

# Lógica de programación I

Juan Pablo Restrepo Uribe

Ing. Biomedico

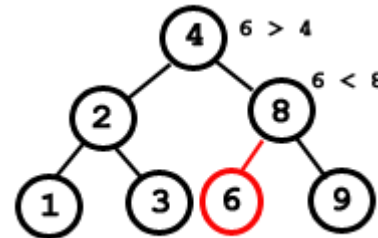
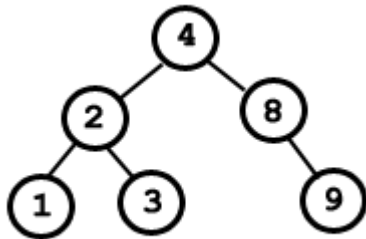
MSc. Automatización y Control Industrial

[jprestrepo@correo.iue.edu.co](mailto:jprestrepo@correo.iue.edu.co)

Institución Universitaria de Envigado

## Insertión

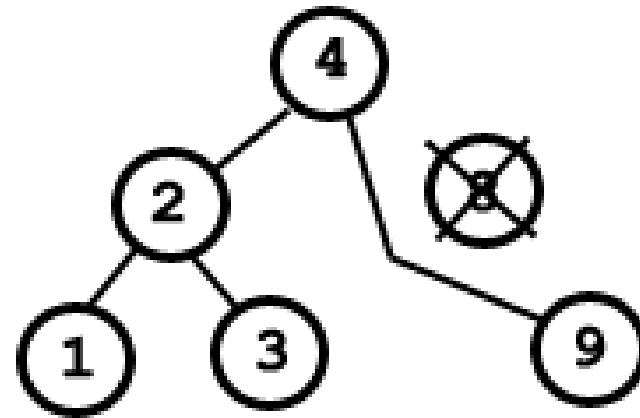
Lo primero es comparar el nuevo elemento con la raíz. Como  $6 > 4$ , entonces la búsqueda prosigue por el lado derecho. Ahora el nuevo nodo se compara con el elemento 8. En este caso  $6 < 8$ , por lo que hay que continuar la búsqueda por la rama izquierda. Como la rama izquierda de 8 no tiene ningún nodo, se cumple la condición de parada de la recursividad y se inserta en ese lugar el nuevo nodo.



## Borrar

Tras realizar la búsqueda del nodo a eliminar observamos que el nodo no tiene hijos. Este es el caso más sencillo, únicamente habrá que borrar el elemento y ya habremos concluido la operación.

Si tras realizar la búsqueda nos encontramos con que tiene un sólo hijo. Este caso también es sencillo, para borrar el nodo deseado, hacemos una especie de puente, el padre del nodo a borrar pasa a apuntar al hijo del nodo borrado.



## Borrar

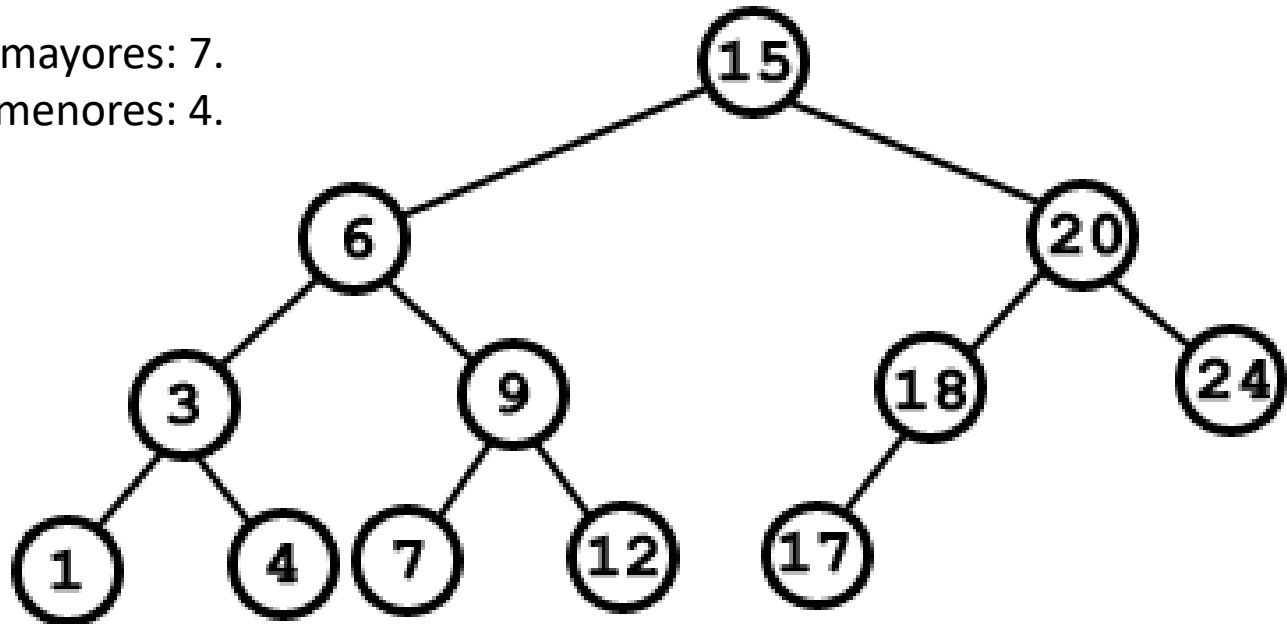
Si el nodo a borrar tiene dos hijos. En este caso se debe sustituir el nodo a borrar por mayor de los nodos menores del nodo borrado, o por el menor de los nodos mayores de dicho nodo. Una vez realizada esta sustitución se borrará el nodo que sustituyó al nodo eliminado (operación sencilla ya que este nodo tendrá un hijo a lo sumo).

## Borrar

Sobre el siguiente árbol queremos eliminar el elemento 6. Tenemos dos opciones para sustituirlo:

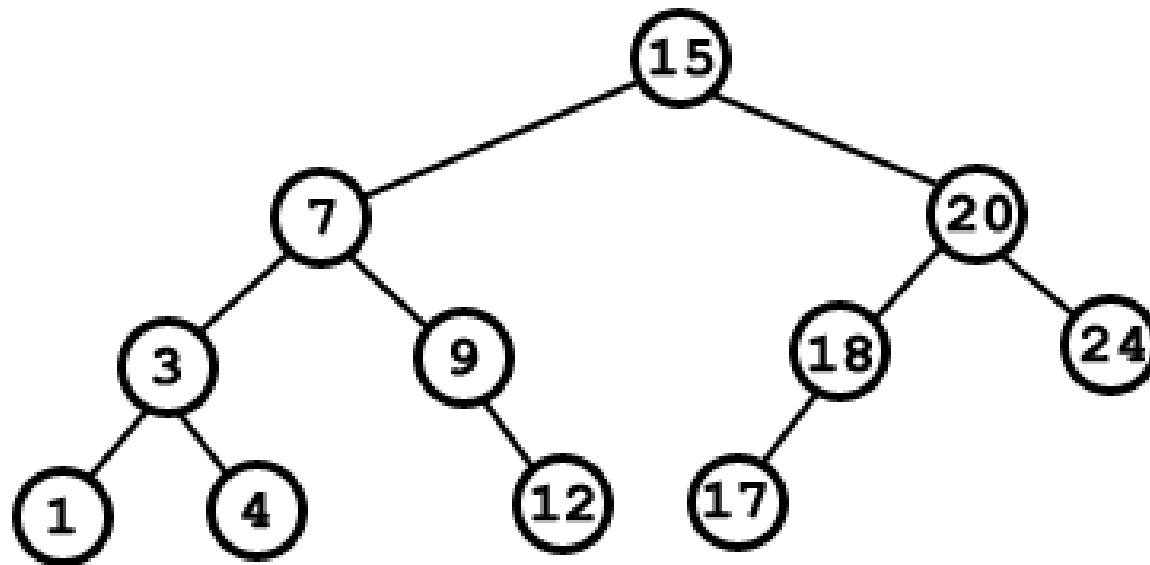
El menor de sus mayores: 7.

El mayor de sus menores: 4.



## Borrar

Vamos a sustituirlo por el 7 (por ejemplo). El árbol resultante sería el siguiente, tras eliminar también el elemento 7 de su ubicación original.



## Otras operaciones

- Vaciar árbol: Esta acción elimina todos los elementos presentes en la lista.
- Camino recorrido: En este lugar se muestran los diferentes nodos por los que fue pasando (con los que se comparó), el elemento insertado o eliminado.

## Aplicaciones de los árboles binarios

- **Eficiencia en la búsqueda:** permiten una búsqueda eficiente de elementos en conjuntos de datos ordenados. Esto se debe a que la estructura del árbol permite descartar rápidamente grandes partes del conjunto de datos durante la búsqueda
- **Almacenamiento de datos ordenados:** Los BST almacenan datos de forma ordenada según un criterio específico (por ejemplo, valor numérico o alfabético).
- **Operaciones de inserción y eliminación eficientes:** admiten operaciones eficientes de inserción y eliminación de elementos, siempre que el árbol se mantenga equilibrado.
- **Evaluación de expresiones aritméticas:** Los árboles de expresiones se utilizan para representar y evaluar expresiones aritméticas y lógicas de manera eficiente. Esto es útil en aplicaciones que requieren procesamiento de fórmulas matemáticas o evaluación de condiciones lógicas.

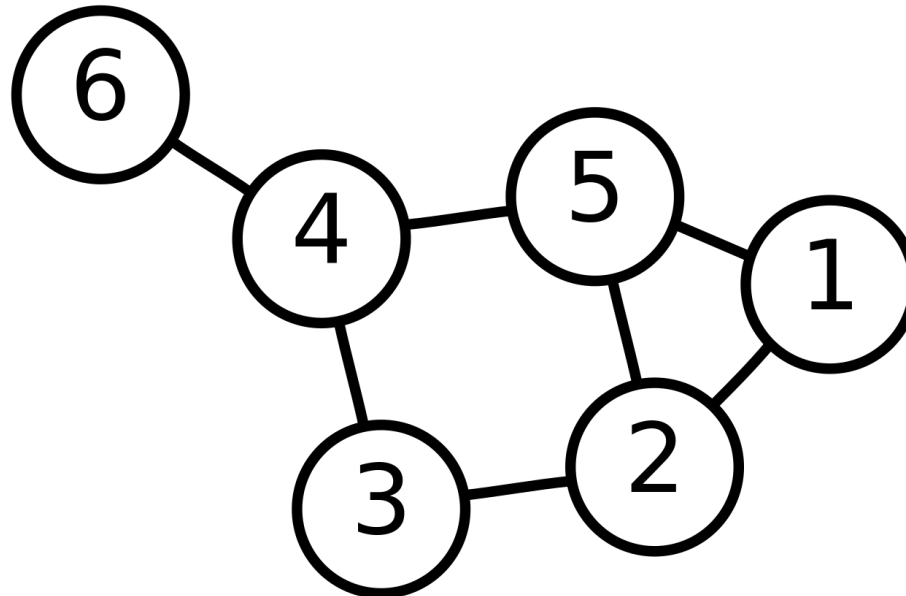


## Aplicaciones de los árboles binarios

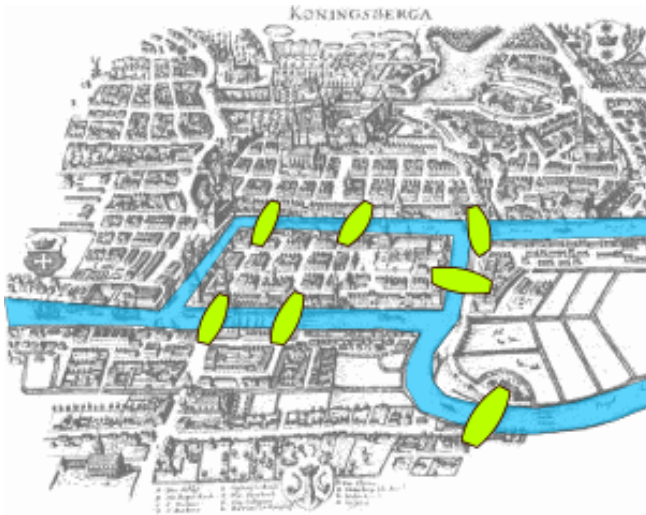
- **Compresión de datos:** Los árboles Huffman se utilizan en algoritmos de compresión de datos para generar códigos de longitud variable que minimizan la longitud total de la codificación.
- **Autocompletado y búsqueda de prefijos:** Los árboles de prefijos (Trie) se utilizan en motores de búsqueda y editores de texto para implementar funciones de autocompletado y búsqueda de palabras por prefijo. Esto facilita la búsqueda y la recuperación rápida de palabras y frases en grandes conjuntos de datos de texto.
- **Estructuras de datos especializadas:** Los árboles de segmentos se utilizan para realizar consultas de rangos en conjuntos de datos, como encontrar la suma, el máximo o el mínimo de un rango específico de elementos.

# GRAFOS

En matemáticas y ciencias de la computación, un grafo es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.<sup>2</sup> Son objeto de estudio de la teoría de grafos.



## GRAFOS

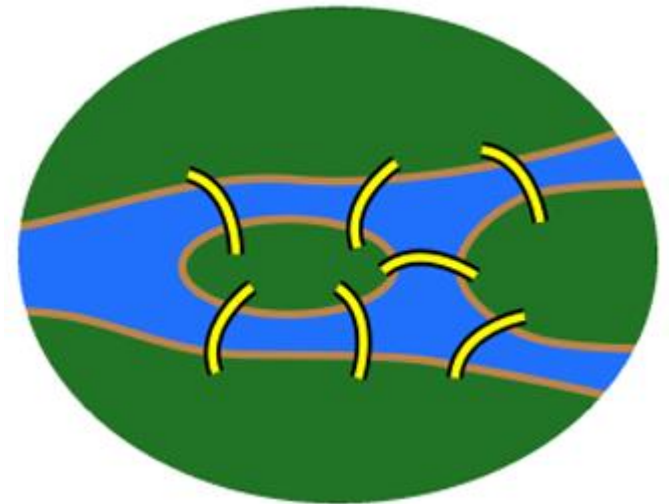


El primer artículo científico relativo a grafos fue escrito por el matemático suizo Leonhard Euler en 1736. Euler se basó en su artículo en el problema de los puentes de Königsberg. La ciudad de Kaliningrado, originalmente Königsberg, es famosa por sus siete puentes que unen ambas márgenes del río Pregel con dos de sus islas

Dos de los puentes unen la isla mayor con la margen oriental y otros dos con la margen occidental. La isla menor está conectada a cada margen por un puente y el séptimo puente une ambas islas. El problema planteaba lo siguiente: "¿Es posible dar un paseo comenzando desde cualquiera de estas regiones, pasando por todos los puentes, recorriendo solo una vez cada uno y regresando al mismo punto de partida?"

# ESTRUCTURAS DE DATOS EN LA REPRESENTACIÓN DE GRAFOS

Existen diferentes formas de almacenar grafos en una computadora. La estructura de datos usada depende de las características del grafo y el algoritmo usado para manipularlo. Entre las estructuras más sencillas y usadas se encuentran las listas y las matrices, aunque frecuentemente se usa una combinación de ambas. Las listas son preferidas en grafos dispersos porque tienen un eficiente uso de la memoria. Por otro lado, las matrices proveen acceso rápido, pero pueden consumir grandes cantidades de memoria.



# GRAFOS

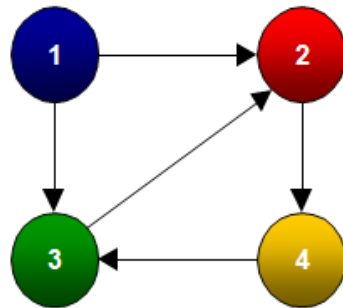
## Lista de incidencia

Las aristas son representadas con un vector de pares (ordenados, si el grafo es dirigido), donde cada par representa una de las aristas.

## Lista de adyacencia.

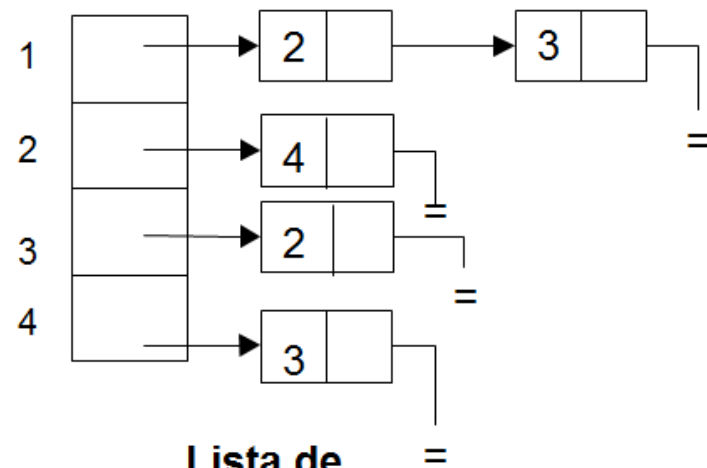
Cada vértice tiene una lista de vértices los cuales son adyacentes a él. Esto causa redundancia en un grafo no dirigido (ya que A existe en la lista de adyacencia de B y viceversa), pero las búsquedas son más rápidas, al costo de almacenamiento extra.

# GRAFOS



**Grafos**

## Vértices



**Lista de  
Adyancencia**

# GRAFOS

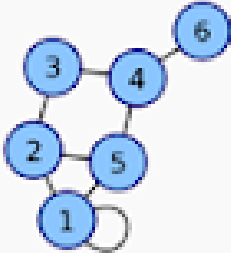
## Matriz de incidencia

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

## Matriz de adyacencia

El grafo está representado por una matriz cuadrada  $M$  de tamaño  $n$ , donde  $n$  es el número de vértices. Si hay una arista entre un vértice  $x$  y un vértice  $y$ , entonces el elemento es 1, de lo contrario, es 0.

# GRAFOS

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

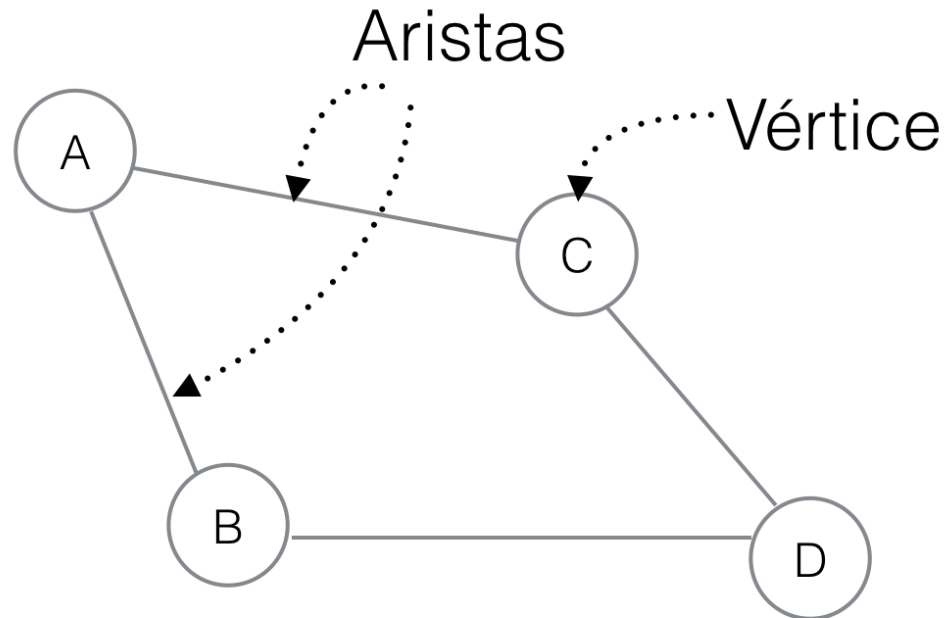


# VÉRTICE

Los vértices constituyen uno de los dos elementos que forman un grafo. Como ocurre con el resto de las ramas de las matemáticas, a la Teoría de Grafos no le interesa saber qué son los vértices.

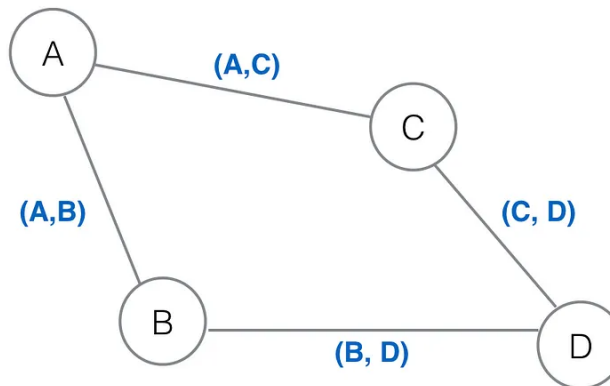
Diferentes situaciones en las que pueden identificarse objetos y relaciones que satisfagan la definición de grafo pueden verse como grafos y así aplicar la Teoría de Grafos en ellos.

# VÉRTICE Y ARISTAS



# GRAFO

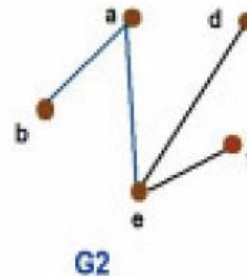
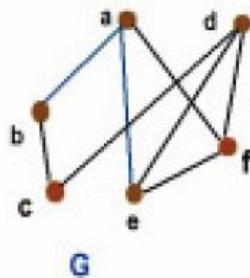
En un grafo dirigido  $G=(V,E)$ , donde  $V$  es el conjunto de nodos y  $E$  es el conjunto de arcos, cada arco  $(u,v)$  en  $E$  representa una conexión unidireccional desde el nodo  $u$  hacia el nodo  $v$ . Esto significa que se puede viajar de  $u$  a  $v$  a lo largo del arco, pero no necesariamente en la dirección opuesta.



# SUBGRAFO

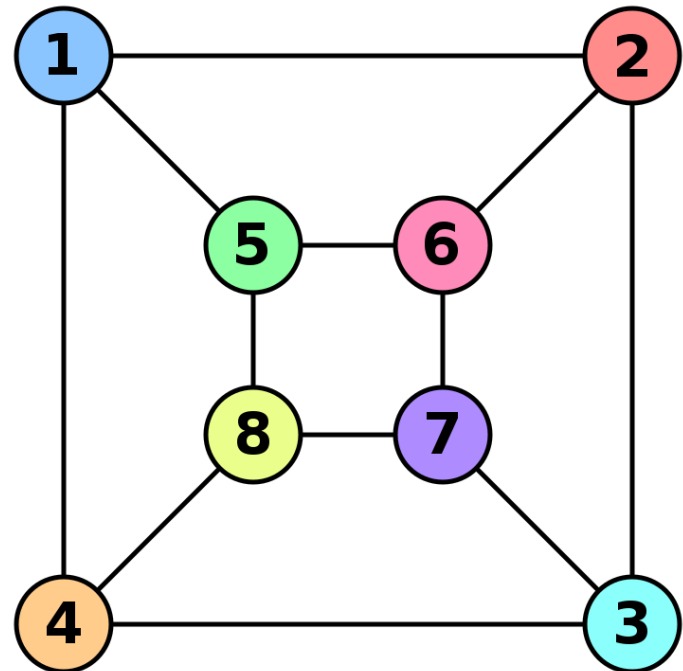
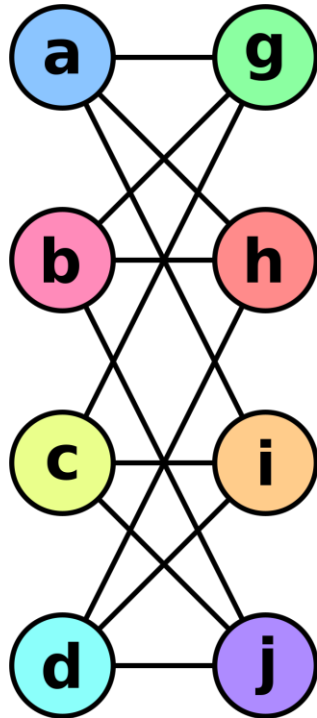
Un subgrafo de un grafo  $G$  es un grafo cuyos conjuntos de vértices y aristas son subconjuntos de los de  $G$ . Se dice que un grafo  $G$  contiene a otro grafo  $H$  si algún subgrafo de  $G$  es  $H$  o es isomorfo a  $H$  (dependiendo de las necesidades de la situación).

El subgrafo inducido de  $G$  es un subgrafo  $G'$  de  $G$  tal que contiene todas las aristas adyacentes al subconjunto de vértices de  $G$ .



# ISOMORFISMO

Es una biyección de los vértices de un grafo sobre otro, de modo que se preserve la adyacencia de los vértices.

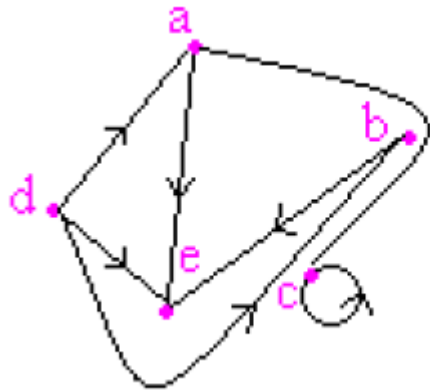


## ARISTAS DIRIGIDAS Y NO DIRIGIDAS

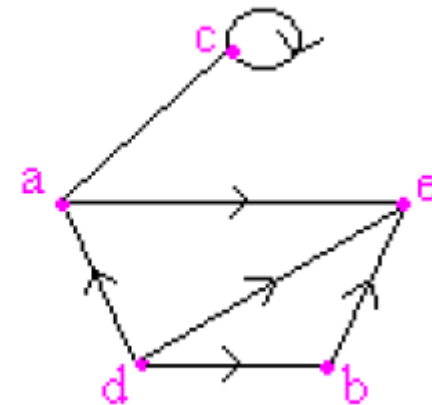
En algunos casos es necesario asignar un sentido a las aristas, por ejemplo, si se quiere representar la red de las calles de una ciudad con sus direcciones únicas. El conjunto de aristas será ahora un subconjunto de todos los posibles pares ordenados de vértices, con  $(a, b) \neq (b, a)$ . Los grafos que contienen aristas dirigidas se denominan grafos orientados, como el siguiente:

- Las aristas no orientadas se consideran bidireccionales para efectos prácticos (equivale a decir que existen dos aristas orientadas entre los nodos, cada una en un sentido).

# ARISTAS DIRIGIDAS Y NO DIRIGIDAS

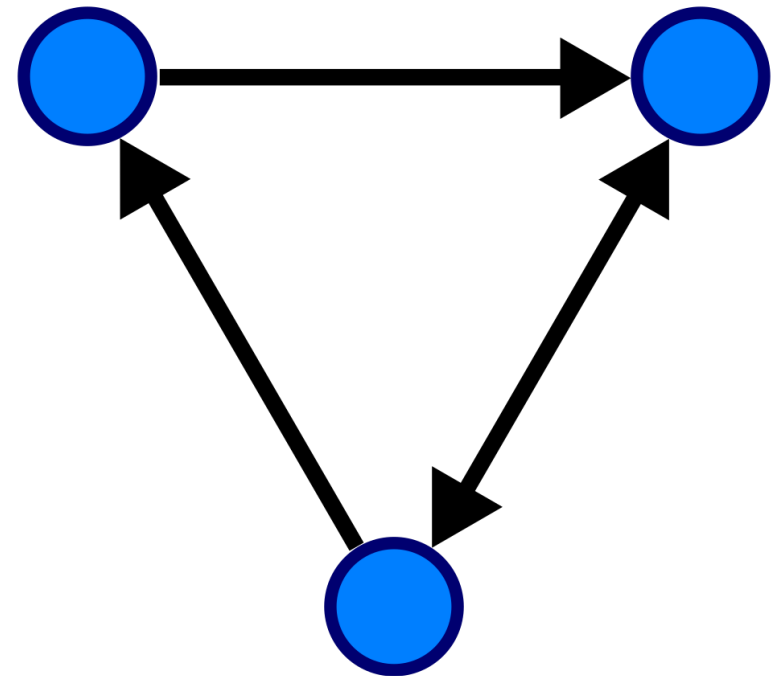


es isomorfo a



## GRAFO DIRIGIDO

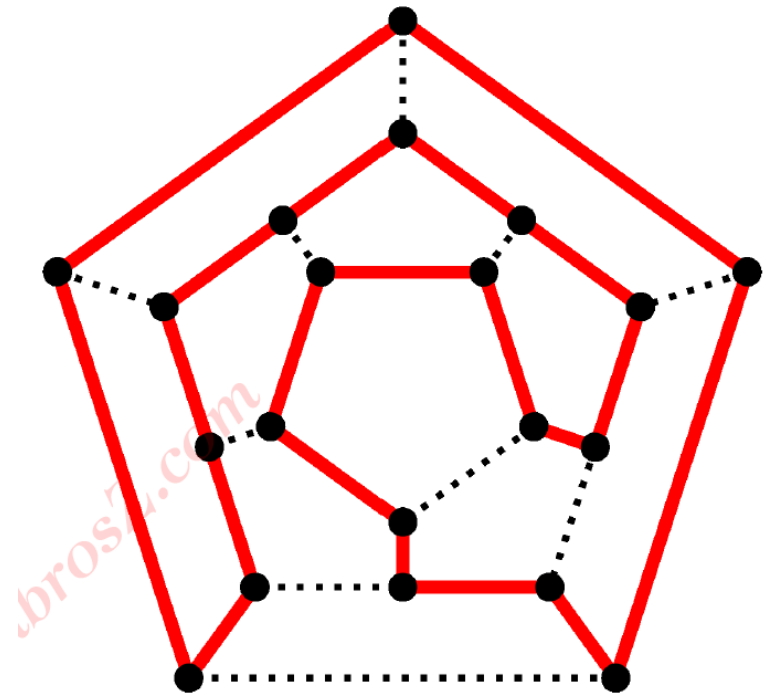
Es un tipo de estructura de datos que consiste en un conjunto de nodos (también llamados vértices) conectados entre sí por arcos o aristas direccionales. En un grafo dirigido, las aristas tienen una dirección asociada que indica la relación unidireccional entre los nodos conectados.

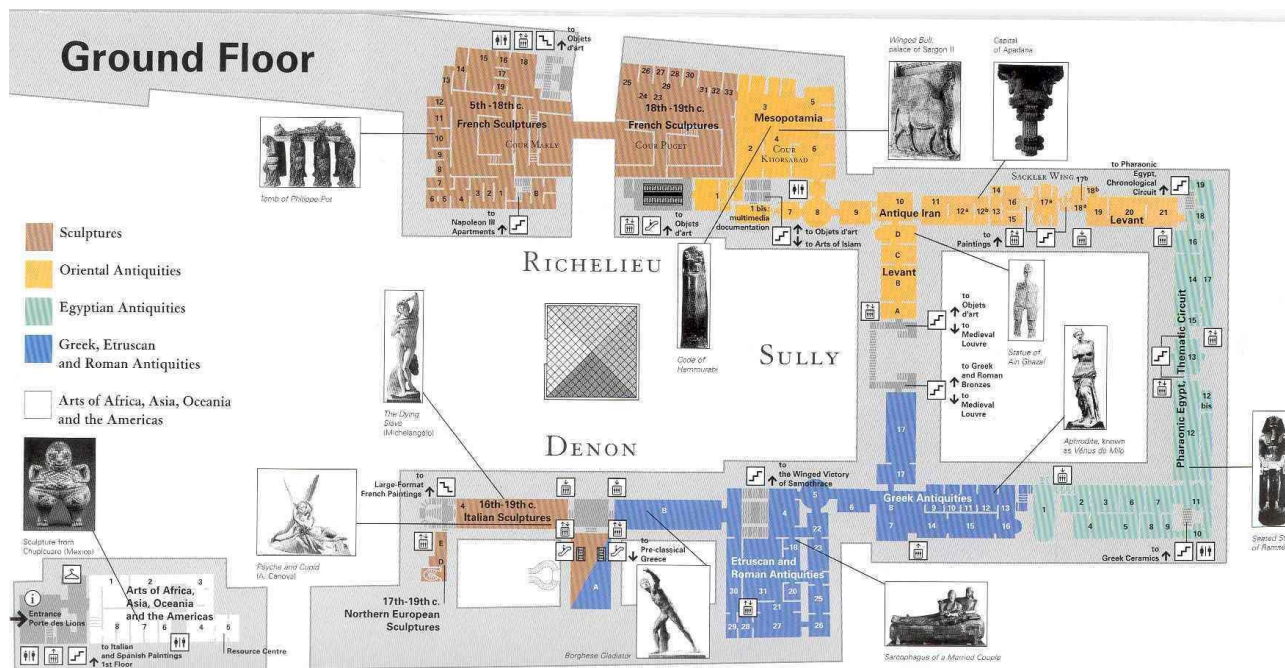




# CICLOS Y CAMINOS HAMILTONIANOS

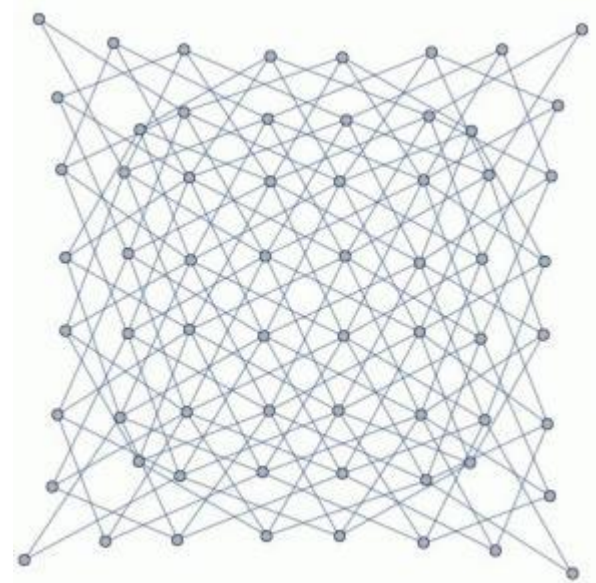
Un ciclo es una sucesión de aristas adyacentes, donde no se recorre dos veces la misma arista, y donde se regresa al punto inicial. Un ciclo hamiltoniano tiene además que recorrer todos los vértices exactamente una vez (excepto el vértice del que parte y al cual llega).





# CICLOS Y CAMINOS HAMILTONIANOS

Hoy en día, no se conocen métodos generales para hallar un ciclo hamiltoniano en tiempo polinómico, siendo la búsqueda por fuerza bruta de todos los posibles caminos u otros métodos excesivamente costosos. Existen, sin embargo, métodos para descartar la existencia de ciclos o caminos hamiltonianos en grafos pequeños.



# CARACTERIZACIÓN DE GRAFOS

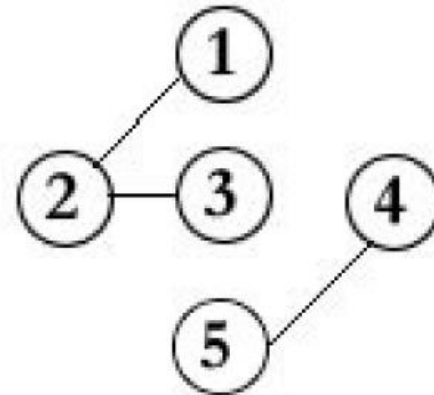
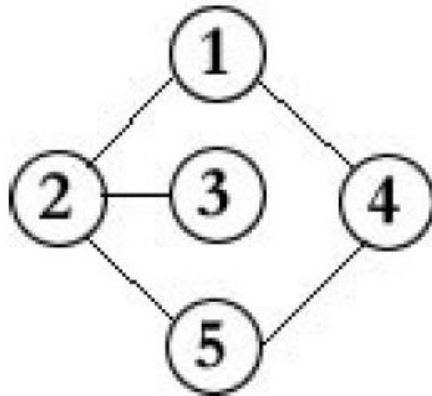
## Grafos simples

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.

## Grafos conexos

Un grafo es conexo si cada par de vértices está conectado por un camino; es decir, si para cualquier par de vértices  $(a, b)$ , existe al menos un camino posible desde  $a$  hacia  $b$ . Un grafo es doblemente conexo si cada par de vértices está conectado por al menos dos caminos disjuntos; es decir, es conexo y no existe un vértice tal que al sacarlo el grafo resultante sea desconexo.

# CARACTERIZACIÓN DE GRAFOS



# HOMEOMORFISMO DE GRAFOS

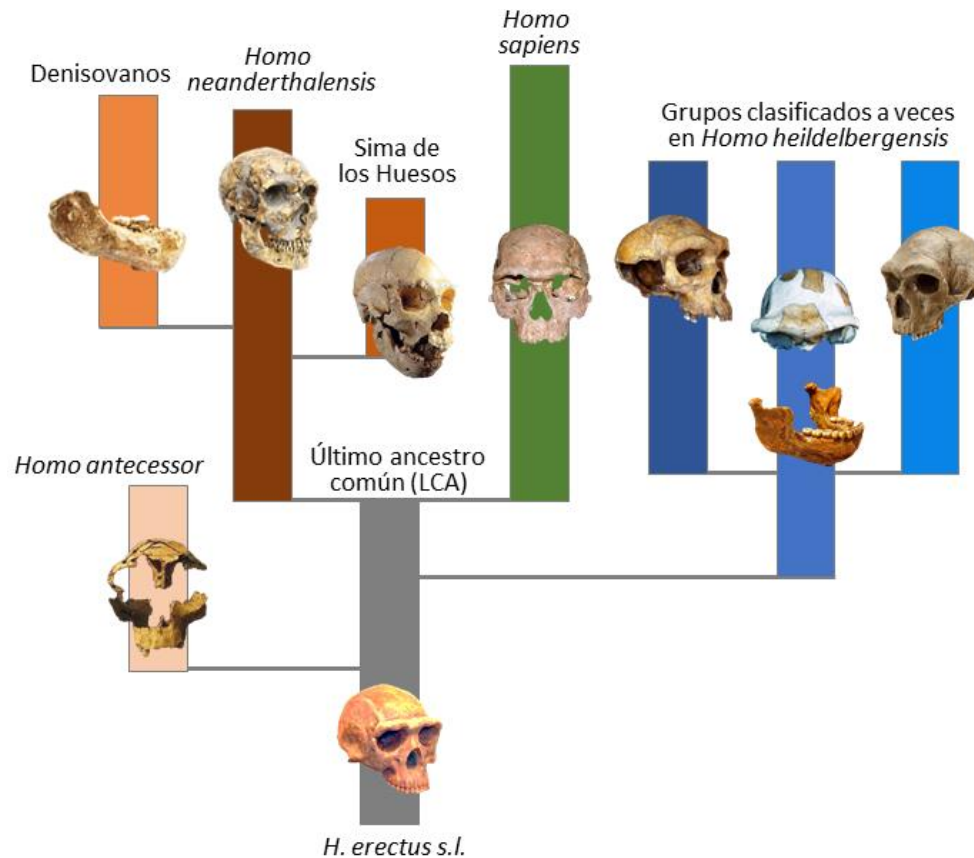
Dos grafos  $G$  y  $H$  son homeomorfos si ambos pueden obtenerse a partir del mismo grafo con una sucesión de subdivisiones elementales de aristas.

# ARBOLES

Un grafo que no tiene ciclos y que conecta a todos los puntos, se llama un árbol. En un grafo con  $n$  vértices, los árboles tienen exactamente  $n - 1$  aristas, y hay  $n^{n-2}$  árboles posibles. Su importancia radica en que los árboles son grafos que conectan todos los vértices utilizando el menor número posible de aristas.

Un importante campo de aplicación de su estudio se encuentra en el análisis filogenético, el de la filiación de entidades que derivan unas de otras en un proceso evolutivo, que se aplica sobre todo a la averiguación del parentesco entre especies; aunque se ha usado también, por ejemplo, en el estudio del parentesco entre lenguas.

# ÁRBOL FILOGENÉTICO





# ALGORITMOS IMPORTANTES

## Algoritmos importantes

- Algoritmo de búsqueda en anchura (BFS)
- Algoritmo de búsqueda en profundidad (DFS)
- Algoritmo de búsqueda A\*
- Algoritmo del vecino más cercano
- Ordenación topológica de un grafo
- Algoritmo de cálculo de los componentes

fuertemente conexos de un grafo

- Algoritmo de Dijkstra
- Algoritmo de Bellman-Ford
- Algoritmo de Prim
- Algoritmo de Ford-Fulkerson
- Algoritmo de Kruskal
- Algoritmo de Floyd-Warshall

# Grafo dirigido