Grafos I

Los **grafos** son estructuras de datos que representan relaciones arbitrarias entre objetos, a diferencia de los árboles que suelen reflejar jerarquías. Un grafo está compuesto por:

- **Vértices o nodos (V)**: Son los puntos que representan los objetos.
- **Arcos o aristas (A)**: Conjunto de conexiones entre pares de nodos.

Tipos de grafos

- **Grafo dirigido (dígrafo)**: Los arcos tienen una dirección, conectando un nodo de origen con otro de destino, pero no necesariamente en sentido inverso.
- **Grafo no dirigido**: Los arcos son bidireccionales, conectando dos nodos de manera simétrica.
- **Grafo valorado**: Los arcos tienen un peso asociado, que podría representar una distancia, coste, o cualquier otro valor.

Propiedades importantes

- **Nodos adyacentes**: En un grafo no dirigido, dos nodos son adyacentes si están conectados por un arco. En un grafo dirigido, solo el primer nodo es adyacente del segundo.
- **Grado de un nodo**: Número de arcos que inciden en un nodo. En dígrafos, se distingue entre grado entrante (arcos que llegan al nodo) y grado saliente (arcos que parten del nodo).
- **Camino**: Secuencia de nodos conectados por arcos. Un camino tiene una longitud que se define como el número de arcos que lo componen.
- **Bucle**: Arco que conecta un nodo consigo mismo.
- **Grafo conexo**: Grafo donde existe un camino entre cualquier par de nodos.
- **Fuertemente conexo**: En dígrafos, si existe un camino en ambas direcciones entre cualquier par de nodos.
- **Grafo completo**: Grafo en el que existe un arco entre cada par de nodos.

Aplicaciones de los grafos

Los grafos tienen muchas aplicaciones prácticas, como:

- Redes de alcantarillado.
- Redes de comunicaciones.
- Circuitos eléctricos.
- Diagramas de flujo de algoritmos.

Representación de Grafos

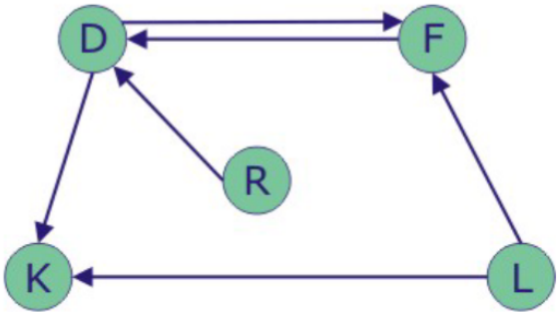
Los grafos se pueden representar de diferentes maneras, dependiendo de la operación a realizar:

Matriz de adyacencia

- Es una matriz cuadrada de tamaño $n \times n$, donde n es el número de nodos.
- Cada posición i, j en la matriz indica si existe un arco entre los nodos i y j (1 si existe, 0 si no).
- En grafos no dirigidos, la matriz es simétrica. En los dígrafos, no necesariamente.
- En grafos valorados, los elementos de la matriz representan el peso del arco.

Ventajas: - Muy eficiente para obtener el coste o peso asociado a un arco. - Comprobación de adyacencia entre dos nodos es directa.

Desventajas: - No permite eliminar nodos fácilmente, ya que implicaría modificar toda la matriz. - Poco eficiente en grafos dispersos, ya que muchas posiciones estarán llenas



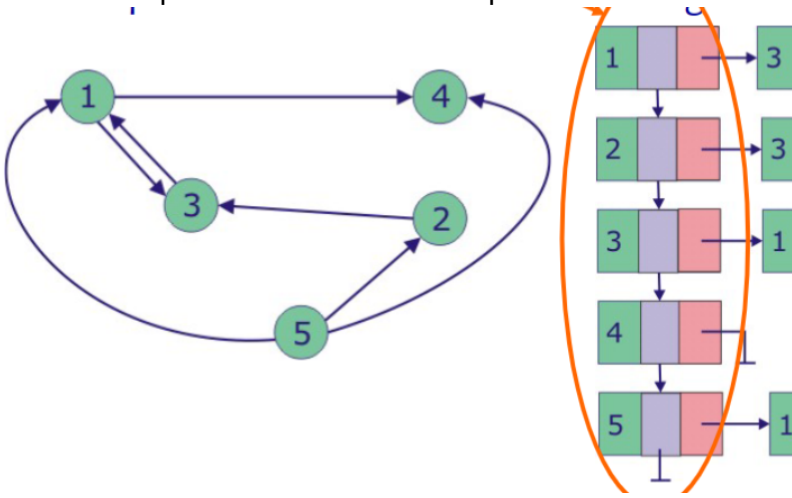
$$A = \begin{array}{c} \begin{matrix} D & F & K & L & R \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \begin{matrix} D \\ F \\ K \\ L \\ R \end{matrix}$$

R→D

de ceros.
Lista de adyacencia - Cada nodo tiene asociada una lista enlazada que contiene los nodos con los que es adyacente.

Ventajas: - Ocupa menos espacio en memoria para grafos con pocos arcos.

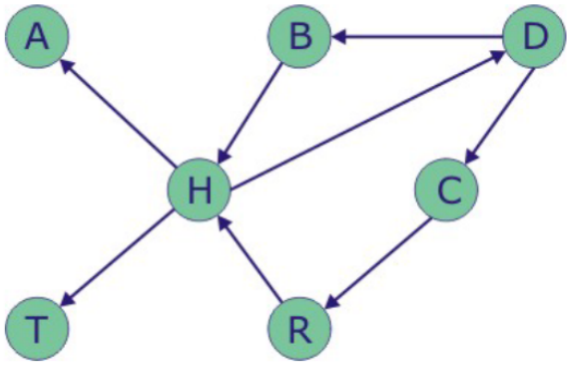
Desventajas: - Más compleja de manejar por el uso de punteros. - Ineficiente para encontrar todos los arcos que llegan a un nodo.



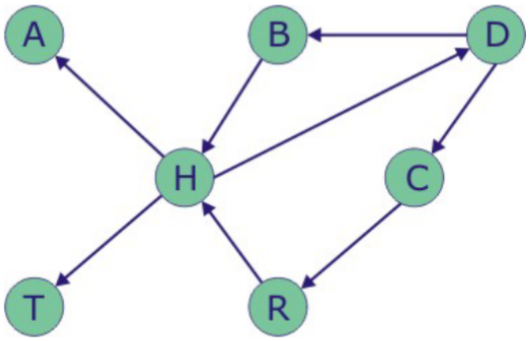
Recorridos en Grafos

Recorrido en anchura (BFS - Breadth-First Search)

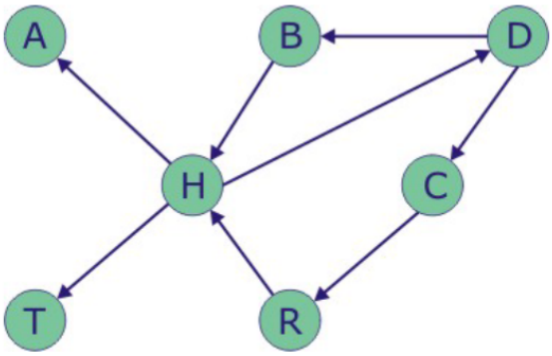
- Se comienza desde un nodo y se visitan todos sus nodos adyacentes. Luego se continúa con los adyacentes de esos nodos, y así sucesivamente.

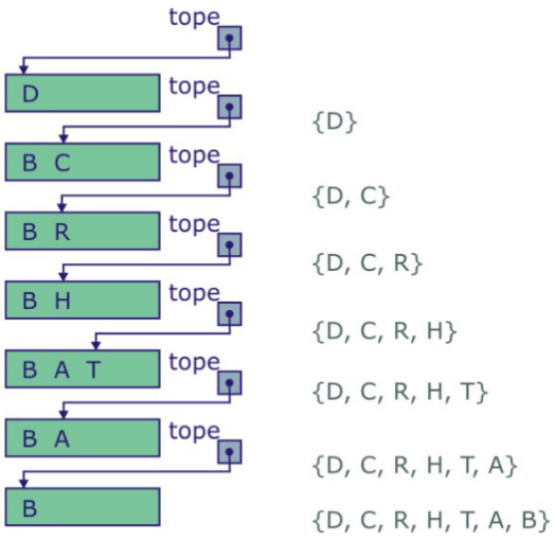


- Utiliza una **cola** para almacenar los nodos a visitar.
  ### Recorrido en profundidad (DFS - Depth-First Search)
- A partir de un nodo, se recorre cada vértice adyacente hasta no encontrar más nodos no visitados.
- Puede implementarse de manera **recursiva** o utilizando una **pila** como estructura auxiliar.



D →RR(D) {D} Adyacentes: {B,C}
→RR(B) {D,B} Adyacentes: {H}
→RR(H) {D,B,H} Adyacentes: {A,T}
→RR(A) {D,B,H,A} Adyacentes: {}
→RR(T) {D,B,H,A,T} Adyacentes:{}
→RR(C) {D,B,H,A,T,C} Adyacentes:{R}
→RR(R) {D,B,H,A,T,C,R} Adyacentes: {}
No hay nodos sin visitar, por lo que termina el recorrido:
{D,B,H,A,T,C,R}

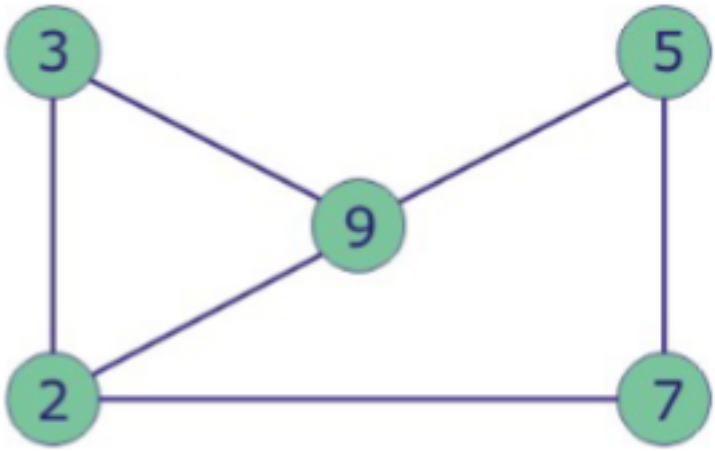




Componentes conexas

Componentes conexas en un grafo no dirigido

1. Se realiza un recorrido desde cualquier vértice, almacenando los nodos visitados.
2. Si todos los vértices fueron visitados, el grafo es conexo.
3. Si no, los vértices visitados forman una componente conexa. Se repite el proceso con un vértice no visitado.



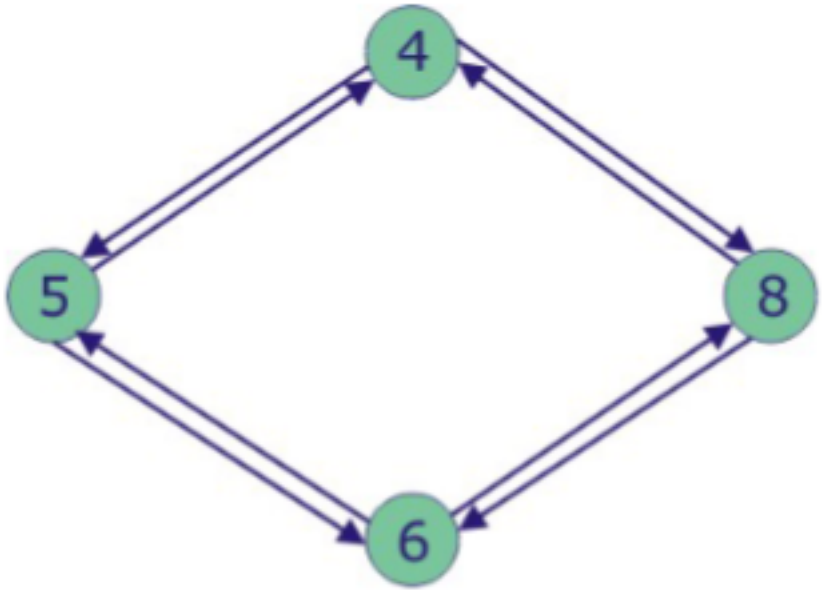
Componentes fuertemente conexas en un dígrafo

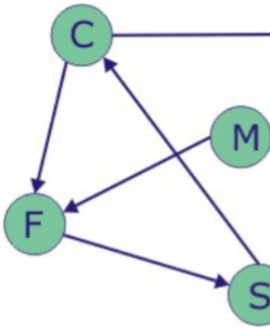
4. Se obtiene el conjunto de descendientes de un nodo. $D(n)$
5. Se obtiene el conjunto de sus ascendientes. $A(n)$
6. Si la intersección de ambos conjuntos incluye todos los nodos, el dígrafo es fuertemente conexo.

$$G = D(n) \cap A(n) \Longrightarrow \text{G es fuertemente conexo}$$

$G \neq D(n) \cap A(n) \Longrightarrow$ G no es fuertemente conexo

as

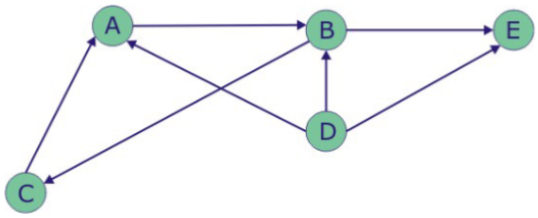




Grafo dirigido fuertemente conexo

Matriz de caminos - Se obtiene multiplicando la matriz de adyacencia por sí misma repetidamente. La matriz resultante indica la existencia de caminos de diferente longitud entre nodos.

- Ejemplo:



Matriz de adyacencia A

$$A = \begin{pmatrix} F & T & F & F & F \\ F & F & T & F & T \\ T & F & F & F & F \\ T & T & F & F & T \\ F & F & F & F & F \end{pmatrix}$$

Producto Booleano AxA

$$A \times A = A^2 = \begin{pmatrix} F & F & T & F & T \\ T & F & F & F & F \\ F & T & F & F & F \\ F & T & T & F & T \\ F & F & F & F & F \end{pmatrix}$$

A_{15}=TRUE pues hay camino de longitud 2 de A a E (A→B→E)

Producto Booleano A^2xA

$$A^2 \times A = A^3 = \begin{pmatrix} T & F & F & F \\ F & T & F & F \\ F & F & T & F \\ T & F & T & F \\ F & F & F & F \end{pmatrix}$$

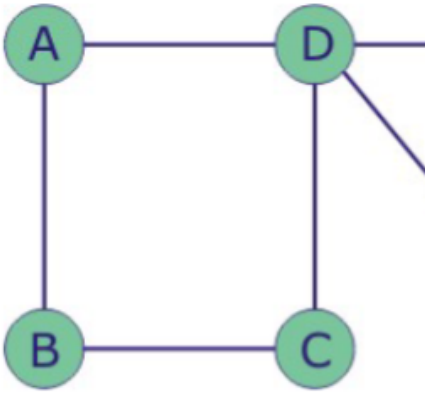
A_{41}=TRUE pues hay camino d longitud 3 de D a A (D→B→C→A)

- En un grafo fuertemente conexo, todas las entradas de la matriz de caminos tienen un 1 excepto la diagonal principal.

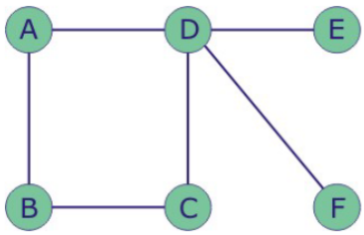
Puntos de articulación

Un **punto de articulación** es un nodo que, si se elimina junto con sus arcos, divide una componente conexa en dos o más componentes.

Algoritmo para encontrar puntos de articulación

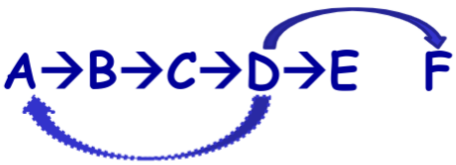


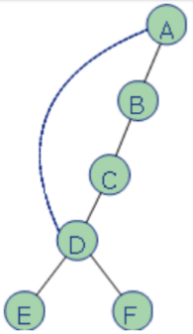
1. Se realiza un recorrido en profundidad, numerando los nodos.



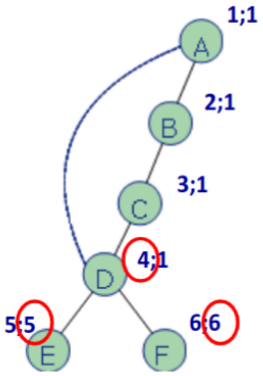
A →RR(A) {A} Adyacentes no visitados: {B,D}
→RR(B) {A,B} Adyacentes no visitados: {C}
→RR(C) {A,B,C} Adyacentes NV: {D}
→RR(D) {A,B,C,D} Adyacentes NV: {E,F}
→RR(E) {A,B,C,D,E} Adyacentes NV:{}
→RR(F) {A,B,C,D,E,F} Adyacentes NV:{}
→RR(D): YA VISITADO!! Arco hacia atrás
No hay nodos sin visitar, por lo que termina el recorrido: {A,B,C,D,E,F}

Numerando los nodos del árbol en preorden (RID) obtenemos el orden en que han sido visitados





2. Para cada nodo v, se calcula $Bajo(v)$, que es el mínimo de:
   - Su número $Num(v)$.
   - El menor número de los vértices alcanzados por aristas hacia atrás.



- Num(v)=recorrido pre
- Bajo(v)=min{Num(v),
Como necesito Bajo(hijo
- Bajo(E)=min{Num(E)
- Bajo(F)=min{Num(F)
- Bajo(D)=min{Num(D)
- Bajo(C)=min{Num(C)
- Bajo(B)=min{Num(B)
- Bajo(A)=min{Num(A)

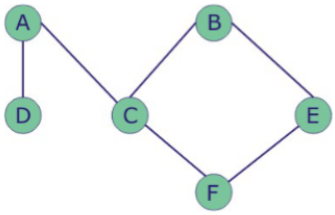
Sólo D cumple la condición, pu

Se cumple para los dos hijos

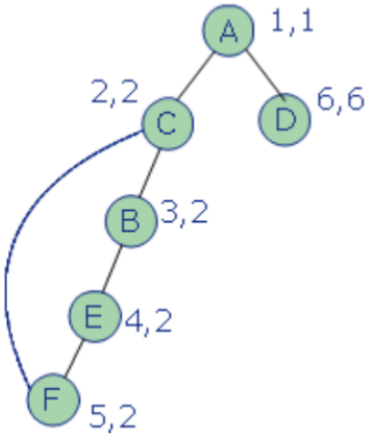
- El menor valor de $Bajo$ entre sus descendientes.

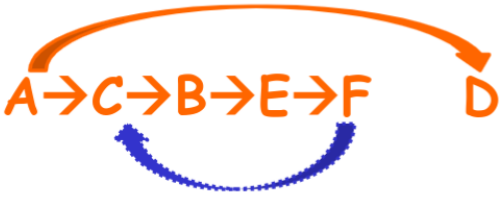
3. Un nodo es punto de articulación si:
   - Es la raíz del recorrido y tiene al menos dos hijos.
   - Tiene un hijo u tal que $Num(v) \leq Bajo(u)$.

Ejemplo:

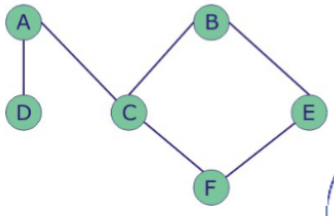


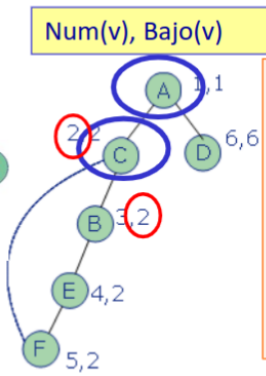
A →RR(A) {A} Adyacentes NV: {C,D}
→RR(C) {A,C} Adyacentes NV: {B,F}
→RR(B) {A,C,B} Adyacentes NV: {E}
→RR(E) {A,C,B,E} Adyacentes NV: {F}
→RR(F) {A,C,B,E,F} Adyacentes NV:{}
→RR(F): YA VISITADO. ARCO HACIA ATRÁS a quien lo llamó: C
→RR(D) {A,C,B,E,F,D} Adyacentes NV:{}
No hay nodos sin visitar, por lo que termina el recorrido:
{A,C,B,E,F,D}





- **Ejemplo:**





Bajo(F)=min {Num(F), Num(C),Bajo(hijos)}= min{5,2}=2
Bajo(E)=min{Num(E),Num(arista_atrás),Bajo(F)]=min{4,2}=2
Bajo(B)=min{Num(B),Num(arista_atrás),Bajo(E)] =min{3,2}=2
Bajo(C)=min[Num(C),Num(arista_atrás),Bajo(B)]=min{2,2}=2

Bajo(D)=min{Num(D), Num(arista_atrás), Bajo(hijos)}=6

Bajo(A)=min{Num(A),Num(arista_atrás),Bajo(C),Bajo(D)]=
min{1,2,6}=1

- Num(v)=posición en recorrido en profundidad: {A,C,B,E,F,D}
- Bajo(v)=mín de:
  - Num(v)
  - min(Num(w)), siendo (v,w) arista hacia atrás
  - min(Bajo(w)) siendo (v,w) arista
- Puntos articulación:
  - **A**: raíz con nº hijos ≥ 2
  - **C**: Tiene hijo B con Num(C)<=Bajo(B)