el nodo actual y por último el subárbol derecho. En un AB este recorrido daría los valores de clave ordenados de menor a mayor. En el árbol de la figura el recorrido en inorden sería: 2, 7, 5, 6, 11, 2, 5, 4 y 9.

RECORRIDO POR NIVELES

Concluimos implementando el recorrido por niveles. este recorrido procesa los nodos comenzando en la raíz y avanzando en forma descendente y de izquierda a derecha.
El nombre se deriva del hecho de que primero visitamos:

- los nodos del nivel 0 (la raíz),
- después los del nivel 1 (los hijos de la raíz),
- los del nivel 2 (los nietos de la raíz),
- y asi sucesivamente.

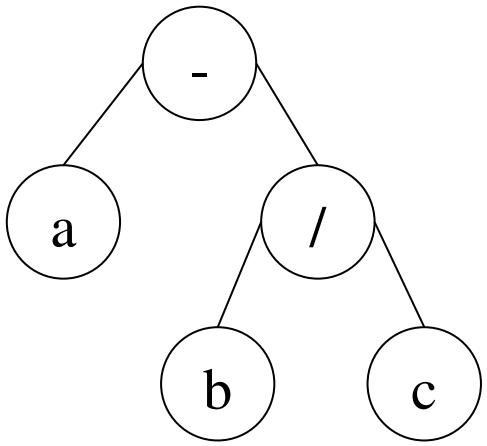
Un recorrido por niveles se implementa usando una cola en lugar de una pila. La cola almacena los nodos que van a ser visitados. Cuando se visita un nodo, se colocan sus hijos al final de la cola, donde serán visitados después de los nodos que ya están en la cola. Es fácil ver que esto garantiza que los nodos se visitan por niveles.

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.

4.- COMPETENCIAS ESPECÍFICAS

Sea el siguiente árbol, realizar los recorridos preorder, inorder y posorder; en un procesador de palabras, ir indicando con flechas y etiquetas los nodos que se visitan desde el inicio hasta el fin, también colocar el resultado de evaluar la expresión en cada recorrido. ¿Qué recorrido permite que el orden de los operadores sea primero división y luego diferencia y, cuál de los recorrido ha ce que primero se realice la resta y luego la división?, ¿Para qué nos puede servir cada uno de los recorridos?



5. RESULTADOS

Deberá mostrar en formato digital, los resultados parciales de cada recorrido y el resultado final de cada recorrido. El reporte deberá ser entregado al docente en formato digital.

6. CONCLUSIONES

Cuando termine ésta práctica, el alumno aprenderá diferenciar los recorridos en árboles y las aplicaciones que puede tener cada recorrido para tener los resultados esperado a la hora de evaluar una expresión matemática mediante árboles.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 14

ÁRBOLES BINARIOS

Observaciones: Esta práctica incluye a la Práctica # 14 del temario de Matemáticas Discretas que dice "Crear un árbol binario a partir de una lista de números aleatorios y llevar a cabo búsquedas y ordenamiento de dichos datos".

1.- OBJETIVO

Aplicar los ordenamientos de valores que se encuentren dentro de un árbol binario.

2.- MARCO TEÓRICO

Los árboles binarios son estructuras matemáticas que organizan un conjunto de elementos. Supondremos en este ejercicio que en un mismo árbol no puede haber elementos repetidos. Cada elemento se almacena en un nodo. Algunos de los nodos pueden estar relacionados, y son estas relaciones las que definen el árbol. Definimos un árbol binario de la manera siguiente:

- El árbol vacío es un árbol binario que no contiene ningún nodo.
- Dados dos árboles binarios α y β y dado un elemento x, se puede formar un tercer árbol binario $\llcorner$ enraizando α y β con un nodo que almacene x. Diremos que el nuevo nodo que almacena x es la raíz de γ, α es el subárbol izquierdo de γ, y β el subárbol derecho. Diremos también que la raíz de α es el hijo izquierdo de la raíz de $\sqcup$, la raíz de β es el hijo derecho de la raíz de γ y, por consiguiente, la raíz de γ es el padre de las raíces de ambos subárboles.

Un árbol binario es un grafo conexo, acíclico y no dirigido tal que el grado de cada vértice no es mayor a 3». De esta forma sólo existe un camino entre un par de nodos.

Un árbol binario con enraizado es como un grafo que tiene uno de sus vértices, llamado raíz, de grado no mayor a 2. Con la raíz escogida, cada vértice tendrá un único padre, y nunca más de dos hijos. Si rehusamos el requerimiento de la conectividad, permitiendo múltiples componentes conectados en el grafo, llamaremos a esta última estructura un bosque.

Tipos de árboles binarios.

- Un árbol binario es un árbol con raíz en el que cada nodo tiene como máximo dos hijos.
- Un árbol binario lleno es un árbol en el que cada nodo tiene cero o dos hijos.

- Un árbol binario perfecto es un árbol binario lleno en el que todas las hojas (vértices con cero hijos) están a la misma profundidad (distancia desde la raíz, también llamada altura).
- A veces un árbol binario perfecto es denominado árbol binario completo. Otros definen un árbol binario completo como un árbol binario lleno en el que todas las hojas están a profundidad n o n-1, para alguna n.

Un árbol binario es un árbol en el que ningún nodo puede tener más de dos subárboles. En un árbol binario cada nodo puede tener cero, uno o dos hijos (subárboles). Se conoce el nodo de la izquierda como hijo izquierdo y el nodo de la derecha como hijo derecho.

3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.

4.- COMPETENCIAS ESPECÍFICAS

Supongamos que tenemos una función valor tal que dado un valor del tipo char (una letra del alfabeto), devuelve un valor entero asociado a dicho identificador. Supongamos también la existencia de un árbol de expresión T suyos nodos hoja son letras del alfabeto y cuyos nodos interiores son los caracteres +, - , * y /. Diseñar una función que tome como parámetros un nodo y un árbol binario y devuelva el resultado entero de la evaluación de la expresión representada.

Genera un reporte electrónico con los resultados obtenidos..

5. RESULTADOS

El alumno deberá entregar de manera digital el reporte de la asignación. Y deberá mostrarse el árbol binario construido.

6. CONCLUSIONES

Al finalizar la práctica, el alumno será capaz de insertar y buscar valores de forma más rápida dentro de un árbol binario.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

Práctica 15

ÁRBOLES Y SUS OPERACIONES

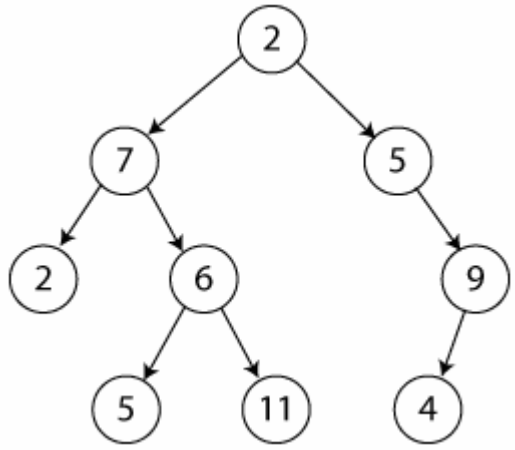
Observaciones: Esta práctica incluye a la Práctica # 15 del temario de Matemáticas Discretas que dice "Usar software disponible para el estudiante, con el cual se simule el recorrido, búsqueda de información, representación y evaluación de un árbol".

1.- OBJETIVO

Evaluar el recorrido de un árbol mediante software especializado.

2.- MARCO TEÓRICO

Recorridos sobre árboles binarios



Recorridos en profundidad
El método de este recorrido es tratar de encontrar de la cabecera a la raíz en nodo de unidad binaria. Ahora pasamos a ver la implementación de los distintos recorridos:

Recorridos en amplitud (o por niveles)
En este caso el recorrido se realiza en orden por los distintos niveles del árbol. Así, se comenzaría tratando el nivel 1, que sólo contiene el nodo raíz, seguidamente el nivel 2, el 3 y así sucesivamente. En el árbol de la figura el recorrido en amplitud sería: 2, 7, 5, 2, 6, 9, 5, 11 y 4.

Al contrario que en los métodos de recorrido en profundidad, el recorrido por niveles no es de naturaleza recursiva. Por ello, se debe utilizar una cola para recordar los subárboles izquierdos y derecho de cada nodo.

El esquema algoritmo para implementar un recorrido por niveles es exactamente el mismo que el utilizado en la versión iterativa del recorrido en preorden pero cambiando la estructura de datos que almacena los nodos por una cola.

Creación de árboles a partir de los recorridos

Para poder dibujar un árbol binario en base a los recorridos, se necesitan por lo menos dos de los recorridos de profundidad (en caso de que no se repitan los nodos, ya que si se repiten los nodos es recomendable tener los tres recorridos), ya sean inorden y preorden o inorden y postorden, la única diferencia entre usar el recorrido en preorden o postorden es que en preorden se usa el primer nodo para encontrar la raíz y en postorden se usa el último nodo.

El método consiste en ir dividiendo los recorridos del árbol en pequeños subárboles, se va encontrando la raíz con el preorden o postorden y se divide en dos subárboles basándonos en el recorrido en inorden. En el caso de que los nodos se repitan es conveniente tener los 3 recorridos para identificar más fácilmente cuál de los nodos es la raíz actual.

Para el árbol de la figura corresponden los siguientes recorridos:

Preorden $2, 7, 2, 6, 5, 11, 5, 9, 4$
Inorden $2, 7, 5, 6, 11, 2, 5, 4, 9$
Postorden $2, 5, 11, 6, 7, 4, 9, 5, 2$

Para encontrar la raíz es necesario tener el recorrido preorden o postorden, ya que la raíz es el primer nodo o el último nodo respectivamente. En este caso la raíz es el 2.

Una vez encontrada la raíz, es necesario saber su posición en el recorrido inorden, del paso anterior se tiene el nodo 2, pero existen 2 nodos con ese valor, el primero y el de en medio. Si el primer dos es la raíz, entonces no existe ninguna rama del lado izquierdo, en ese caso la siguiente raíz de acuerdo con el recorrido en postorden es 5y de acuerdo con preorden es 7, lo cual es una incongruencia, de esa forma sabemos que el otro 2es la raíz.

Entonces marcamos la raíz en el recorrido inorden:

Preorden $\underline{2}, 7, 2, 6, 5, 11, 5, 9, 4$
Inorden $2, 7, 5, 6, 11, \underline{2}, 5, 4, 9$
Postorden $2, 5, 11, 6, 7, 4, 9, 5, \underline{2}$

El recorrido inorden, es un recorrido de los árboles binarios en los que se empieza desde el nodo que se encuentra más a la izquierda de todos, sigue con la raíz y termina con los nodos del lado derecho, entonces, como en el recorrido inorden ya encontramos la raíz, la parte izquierda representa el subárbol izquierdo y la parte derecha representa el subárbol derecho.

En los recorridos tenemos 5 nodos a la izquierda del 2y a la derecha se encuentran 3 valores, entonces podemos crear los recorridos para el subárbol izquierdo y el subárbol derecho

Subárbol derecho
Preorden $7, 2, 6, 5, 11$
Inorden $2, 7, 5, 6, 11$
Postorden $2, 5, 11, 6, 7$

Subárbol derecho
Preorden $5, 9, 4$
Inorden $5, 4, 9$
Postorden $4, 9, 5$

Se sigue repitiendo el proceso hasta encontrar todos los nodos del árbol, en este punto la siguiente raíz izquierda es el 7 y la raíz derecha el 5.

Cuando se llegan a nodos en los que únicamente cuentan con una rama es necesario saber que rama es la derecha y cuál es la izquierda (para algunos árboles con balanceo como los AVL), por ejemplo siguiendo la rama de la derecha partiendo de que el 5 es la raíz el recorrido inorden es $5, 4, 9$ entonces el siguiente nodo va a la derecha, no hay nodo a la izquierda, después, los recorridos para el subárbol son:

Preorden $9, 4$
Inorden $4, 9$
Postorden $4, 9$

Finalmente el siguiente nodo se coloca a la izquierda del 9.
Este método es 100% efectivo cuando no existen nodos repetidos, cuando los nodos se repiten la complejidad aumenta para poder descubrir cuál es el nodo raíz en el recorrido inorden.

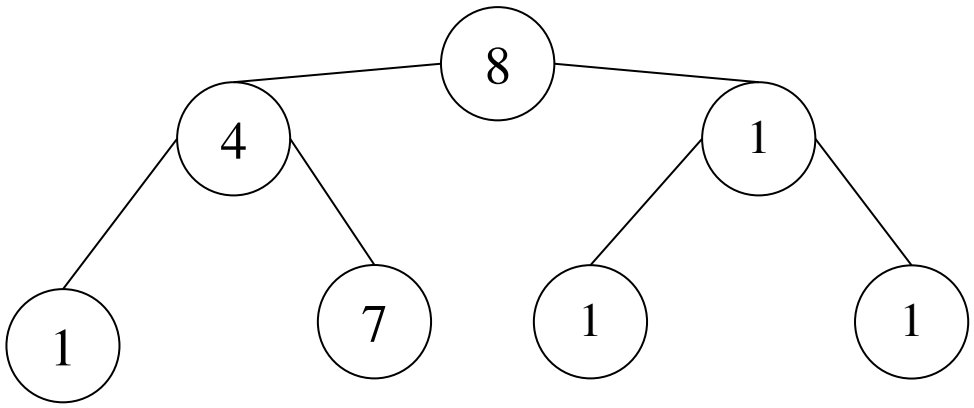
3.- MATERIAL, EQUIPO, REACTIVO o SOTFWARE A UTILIZAR

- Computadora.
- Procesador de palabras.

4.- COMPETENCIAS ESPECÍFICAS

Con el siguiente árbol, introduce los valores 6, 9, 2, 5, 10, 12 15, balancea el árbol. Posteriormente, elimina el nodo etiquetado como 8 y vuelve a balancear el árbol.

Posteriormente genera el recorrido en preorder, inorder y posorder. Escribe los resultados obtenidos por los diferentes recorridos y coloca líneas para indicar en cada recorrido, el orden en que cada nodo se fue analizando.



5. RESULTADOS

Deberá mostrar en formato digital, los cambios realizados en cada operación aplicada al árbol, también los resultados parciales de cada recorrido y el resultado final del recorrido. El reporte deberá ser entregado al docente en formato digital.

6. CONCLUSIONES

Cuando termine ésta práctica, el alumno aprenderá diferenciar los recorridos en árboles y las aplicaciones que puede tener cada recorrido para tener los resultados, así como aprenderá a balancear árboles cuando se inserta o elimina un nuevo nodo.

7.- BIBLIOGRAFÍA

1. Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.
2. Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.
3. Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.

8.- LISTA DE MATERIAL, EQUIPO O REACTIVO A UTILIZAR

FOLIO	NOMBRE DEL MATERIAL, EQUIPO O REACTIVO	CANT.	UNIDAD
1	Computadora.		
2	Hoja de cálculo.		
3	Procesador de Palabras		
4	Proyector digital		
5	Software de presentaciones electrónicas		
6	Software para dibujo (Paint, Corel Draw o algún otro software)		
7	Acceso a Internet		
8	Simulador de compuertas lógicas		

9.- LISTA DE BIBLIOGRAFÍA REQUERIDA

FOLIO	BIBLIOGRAFIA	CANT
1	Grimaldi, Ralph P. “Matemáticas discreta y combinatoria” 3ª. edición. Ed. Pearson Educación. México. 1998.	
2	Jiménez Murillo, José Alfredo. “Matemáticas para la computación”. Ed. Alfaomega. México. 2008.	
3	Lipschutz, Seymour. “Matemáticas para la Computación”. Ed. Mc-Graw Hill. Colombia. 1990.	

10.- CONTROL DE CAMBIOS DEL MANUAL DE PRÁCTICAS

DATOS GENERALES		
FECHA DE ACTUALIZACION	**ELABORÓ Y/O ACTUALIZÓ**	**DESCRIPCIÓN DE LA ACTUALIZACIÓN**
15/08/2013	☑ LIC. IVAN RAFEL SÁNCHEZ JUÁREZ	☑ SE CREÓ POR PRIMERA VEZ ESTE MANUAL DE PRÁCTICAS EN EL MES DE AGOSTO DE 2013, PARA LA DIVISIÓN DE SISTEMAS COMPUTACIONALES DEL ITSMT