Unidad VI

Teoría de Grafos

6.1 Elementos y características de los grafos.

Un grafo, G, es un par ordenado de V y A, donde V es el conjunto de vértices o nodos del grafo y A es un conjunto de pares de vértices, a estos también se les llama arcos o ejes del grafo. Un vértice puede tener 0 o más aristas, pero toda arista debe unir exactamente a dos vértices.

Los grafos representan conjuntos de objetos que no tienen restricción de relación entre ellos. Un grafo puede representar varias cosas de la realidad cotidiana, tales como mapas de carreteras, vías férreas, circuitos eléctricos, etc.

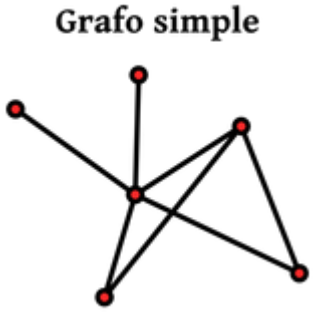
La notación *G = A (V, A)* se utiliza comúnmente para identificar un grafo.

Los grafos se constituyen principalmente de dos partes: las aristas, vértices y los caminos que pueda contener el mismo grafo.

6.1.1 Componentes de un grafo (vértices, aristas, lazos, valencia)

6.1.2 Tipos de grafos (Simples, completos, bipartidos, planos, conexos, ponderados)

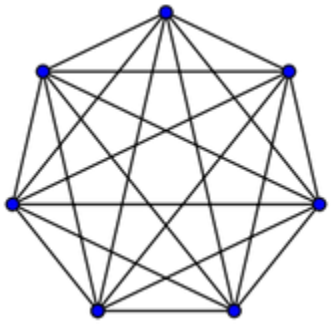
Grafos simples.- Un grafo es *simple* si a lo más existe una arista uniendo dos vértices cualesquiera. Esto es equivalente a decir que una arista cualquiera es la única que une dos vértices específicos. Un grafo que no es simple se denomina multigrafo.



Grafo completo.- Un grafo es *completo* si existen aristas uniendo *todos* los pares posibles de vértices. Es decir, todo par de vértices (a, b) debe tener una arista *e* que los une. El conjunto de los grafos completos es denominado

usualmente ***K***, siendo ***Kn*** el grafo completo de *n* vértices. Un ***Kn***, es decir, grafo completo de n vértices tiene exactamente ***n(n-1)/2*** aristas.

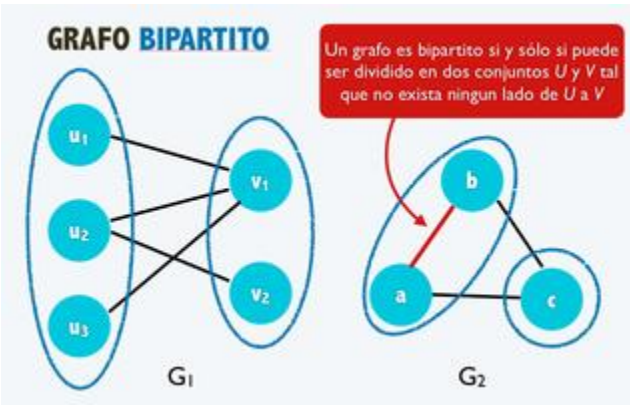
La representación gráfica de los como los vértices de un polígono regular da cuenta de su peculiar estructura.



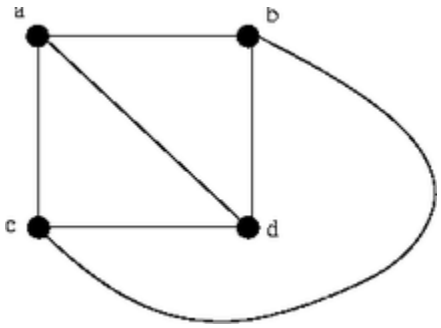
Grafos bipartitos.- Un grafo G es bipartito si puede expresarse como ***G = {V1 U V2, A}*** (es decir, sus vértices son la unión de dos grupos de vértices), bajo las siguientes condiciones:

- **V1 y V2** son disjuntos y no vacíos.
- Cada arista de A une un vértice de V1 con uno de V2.
- No existen aristas uniendo dos elementos de V1; análogamente para V2.

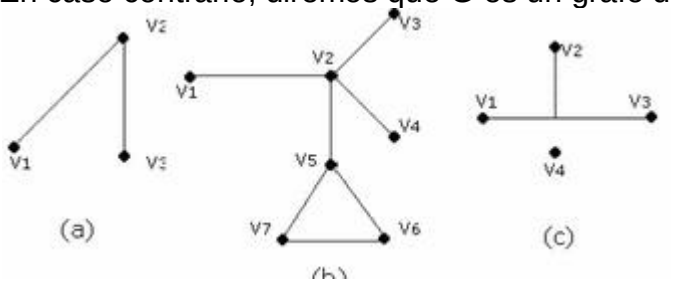
Bajo estas condiciones, el grafo se considera bipartito, y puede describirse informalmente como el grafo que une o relaciona dos conjuntos de elementos diferentes, como aquellos resultantes de los ejercicios y puzzles en los que debe unirse un elemento de la columna A con un elemento de la columna B.



Grafos Planos.- Un grafo G es **planar** si admite una representación en el plano de tal forma que las aristas no se cortan, salvo en sus extremos. A dicha representación se le denomina **grafo plano.** Se dice que un grafo es plano si puede dibujarse en el plano de manera que ningún par de sus aristas se corte. A ese dibujo se le llama representación plana del grafo.



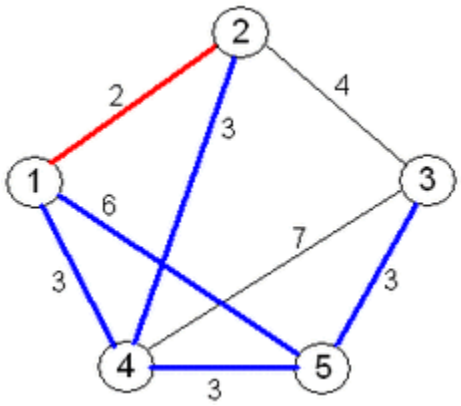
Grafo conexo.- Un grafo se dice que es conexo si cada par de sus vértices están conectados. Es decir, G es conexo $\Leftrightarrow \forall u, v : \exists \mu = [u, v]$
En caso contrario, diremos que G es un grafo desconexo.



Ejemplo
¿Cuál de los grafos siguientes es conexo?

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

Grafos ponderados.- Llamamos grafos ponderados a los grafos en los que se asigna un numero a cada una de las aristas. Este numero representa un peso para el recorrido a través de la arista. Este peso podrá indicar, por ejemplo, la distancia, el costo monetario o el tiempo invertido, entre otros. Definimos la longitud de un camino en un grafo ponderado como la suma delos pesos de las aristas de ese camino.



6.2 Representación de los grafos.

Definición.- Dado un grafo G = (V, E) con n vértices {v1, ..., vn} su **matriz de adyacencia** es la matriz de orden n×n, A(G)=(aij) donde aij es el número de aristas que unen los vértices vi y vj.
La matriz de adyacencia de un grafo es simétrica. Si un vértice es aislado entonces la correspondiente fila (columna) esta compuesta sólo por ceros. Si el grafo es simple entonces la matriz de adyacencia contiene solo ceros y unos (matriz binaria) y la diagonal esta compuesta sólo por ceros.

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

Definición .- Dado un grafo simple G = (V, E) con n=|V| vértices {v1, ..., vn} y m=|E| aristas {e1, ..., em}, su **matriz de incidencia** es la matriz de orden nxm, B(G)=(bij), donde bij=1 si vi es incidente con ej y bij=0 en caso contrario. La matriz de incidencia sólo contiene ceros y unos (matriz binaria). Como cada arista incide exactamente en dos vértices, cada columna tiene exactamente dos unos. El número de unos que aparece en cada fila es igual al grado del vértice correspondiente. Una fila compuesta sólo por ceros corresponde a un vértice aislado.

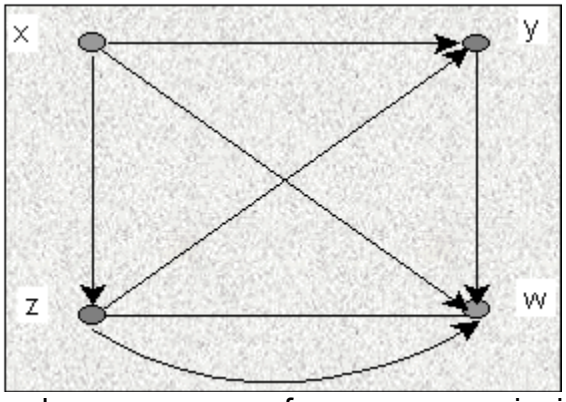
$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

6.2.1 Matemática

6.2.2.Computacional

Existen dos formas de mantener un grafo "**G**" en la memoria de una computadora, una se llama ***Representación secuencial*** de **G**, la cual se basa en la matriz de adyacencia **A**; la otra forma, es la llamada ***Representación enlazada*** de **G** y se basa en listas enlazadas de vecinos. Independientemente de la forma en que se mantenga un grafo **G** en la memoria de una computadora, el grafo **G** normalmente se introduce en la computadora por su definición formal: Un conjunto de nodos y un conjunto de aristas

Representación secuencial de un grafo :



Considere el grafo siguiente "**G**":

y suponga que los nodos se mantienen en memoria en un array DATOS tal como sigue:

DATOS: X, Y, Z, WPara hallar la matriz de adyacencia **A** del grafo "**G**", tenemos que tomar en cuenta que los nodos están normalmente ordenados de acuerdo con la forma en que aparecen en memoria; o sea, asumimos que **V1 = X, V2 = Y, V3 = Z,** y **V4 = W,** la matriz de adyacencia **A** de **G** seria la siguiente:

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

aquí **ai j = 1** si hay una arista **Vi** a **Vj ;** si no **ai j = 0**.

Así entonces para hallar la matriz de camino **P** de **G** mediante las potencias de la matriz de adyacencia **A**, como **G** tiene cuatro nodos se calcula

$$A^2, A^3, A^4, \quad \text{y} \quad B_4 = A + A^2 + A^3 + A^4$$

$$A^2 = \begin{pmatrix} 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

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

$$A^4 = \begin{pmatrix} 0 & 2 & 2 & 3 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

$$B^4 = \begin{pmatrix} 0 & 5 & 6 & 8 \\ 0 & 1 & 2 & 3 \\ 0 & 3 & 3 & 5 \\ 0 & 2 & 3 & 5 \end{pmatrix}$$

por lo tanto la matriz de caminos **P** se obtiene ahora haciendo **Pij = 1** siempre que haya una entrada positiva en la matriz **B4 .** así

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

La matriz de caminos muestra que no hay camino de **u1** a **u2** de hecho, no hay camino de ningún nodo a **u1** por tanto, **G** no es fuertemente conexo.

Representación enlazada de un grafo :

Un grafo "**G**" se guarda en memoria como sigue:

NODO A B E D C

SIG 7 4 0 6 8 0 2 3

ADY 1 2 5 7 9

1 2 3 4 5 6 7 8

PRINCIPIO = 1, NDISP = 5

DEST 2 6 4 6 7 4 4 6

ENL 10 3 6 0 0 0 0 4 0 0

1 2 3 4 5 6 7 8 9 10

ADISP = 8

Para dibujar el respectivo grafo "**G**", primero debemos buscar todos los vecinos de cada **NODO[K]** recorriendo su lista de adyacencia que tiene el puntero de adyacencia **ADY[J].** Esto da como resultado:

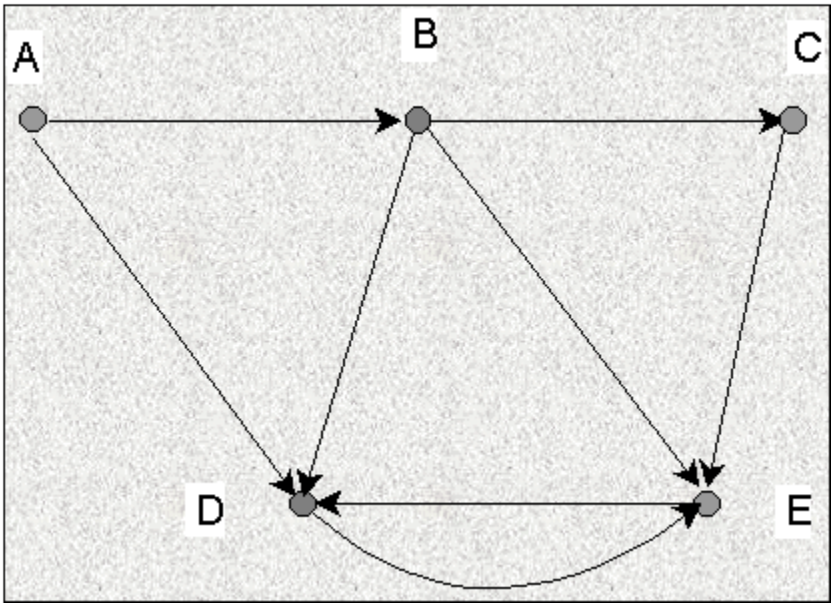
A: 2(B) y 6(D)

B: 6(D), 4(E) y 7(C)

C: 4(E)

D: 4(E)

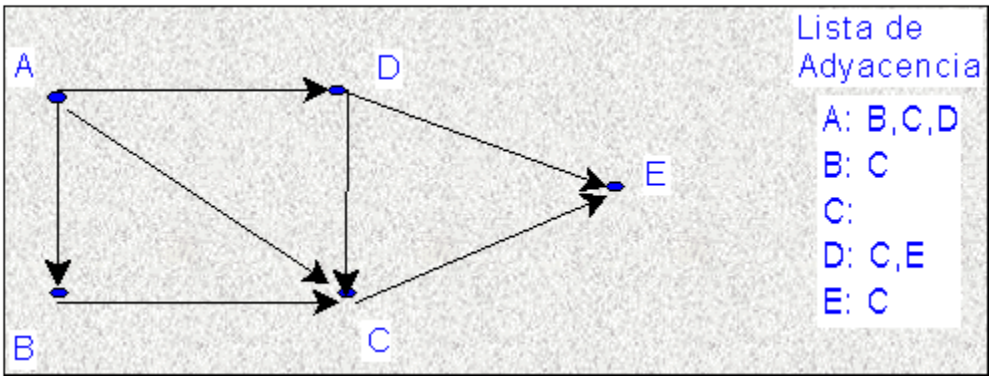
E: 6(D)



Entonces procedemos a dibujar el diagrama del grafo como sigue:

Sea **G** un grafo dirigido con m nodos. La representación secuencial de **G** en la memoria, o sea, la representación de **G** por su matriz de adyacencia **A**, tiene unas cuantas desventajas importantes.

En primer lugar, puede ser difícil insertar y eliminar nodos de **G**, esto es por que el tamaño de **A** debería ser cambiado y los nodos deberían ser reordenados, así que habría muchos cambios en la matriz **A**; más aun, si el numero de aristas es **O(m)** o **O(m log2 m)**, entonces la matriz **A**estará desperdiciada (contendrá muchos ceros); por tanto, se desperdiciará una gran cantidad de espacio; entonces **G** normalmente se representa en memoria por una representación enlazada, también llamada **estructura de adyacencia**.



6.3 Algoritmos de recorrido y búsqueda.

Recorrer un grafo significa tratar de alcanzar todos los nodos que estén relacionados con uno que llamaremos nodos de salida.existen básicamente dos técnicas para recorrer un grafo: el recorrido en anchura y el recorrido en profundidad.
Los algoritmos de búsqueda en grafos nacen por la necesidad de crear un mecanismo de navegación autónoma, bien sea de robots, coches, o personajes en un videojuego. Algunos de los mas conocidos son A*, LPA*, o D*.

6.3.1 El camino más corto

En la Teoría de grafos, el problema de los CAMINOS más cortos es el problema que consiste en encontrar un camino entre dos vértices (o nodos) de tal manera que la suma de los pesos de las aristas que lo constituyen es mínima. Un ejemplo es encontrar el camino más rápido para ir de una ciudad a otra en un mapa. En este caso, los vértices representan las ciudades, y las aristas las carreteras que las unen, cuya ponderación viene dada por el tiempo que se emplea en atravesarlas.
Formalmente, dado un grafo ponderado (que es un conjunto V de vértices, un conjunto E de aristas y una función de variable real ponderada f : E $\rightarrow$ **R**) y un elemento v $\in$ V encuentra un camino P de v a v' $\in$ V, tal que:

$$\sum_{p \in P} f(p)$$

es el mínimo entre todos los caminos que conectan v y v'.

El problema es también conocido como el problema de los caminos más cortos entre dos nodos, para diferenciarlo de la siguiente generalización:

- **El problema de los caminos más cortos desde un origen** en el cual tenemos que encontrar los caminos más cortos de un vértice origen v a todos los demás vértices del grafo.

- **El problema de los caminos más cortos con un destino** en el cual tenemos que encontrar los caminos más cortos desde todos los vértices del grafo a un único vértice destino, esto puede ser reducido al problema anterior invirtiendo el orden.
- **El problema de los caminos más cortos entre todos los pares de vértices**, el cual tenemos que encontrar los caminos más cortos entre cada par de vértices (v , v') en el grafo.

Los algoritmos de los caminos más cortos se aplican para encontrar direcciones de forma automática entre localizaciones físicas, tales como direcciones en mapas callejeros.

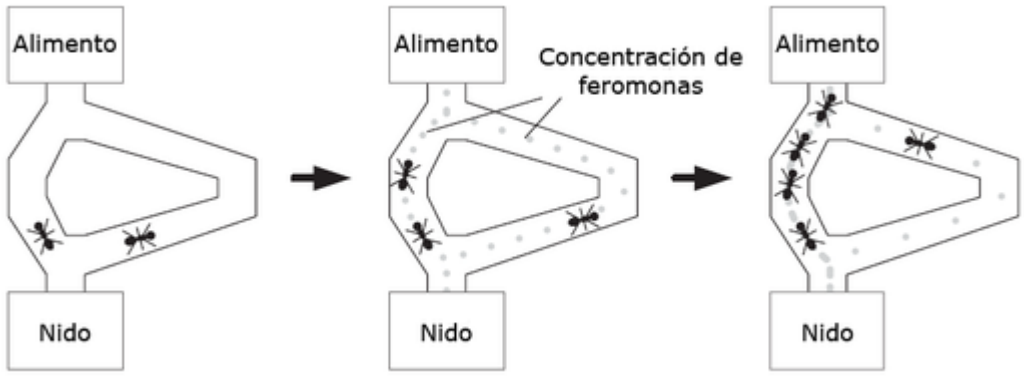
Si un algoritmo representa una máquina abstracta no determinista como un grafo, donde los vértices describen estados, y las aristas posibles transiciones, el algoritmo de los caminos más cortos se usa para encontrar una secuencia óptima de opciones para llegar a un cierto estado final o para establecer límites más bajos en el tiempo, necesario para alcanzar un estado dado. Por ejemplo, si los vértices representan los estados de un puzzle como el Cubo de Rubik, cada arista dirigida corresponde a un simple movimiento o giro. El algoritmo de los caminos más cortos se usa para encontrar la solución que utiliza el mínimo número posible de movimientos.

En el argot de las telecomunicaciones, a este algoritmo es también conocido como el problema del mínimo retraso, y con frecuencia se compara con el problema de los caminos más anchos.

Una aplicación más coloquial es la teoría de los "Seis grados de separación", a partir de la cual se intenta encontrar el camino más corto entre dos personas cualesquiera.

Otras aplicaciones incluyen la Investigación de operaciones, instalaciones y facilidad de diseño, robótica, transporte y VLSI de diseño.

Ejemplo:



6.3.2. A lo ancho

Búsqueda en anchura (en inglés *BFS - Breadth First Search*) es un algoritmo para recorrer o buscar elementos en un grafo (usado frecuentemente sobre árboles). Intuitivamente, se comienza en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo) y se exploran todos los vecinos de este nodo. A continuación para cada uno de los vecinos se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el árbol.

Formalmente, *BFS* es un algoritmo de *búsqueda sin información*, que expande y examina todos los nodos de un árbol sistemáticamente para buscar una solución. El algoritmo no usa ninguna estrategia heurística.

Sea G = (V, A) un grafo conexo, V' = V un conjunto de vértices, A' un vector de arcos inicialmente vacío y P un vector auxiliar inicialmente vacío:

1. Se introduce el vértice inicial en P y se elimina del conjunto.

2. Mientras V' no sea vacío repetir los puntos 3 y 4. En otro caso parar.
3. Se toma el primer elemento de P como vértice activo.
4. Si el vértice activo tiene algún vértice adyacente que se encuentre en V':

Se toma el de menor índice.

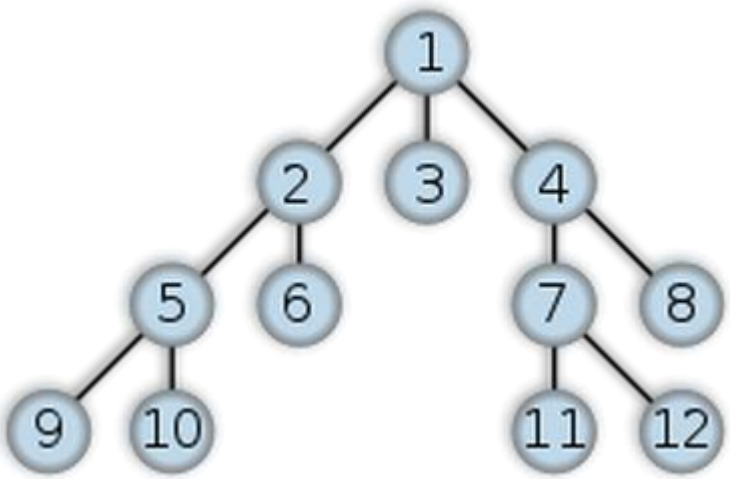
Se inserta en P como último elemento.

Se elimina de V'.

Se inserta en A' el arco que le une con el vértice activo.

Si el vértice activo no tiene adyacentes se elimina de P.

Ejemplo:



6.3.3 En profundidad

Una **Búsqueda en profundidad** (en inglés DFS o *Depth First Search*) es un algoritmo que permite recorrer todos los nodos de un grafo o árbol (teoría de grafos) de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa (Backtracking), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

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

- **ALGORITMO BEP**:

Sea G = (V, A) un grafo conexo, V' = V un conjunto de vértice, A'un vector de arcos inicialmente vacío y P un vector auxiliar inicialmente vacío:

1. Se introduce el vértice inicial en P y se elimina del conjunto V'.
2. Mientras V' no sea vacío repetir los puntos 3 y 4. En otro caso parar.
3. Se toma el último elemento de P como vértice activo.
4. Si el vértice activo tiene algún vértice adyacente que se encuentre en V':

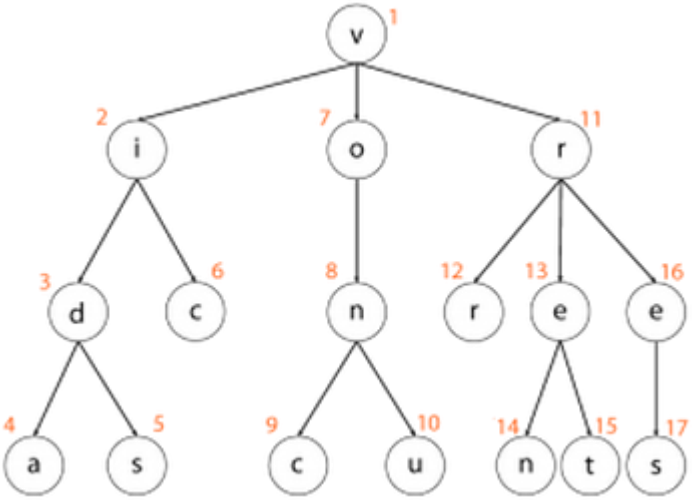
Se toma el de menor índice.

Se inserta en P como último elemento.

Se elimina de V'.

Se inserta en A' el arco que le une con el vértice activo.

Si el vértice activo no tiene adyacentes se elimina de P.



En la siguiente figura mostramos el orden de visita, siendo los números en naranja dicho orden:

6.4 Arboles.

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

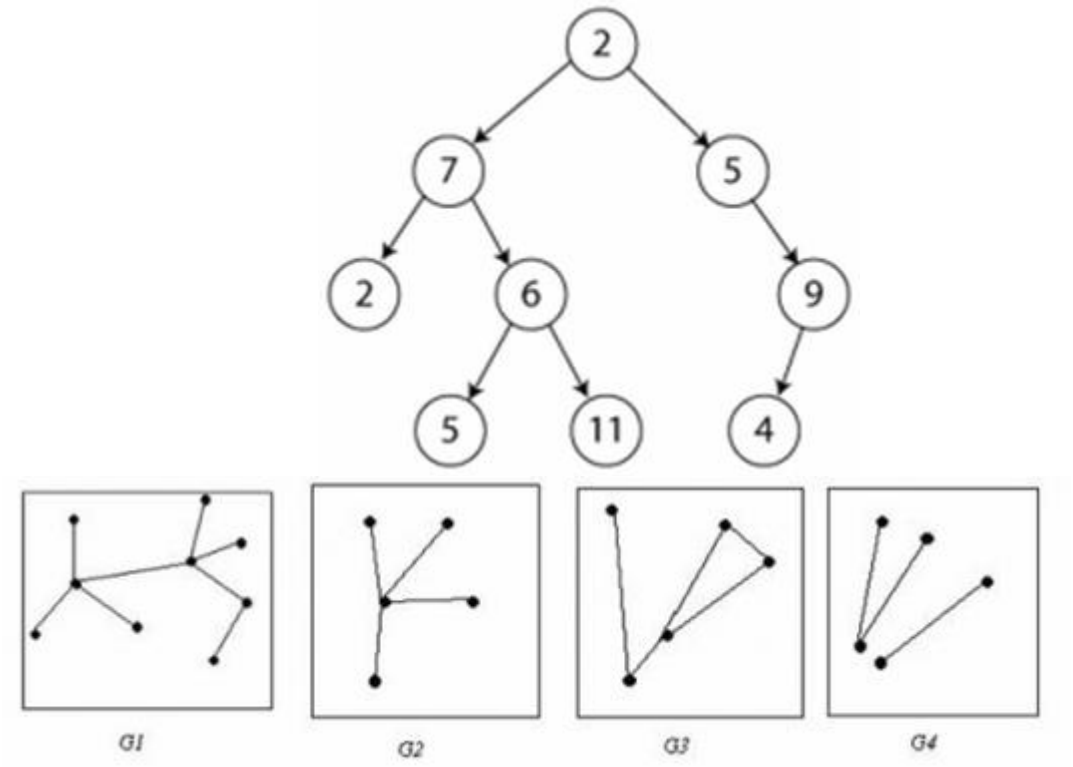
Formalmente se define un árbol de tipo T como una estructura homogénea que es la concatenación de un elemento de tipo T junto con un número finito de árboles disjuntos, llamados subárboles.

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

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

ARBOL, si es conexo y no contiene ciclos.

Ejemplo de un árbol:



Otros ejemplos de árbol:

En donde: Los grafos G1 y G2 son árboles, mientras que los grafos G3 y G4 no lo son.

6.4.1 Componentes (raíz, hoja, padre, hijo, descendientes, ancestros)

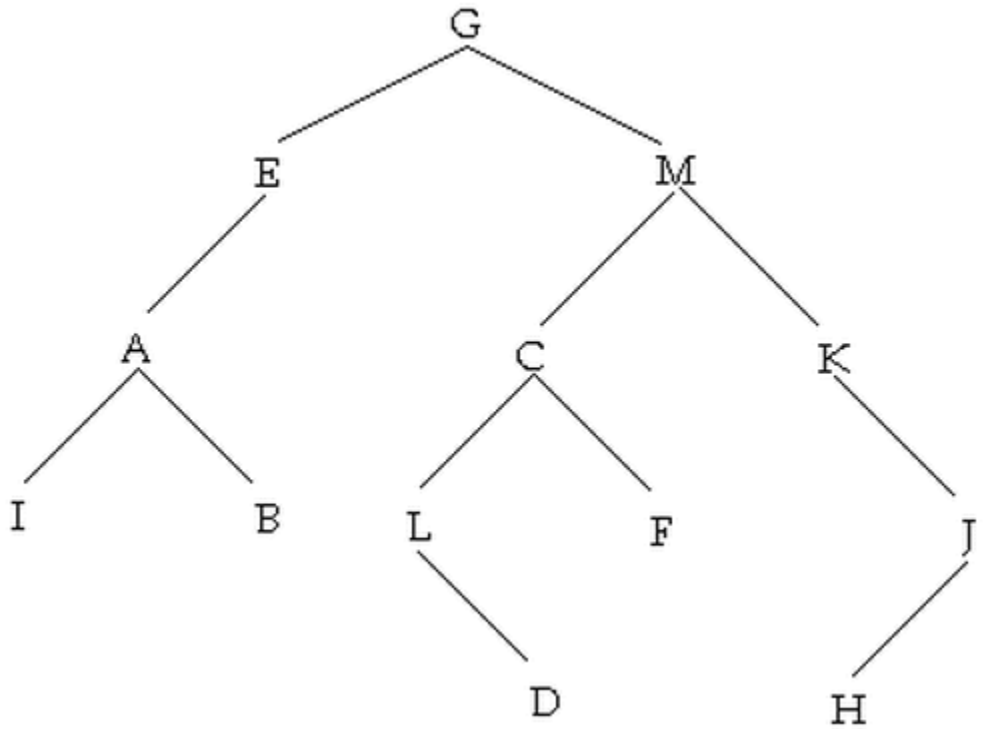
Un árbol está divido en tres subconjuntos separados.

- El primer subconjunto contiene un único elemento llamado **raíz** del árbol.
- Los otros 2 subconjuntos son por si mismos árboles binarios y se les conoce como**subárboles izquierdo** y **derecho** del árbol original. Cada elemento de un árbol binario se denomina **nodo**. La ausencia de una ramificación indica un subárbol vacio.
- Si *A* es la raíz de un árbol binario y *B* es la raíz de su subárbol izquierdo o derecho, se dice que *A* es el **padre** de *B* y se dice que *B* es el **hijo izquierdo** o**derecho** de *A*.
- Un nodo que no tiene hijos se denomina **hoja**. El nodo *n*1 es un **ancestro** del nodo *n*2 (y *n*2 es un **descendiente** de *n*1) si *n*1 es el

padre de *n*2 o el padre de algún ancestro de *n*2. 2 nodos son **hermanos** si son los hijos izquierdo y derecho del mismo padre.

- Si cada nodo que no es hoja es un árbol binario tiene subárboles izquierdo y derecho que no están vacíos, el elemento se clasifica como **árbol estrictamente binario**.
- Los árboles representan las estructuras **no lineales y dinámicas** de datos más importantes en computación. **Dinámicas** porque las estructuras de árbol pueden cambiar durante la ejecución de un programa. **No lineales**, puesto que a cada elemento del árbol pueden seguirle varios elementos.
- Se utiliza la recursión para definir un árbol porque representa la forma más apropiada y porque además es una característica inherente de los mismos. Los árboles tienen una gran variedad de aplicaciones. Por ejemplo, se pueden utilizar para representar fórmulas matemáticas, para organizar adecuadamente la información, para construir un árbol genealógico, para el análisis de circuitos eléctricos y para numerar los capítulos y secciones de un libro.

Raíz: Nodo que constituye la única entrada a la estructura (por elloes necesario tener un puntero sobre él).
Ramas o Arcos: Conexión entre dos nodos del árbol querepresenta una relación de jerarquía.
Hojas: Nodo sin hijos



6.4.2 Propiedades

Un árbol es un grafo simple en el cual existe un único camino entre cada par de vértices.

Sea **G =(V,A)** un grafo no dirigido. **G** se denomina **ÁRBOL**, si es conexo con **n** nodos y **n-1** aristas y no contiene ciclos.

Formas equivalentes de definir un árbol.

1. Un árbol es un grafo conexo con n nodos y n-1 aristas.
2. Un árbol es un grafo conexo que no contiene ciclos.
3. Un árbol es un grafo con n nodos, n -1 arista y sin ciclos.
4. Un árbol es un grafo tal que entre cualquier par de nodos distintos existe un camino simple único.

Propiedades:

• Existe un único paseo entre dos vértices cualesquiera de un árbol.
• El número de vértices es mayor en uno al número de aristas de un árbol.
• Un árbol con dos o más vértices tiene al menos dos hojas.
Un árbol T (libre) es una gráfica simple que satisface lo siguiente; si v y w son vértices en T, existe una trayectoria simple única de v a w.

6.4.3 Clasificación (altura, número de nodos)

Características del árbol, en relación a su tamaño:

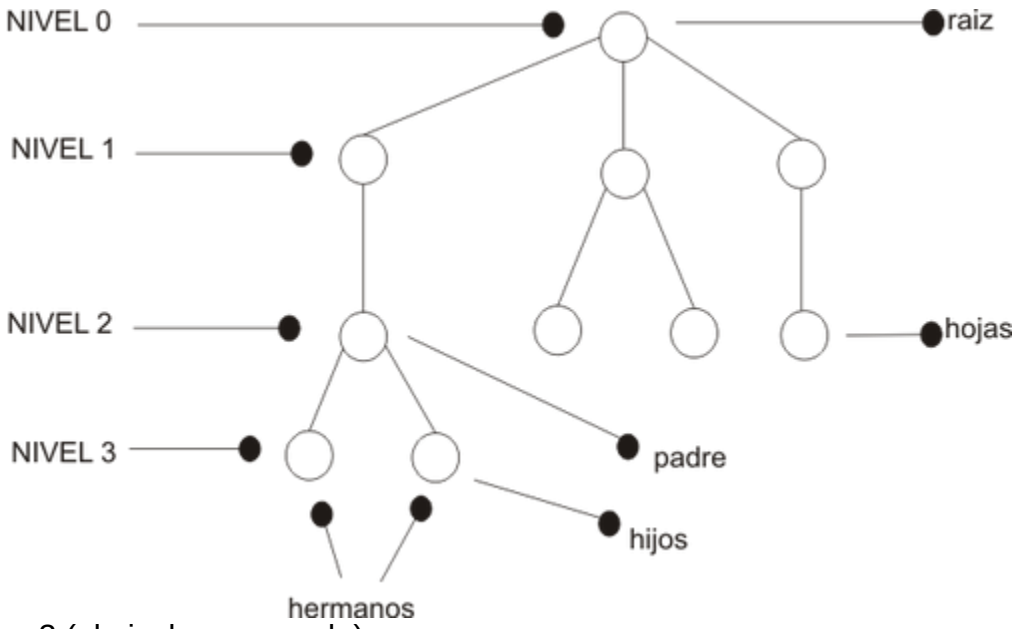
Orden: es el número potencial de hijos que puede tener cada elemento de árbol. De este modo,diremos que un árbol en el que cada nodo puede apuntar a otros dos es de orden dos, si puede apuntar a tres será de orden tres, etc.

Grado: el número de hijos que tiene el elemento con más hijos dentro del árbol. En el árbol delejemplo, el grado es tres, ya que tanto 'A' como 'D' tienen tres hijos, y no existen elementos conmás de tres hijos.

Nivel: se define para cada elemento del árbol como la distancia a la raíz, medida en nodos. El nivel de la raíz es cero y el de sus hijos uno. Así sucesivamente. En el ejemplo, el nodo 'D' tiene nivel 1, el nodo 'G' tiene nivel 2, y el nodo 'N', nivel 3.

Altura: la altura de un árbol se define como el nivel del nodo de mayor nivel. Como cada nodode un árbol puede considerarse a su vez como la raíz de un árbol, también podemos hablar dealtura de ramas. El árbol del ejemplo tiene altura 3, la rama 'B' tiene altura 2, la rama 'G' tienealtura 1, la 'H' cero, etc.

Si un grafo tiene un vértice Uo que solo contiene una diferente de Uo−U1 (a sí mismo) entonces es un árbol.



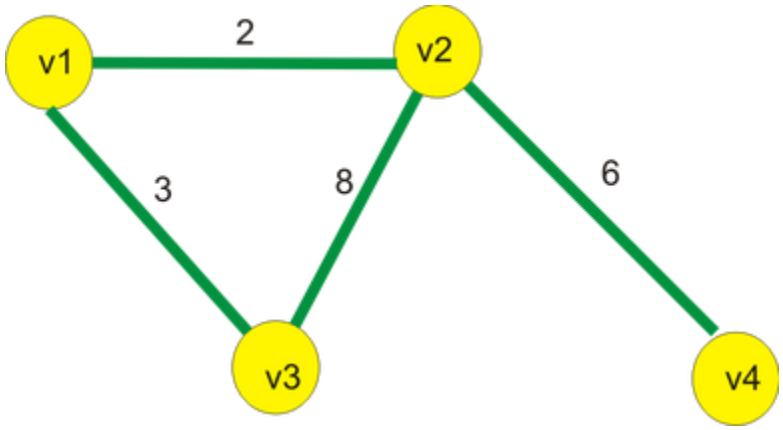
Altura = 3 (el nivel mas grande)
raíz = que no tiene padre (inicial)
padre = que tiene hijo(s)
hoja = no tiene hijo(s), tiene padre
Árbol ordenado: tiene nivel, los hijos de izquierda a derecha.
n-árbol: cuando cada padre tiene a lo más n hijos
árbol binario: cada padre tiene a lo más 2 hijos.

6.4.4 Árboles con peso

El peso de un árbol en un nodo dado es el número de nodos en el árbol sin contarse el mismo.El peso de un nodo en un árbol es la longitud del camino más largo del nodo a una hoja.
El peso de un árbol es el peso de la raíz.

Un árbol con peso es un grafo donde cada lado tiene un número asociado o peso. Normalmente, al peso de un lado e se le designa por w(e).
La suma de todos los pesos de todos los lados de un grafo con peso se llama el peso del grafo.

Ejemplo: cual es el peso de un árbol?



Peso total del grafo = 19

6.4.5 Recorrido de un árbol: Preorden, Inorden, Postorden,

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

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

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

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

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

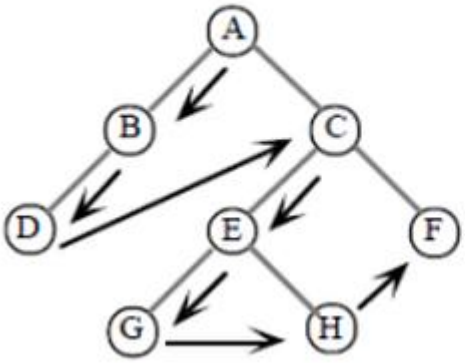
1. Atraviese el sub-árbol izquierdo
2. Atraviese el sub-árbol derecho
3. Visite la raíz

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

- En preorden, la raíz se recorre antes que los recorridos de los subárboles izquierdo y derecho
- En inorden, la raíz se recorre entre los recorridos de los árboles izquierdo y derecho, y
- En postorden, la raíz se recorre después de los recorridos por el subárbol izquierdo y el derecho

Preorden (antes), inorden (en medio), postorden (después).

- **Preorden**: primero se accede a la información del nodo, después al subárbo izquierdo y después al derecho.



A - B - D - C - E - G - H - F

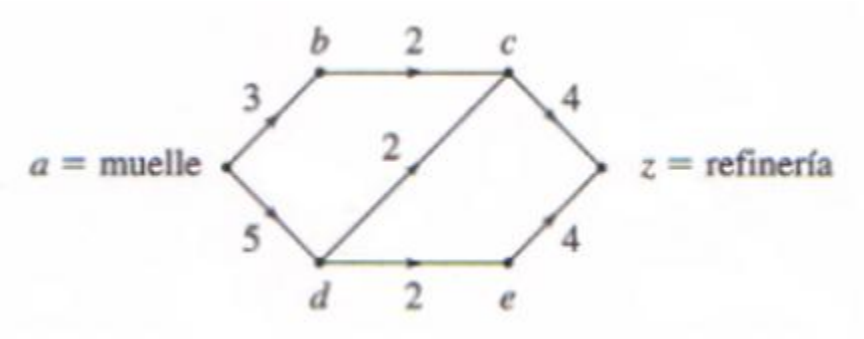
6.5 Redes.(teorema de flujo máximo, teorema de flujo mínimo, pareos y redes de Petri)

Definición: Una Red de Transporte es una gráfica dirigida, simple, con pesos y que debe cumplir las siguientes:

· Poseer una fuente o vértice fijo que no tiene aristas de entrada.
· Poseer un sumidero o vértice fijo que no tiene arista de salida
· El peso Cij de la arista dirigida de i a j llamado capacidad de “ij” es un numero no negativo.

Una red de transporte es una gráfica dirigida, simple con pesos que satisface:

a) Un vértice fijo, designado como el origen o fuente, no tiene aristas de entrada.
b) Un vértice, designado como destino o sumidero, no tiene aristas salientes.
c) El peso Cij de la arista dirigida (i, j) llamada capacidad de (i, j) es un numero no negativo.



Flujo maximo

En una red G, el flujo máximo es *un* flujo máximo. Generalmente existen varios flujos con el mismo valor máximo. Para encontrar el flujo máximo consideraremos un flujo inicial en cada arista igual a cero, después se determina un camino específico de la fuente al sumidero y se incrementa el flujo.

Si una arista esta dirigida hacia la fuente decimos que esta arista esta dirigida en forma impropia, en caso contrario esta dirigida en forma propia.

Si se determina un camino P de la fuente al sumidero en donde cada arista de P esta orientada en forma propia y el flujo en cada arista es menor que la capacidad de la arista, es posible aumentar el valor de flujo.

Es posible incrementar el flujo en ciertos caminos de la fuente al sumidero que tenga aristas orientadas en forma impropia y propia. Sea P un camino de “a” a “z” y sea “x” un vértice en P que no sea “a” ni ”z”

- Ambas aristas están orientadas en forma propia, en este caso, si incrementamos el flujo en ", el flujo en la entrada en x seguirá siendo igual al flujo de salida de x.
- Si incrementamos el flujo en e2 en ", debemos disminuir el flujo en e1 en " de modo que el flujo de entrada en x siga siendo igual al flujo de salida en x.
- Es análogo en el caso b
- Disminuimos el flujo en ambas aristas en ". En cada caso las asignaciones resultantes de las aristas dan como resultado un flujo.

Para realizar estas alteraciones debemos tener un flujo menor que la capacidad en una arista orientada en forma propia y un flujo distinto de cero en una arista orientada en forma impropia.

Teorema 2:
Sea P un camino de “a” a ”z” en una red G tal que:

- Para cada arista (i,j) de P, orientada en forma propia.

Fij <Cij

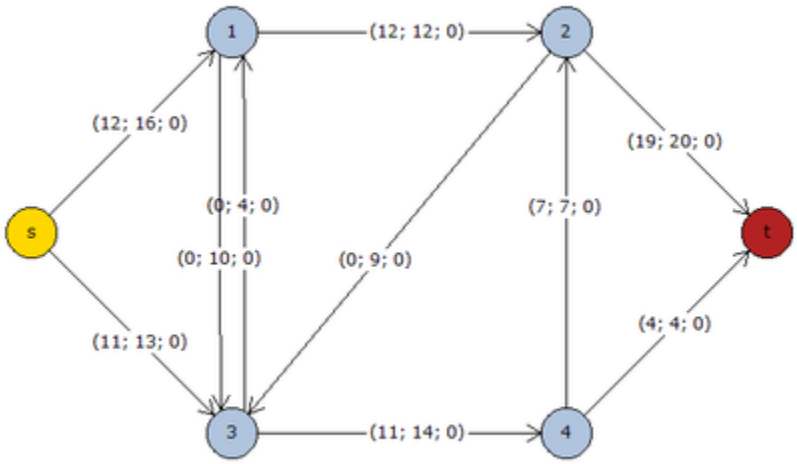
- Para cada arista (i,j) de P, orientada en forma impropia

0 <Fij
Se define
F'ij =
Si no existieran caminos que concuerden con el teorema 2, el flujo es máximo, entonces se considera el algoritmo:

- Iniciar con un flujo
- Buscar un camino que satisfaga con las condiciones del teorema 2
- Si no existe el camino el flujo es máximo.
- Se incrementa el flujo en ", y se regresa a línea 2.

A dicho algoritmo se le llama Algoritmo etiquetado.

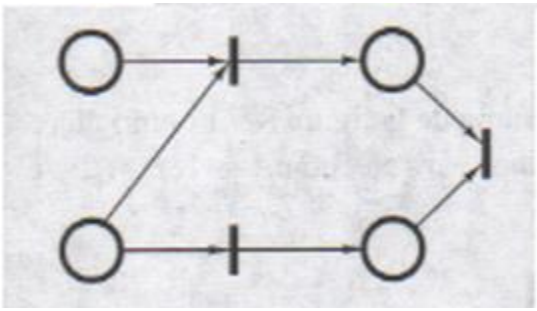
Ejemplo del Flujo Maximo.-



Flujo minimo
Pareos
Haz clic aquí para modificar.
Redes de petri

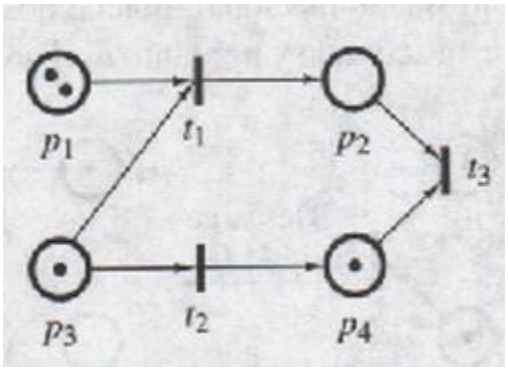
Una red de Petrí es un grafo dirigido bipartito, con un estado inicial, llamado *marcación inicial*. Los dos componentes principales de la red de Petrí son

los *sitios* (también conocidos como *estados*) y las*transiciones*. Gráficamente, los sitios son dibujados como círculos y las transiciones como barras o rectángulos. Las aristas del grafo son conocidas como *arcos*. Estos tienen un peso específico, el cual es indicado por un número entero positivo, y van de sitio a transición y viceversa. Por simplicidad, el peso de los arcos no se indica cuando éste es igual a 1. Un arco que esté etiquetado con *k* puede ser interpretado como *k* arcos paralelos.



Ejemplo de una Red de Petrí

Es una grafica dirigida G = (V, E) donde V = P U T y P "T = Ø, cualquier arista *e* en E es incidente en un miembro de P y un miembro de T, el conjunto P es el conjunto de lugares y el conjunto T es en conjunto de Transiciones.
Un *marcado* de una *Red de Petrí* asigna a cada lugar un entero no negativo, una red de Petrí con un marcado es una *Red de Petrí Marcada* (o simplemente una Red de Petrí).
Con un marcado se asigna al valor no negativo *n* al lugar *p*, decimos que existen n elementos en p, mediante los elementos a representar son los puntos.



Los lugares representan condiciones, las transiciones representan eventos, y la presencia de al menos un elemento en un lugar (condición) indica que tal condición se cumple.

6.6 Aplicaciones de grafos y árboles.

Gracias a la teoría de grafos se pueden resolver diversos problemas como por ejemplo la síntesis de circuitos secuenciales, contadores o sistemas de apertura. Se utiliza para diferentes áreas por ejemplo, Dibujo computacional, en toda las áreas de Ingeniería.

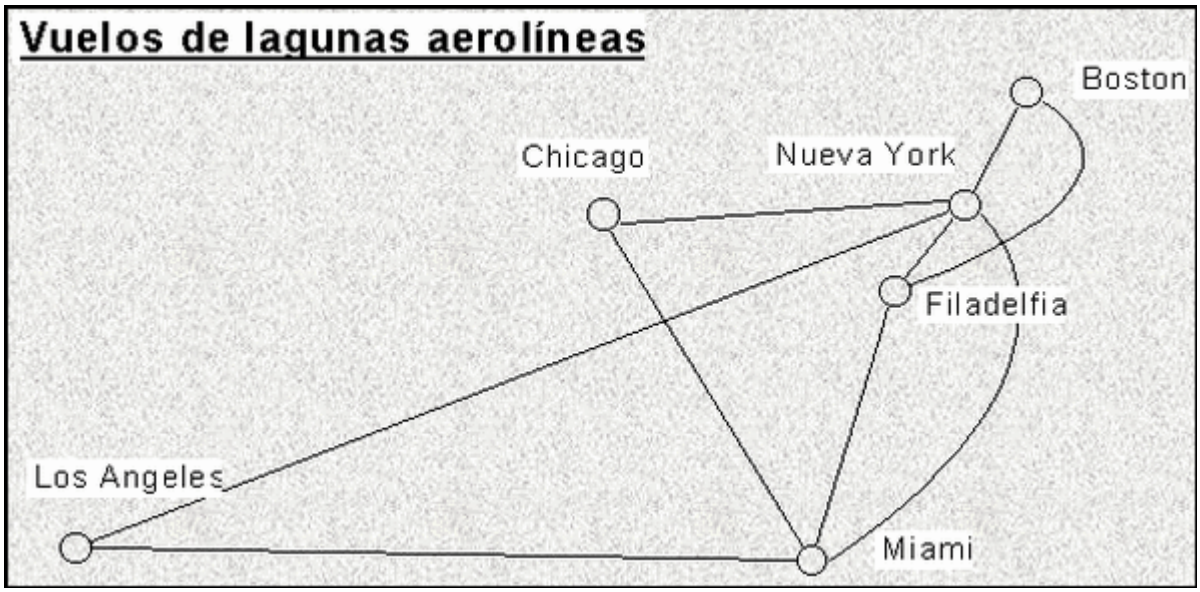
Los grafos se utilizan también para modelar trayectos como el de una línea de autobús a través de las calles de una ciudad, en el que podemos obtener caminos óptimos para el trayecto aplicando diversos algoritmos como puede ser el algoritmo de Floyd.

Para la administración de proyectos, utilizamos técnicas como PERT en las que se modelan los mismos utilizando grafos y optimizando los tiempos para concretar los mismos.

La teoría de grafos también ha servido de inspiración para las ciencias sociales, en especial para desarrollar un concepto no metafórico de red social que sustituye los nodos por los actores sociales y verifica la posición, centralidad e importancia de cada actor dentro de la red. Esta medida permite cuantificar y abstraer relaciones complejas, de manera que la estructura social puede representarse gráficamente. Por ejemplo, una red social puede representar la estructura de poder dentro de una sociedad al identificar los vínculos (aristas), su dirección e intensidad y da idea de la manera en que el poder se transmite y a quiénes.

Los grafos son importantes en el estudio de la biología y hábitat. El vértice representa un hábitat y las aristas (o "edges" en inglés) representa los senderos de los animales o las migraciones. Con esta información, los científicos pueden entender cómo esto puede cambiar o afectar a las especies en su hábitat.

Por ejemplo, supongamos que unas líneas aéreas realizan vuelos entre las ciudades conectadas por líneas como se ve en la figura anterior (más adelante se presentaran grafos con estructuras de datos); la estructura de datos que refleja esta relación recibe el nombre de grafo.



Algunos ejemplos son los siguientes:

Sociograma de una red social

Isomeros

Organigramas

Arquitectura de redes detelefonía móvil

Draws de eliminación directa (ej: tenis)

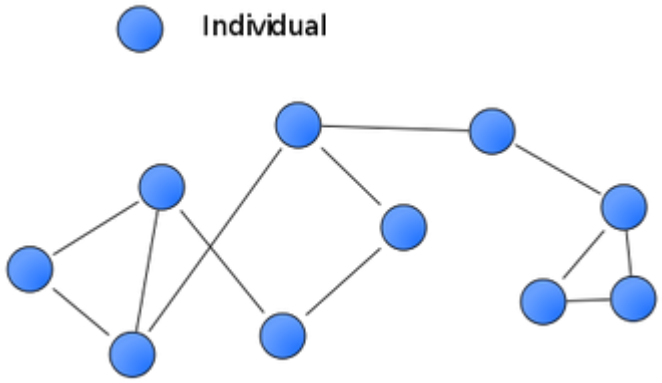
Circuito eléctrico

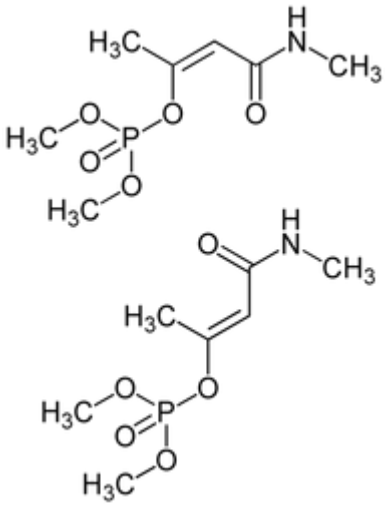
Mapas conceptuales

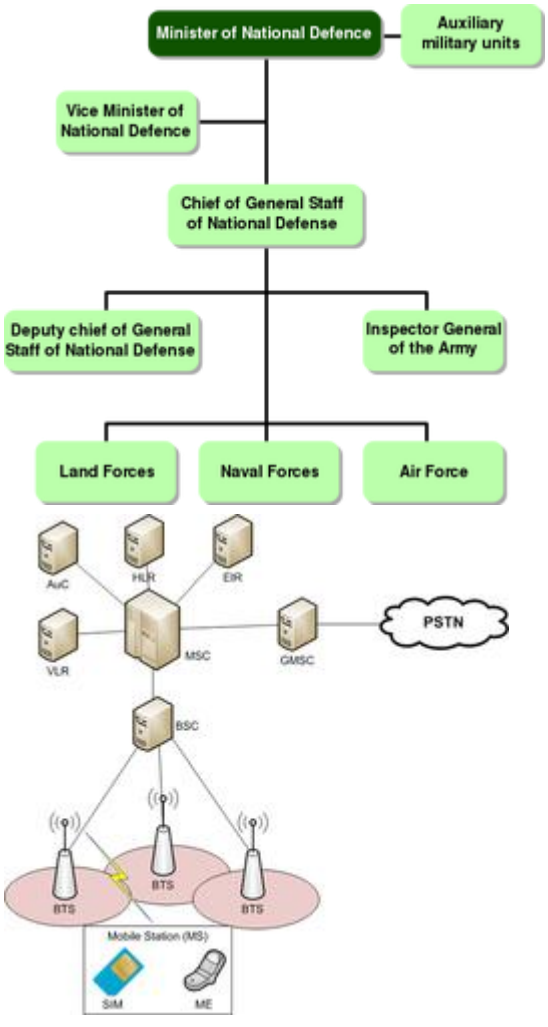
Plano de estaciones delmetro

Topología de red decomputadores

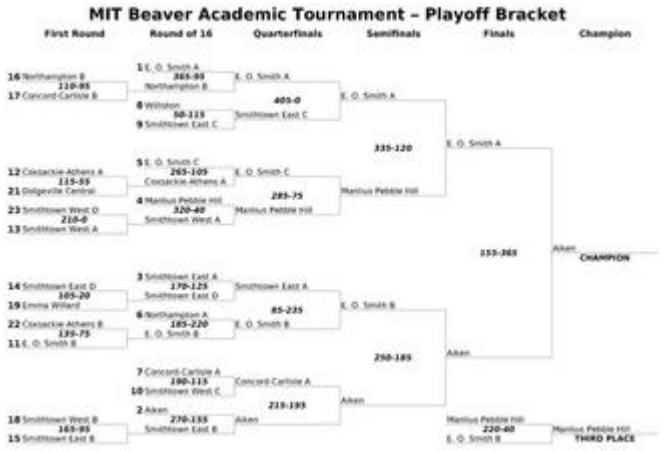
Plano de autopistas

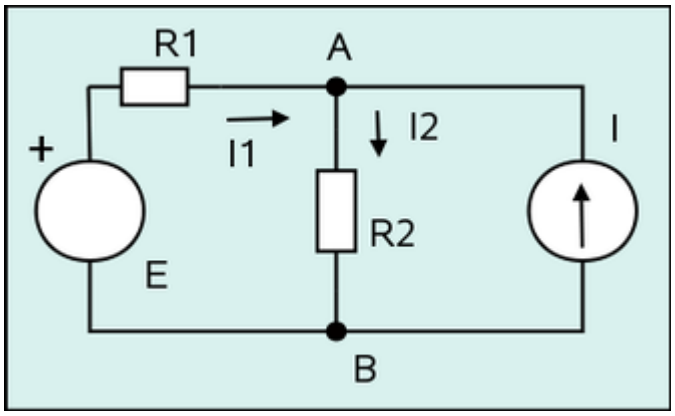


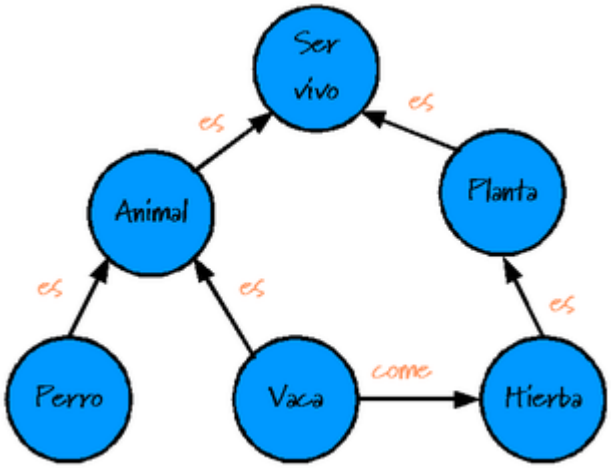



Minister of National Defence
Auxiliary military units
Vice Minister of National Defence
Chief of General Staff of National Defense
Deputy chief of General Staff of National Defense
Inspector General of the Army
Land Forces
Naval Forces
Air Force

AuC
HLR
EIR
VLR
MSC
GMSC
PSTN
BSC
BTS
BTS
BTS
Mobile Station (MS)
SIM
ME


MIT Beaver Academic Tournament – Playoff Bracket
First Round
Round of 16
Quarterfinals
Semifinals
Finals
Champion
CHAMPION
THIRD PLACE


R1
A
I1
I2
I
+
R2
E
B


Ser vivo
es
es
Animal
Planta
es
es
es
come
Perro
Vaca
Hierba


Metro de Madrid