Conceptos de Teoría de Grafos (2010/2011)

Luis Valencia Cabrera
lvalencia@us.es
(http://www.cs.us.es/~lvalencia)

Ciencias de la Computacion e IA
(http://www.cs.us.es/)

Universidad de Sevilla

Introducción

- Se presentan conceptos necesarios sobre teoría de grafos
- Los grafos son ***herramientas muy útiles para definir sistemas expertos***
- Sus conceptos pueden ser utilizados para analizar diversos aspectos de estos campos

Contenidos

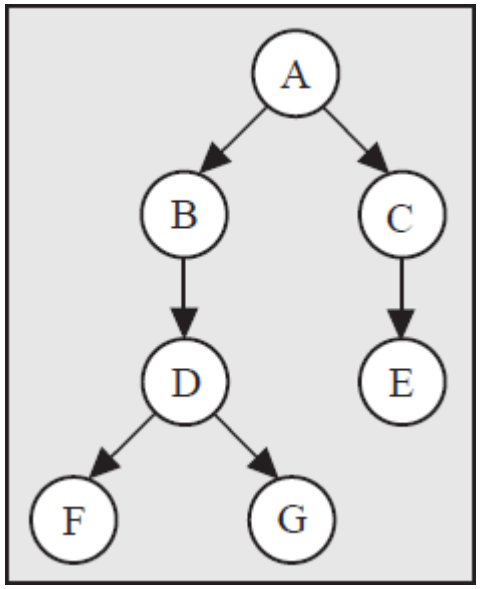
- Conceptos básicos y definiciones
- Grafos no dirigidos y grafos dirigidos
- Grafos triangulados
- Grafos de aglomerados
  - Grafos de conglomerados
  - Grafos de unión
  - Grafos de familias
- Formas de representación de un grafo (simbólica, gráfica y numérica)
- Algoritmos para el análisis de la estructura topológica de un grafo.

Conceptos Básicos y Definiciones

- Sea un conjunto de objetos $X = \{X1, X2, \ldots, Xn\}$ que pueden relacionarse entre sí.
- El conjunto *X* puede representarse por una colección de *nodos* o ***vértices***, asociando un nodo a cada elemento de *X*.
- Los nodos pueden conectarse por ***aristas***, indicando las relaciones existentes entre los mismos.
  - Arista entre *Xi* y *Xj*: *Lij*.
  - Conjunto de aristas: $L = \{Lij \mid Xi \text{ y } Xj \text{ conectados}\}$.
- Conclusión: un grafo G puede definirse de forma intuitiva mediante el conjunto de nodos, *X*, y las relaciones entre los mismos, *L*.
  - G = (X,L)

Grafos. Ejemplo

- Se muestra un grafo compuesto de 6 nodos X = {A,B, . . . , G} y 6 aristas, L = {L_{AB}, L_{AC}, L_{BD}, L_{CE}, L_{DF}, L_{DG}}.
- Los nodos se representan por círculos.
- Las aristas se representan por líneas que unen los nodos.



Grafo o Red

- *Un grafo es un par de conjuntos G = (X,L), donde:*
  - *X = {X1,X2, . . . , Xn} es un conjunto finito de elementos (nodos), y*
  - *L es un conjunto de aristas, es decir, un subconjunto de pares ordenados de elementos distintos de X.*
- *Los términos grafo y red se emplearán como sinónimos.*

Grafos y Sistemas Expertos

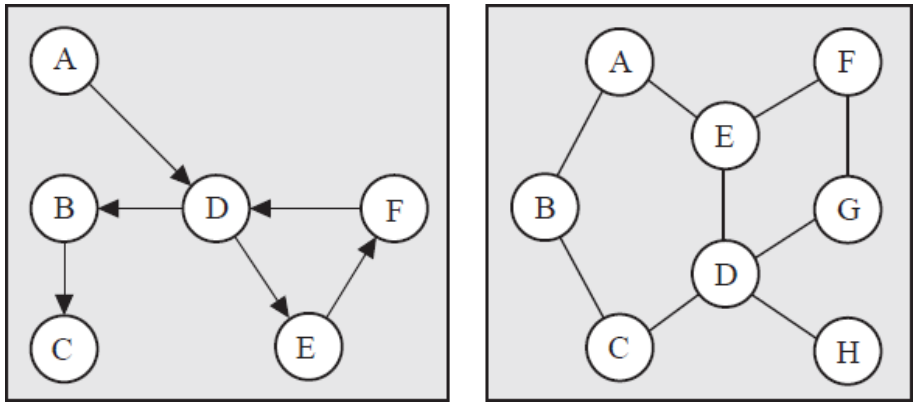
- El concepto de grafo puede definirse de forma más general.
  - Puede permitirse que dos nodos estén conectados por más de una arista
  - Puede que un nodo esté conectado consigo mismo.
- Sin embargo, en el campo de los sistemas expertos, los grafos se utilizan para ***representar un conjunto de variables proposicionales (nodos)***, y ***unas relaciones de dependencia entre ellas (aristas)***.
  - Por tanto, no es necesario que dos nodos estén unidos por más de una arista, o que una arista una un nodo consigo mismo.
- Las aristas pueden ser *dirigidas* o *no dirigidas*, dependiendo de si se considera o no, el orden de los nodos. Depende de si importa el orden en que se relacionan los objetos.

Aristas dirigidas y no dirigidas

- **Arista dirigida.**
  - *Dado un grafo G = (X,L), si Lij ∈ L y Lji /∈ L, la arista Lij entre los nodos Xi y Xj se denomina dirigida y se denota mediante Xi → Xj .*
- **Arista no dirigida.**
  - *Dado un grafo G = (X,L), si Lij ∈ L y Lji ∈ L, la arista Lij se denomina no dirigida y se denota mediante Xi − Xj o Xj − Xi.*

Grafos dirigidos y no dirigidos

- **Grafo dirigido y no dirigido.**
  - *Un grafo en el cual todas las aristas son dirigidas se denomina **grafo dirigido**, y un grafo en el que todas sus aristas son no dirigidas se denomina **no dirigido**.*
- Por tanto, en un grafo dirigido es importante el orden del par de nodos que definen cada arista, mientras que en un grafo no dirigido, el orden carece de importancia.



Grafos dirigidos y no dirigidos. Ejemplo

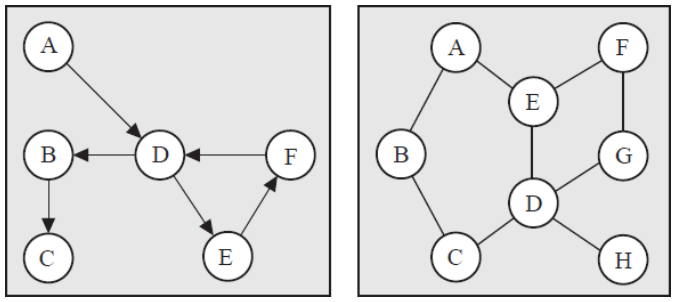
- En las Figuras anteriores se muestran ejemplos de un grafo dirigido y de un grafo no dirigido, respectivamente.
- El grafo de la 1ª figura está definido por:
  - $X = \{A,B,C,D,E,F\}$,
  - $L = \{A \rightarrow D, B \rightarrow C, D \rightarrow B, F \rightarrow D, D \rightarrow E, E \rightarrow F\}$,
- mientras que para el grafo de la 2ª se tiene:
  - $X = \{A,B,C,D,E,F,G,H\}$,
  - $L = \{A - B, B - C, C - D, D - E, E - A, E - F, F - G, G - D, D - H\}$.

Conjunto adyacente

- **Conjunto adyacente.**
  - *Dado un grafo G = (X,L) y un nodo Xi, el conjunto adyacente del nodo Xi es el conjunto de nodos que son directamente alcanzables desde Xi, es decir, Ady(Xi) = {Xj ∈ X | Lij ∈ L}.*
- Se trata de una descripción alternativa de un grafo mediante:
  - un conjunto de nodos, *X*, y
  - los conjuntos adyacentes de cada uno de los nodos en *X*;
- El grafo (*X,L*) puede ser representado de forma equivalente mediante (*X, Ady*), donde:
  - *X = {X1, . . . , Xn}* es el conjunto de nodos y
  - *Ady = {Ady(X1), . . . , Ady(Xn)}* es la lista de conjuntos adyacentes.
- Como veremos en más detalle, esta representación es muy conveniente desde un punto de vista computacional.

Conjuntos adyacentes. Ejemplo

- El grafo dirigido dado en la 1ª figura tiene asociados los siguientes conjuntos de nodos adyacentes:
  - $Ady(A) = \{D\}$, $Ady(B) = \{C\}$, $Ady(C) = \Phi$,
  - $Ady(D) = \{B,E\}$, $Ady(E) = \{F\}$, $Ady(F) = \{D\}$.
- Los conjuntos adyacentes del grafo no dirigido de la 2ª figura son:
  - $Ady(A) = \{B,E\}$, $Ady(B) = \{A,C\}$,
  - $Ady(C) = \{B,D\}$, $Ady(D) = \{C,E, G,H\}$,
  - $Ady(E) = \{A,D, F\}$, $Ady(F) = \{E,G\}$,
  - $Ady(G) = \{D,F\}$, $Ady(H) = \{D\}$.
- Por tanto, los grafos mostrados en figura pueden ser definidos de forma equivalente por (X,L) o por (X, Ady).



Caminos

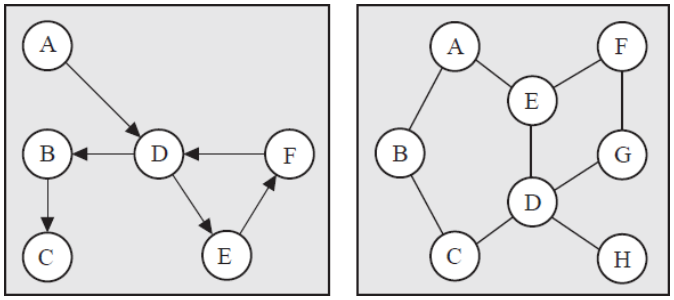
- El conjunto adyacente de un nodo *Xi* contiene los nodos que son directamente alcanzables desde *Xi*.
  - Comenzando en un nodo dado y pasando de forma sucesiva a uno de sus nodos adyacentes, se puede formar un ***camino*** a través del grafo.
    - El concepto de camino entre dos nodos juega un papel central en la teoría de grafos.

Caminos

- **Camino entre dos nodos.**
  - *Un camino del nodo Xi al nodo Xj es un sucesión de nodos (Xi1, . . . , Xir), comenzando en Xi1 = Xi y finalizando en Xir = Xj , de forma que existe una arista del nodo Xik al nodo Xik+1, k = 1, . . . , r − 1, es decir, Xik+1 ∈ Ady(Xik), k = 1, . . . , r − 1.*
- *La longitud del camino, (r − 1), es el nº aristas que contiene.*
- En el caso de grafos no dirigidos, un camino (*Xi*1, . . . , *Xir*) puede representarse mediante *Xi*1− . . . − *Xir* , indicando el carácter no dirigido de las aristas.
- Un camino en un grafo dirigido se representa mediante *Xi*1→ . . . → *Xir* .

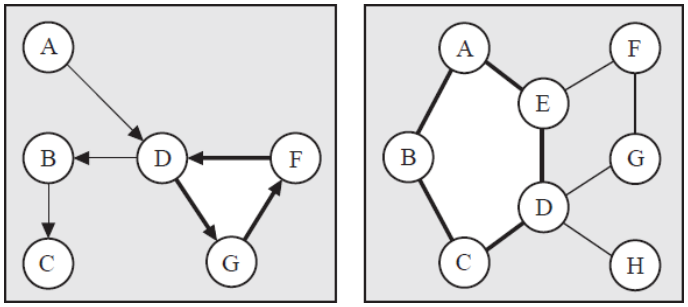
Caminos. Ejemplo

- Sea el grafo dirigido de la 1ª figura.
  - Existe un único camino de longitud 2 de *D* a *F* en este grafo, $D \rightarrow E \rightarrow F$.
  - Por otra parte, existe un camino de *A* a *B* de longitud 2, $A \rightarrow D \rightarrow B$, y otro de longitud 5, $A \rightarrow D \rightarrow E \rightarrow F \rightarrow D \rightarrow B$.
  - Por el contrario, no existe ningún camino de *B* a *A*.
- Sea el grafo no dirigido de la 2ª figura. Existe al menos un camino entre cada par de nodos del grafo. Por ejemplo, algunos de los caminos entre *A* a *H* son:
  - $A - E - D - H$, de longitud 3,
  - $A - B - C - D - H$, de longitud 4, y
  - $A - E - F - G - D - H$, de longitud 5.
- Nota: en un grafo dirigido han de tenerse en cuenta las direcciones de las aristas para formar un camino. Por ejemplo, en el grafo dirigido de la 1ª figura existe un camino de *A* a *C* ($A \rightarrow D \rightarrow B \rightarrow C$), pero no existe camino que una los nodos en sentido inverso.



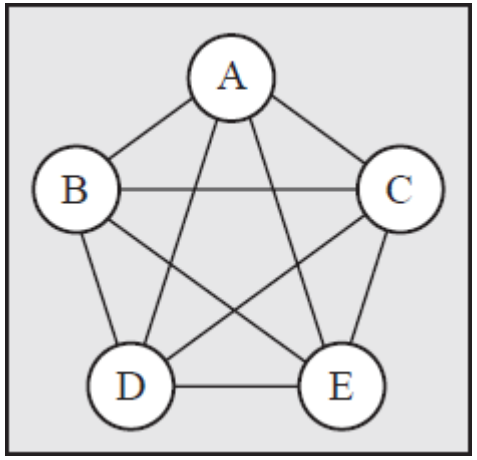
Camino cerrado

- **Camino cerrado.**
  - *Un camino (Xi1, . . . , Xir) se dice que es cerrado si el nodo inicial coincide con el final, es decir, Xi1 = Xir .*
- **Ejemplo Caminos cerrados.**
  - El camino $D \rightarrow G \rightarrow F \rightarrow D$ en el grafo dirigido de la 1ª figura es un camino cerrado.
  - El grafo no dirigido dado en la 2ª figura contiene varios caminos cerrados como, por ejemplo, el camino $A - B - C - D - E - A$.
- Si un camino contiene un nodo más de una vez, entonces el camino contiene un subcamino cerrado. Por ejemplo, en el grafo de la 2ª figura, el camino $C - D - E - F - G - D - H$ contiene dos veces el nodo D. Por tanto, este camino ha de contener un subcamino cerrado: $D–E–F–G–D$.
- Eliminando este camino cerrado, se puede hallar un camino más corto entre los nodos extremos, $C - D - H$.



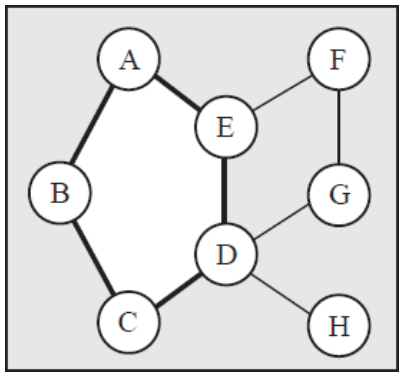
Características de los Grafos no Dirigidos

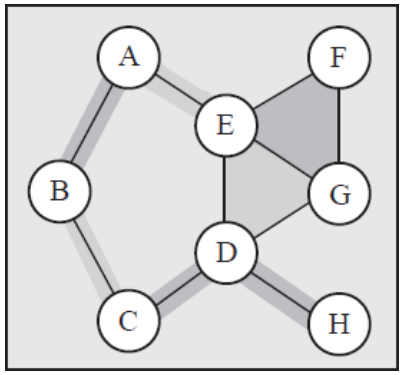
- Se presentan algunas características de los grafos no dirigidos.
- **Grafo completo.**
  - *Un grafo no dirigido se denomina completo si contiene una arista entre cada par de nodos.*
- Existe un único grafo completo de *n* nodos, denotado por ***Kn***. La figura muestra gráficamente *K*5.



Conjunto completo

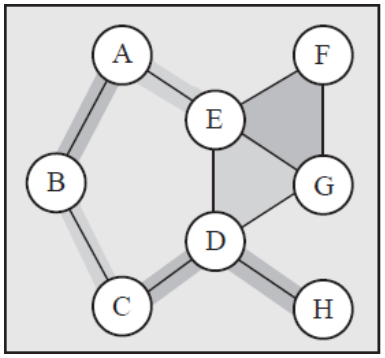
- **Conjunto completo.**
  - *Un subconjunto de nodos S de un grafo G se denomina completo si existe una arista en G para cada par de nodos en S.*
- Consecuencia: cualquier par de nodos adyacentes en un grafo forma un conjunto completo.
- Por ejemplo, el grafo de la 1ª figura no contiene conjuntos completos con más de dos nodos.
- Por el contrario, el grafo mostrado en la 2ª figura contiene dos subconjuntos completos de tres nodos: *{D,E,G}* y *{E,F,G}*.
- **Conjunto completo maximal (conglomerado).**
  - *Un conjunto completo S de G es maximal si al añadir cualquier nodo de G a S, éste deja de ser completo*
- Los conjuntos completos maximales de un grafo desempeñan un papel fundamental en la caracterización de su estructura topológica.

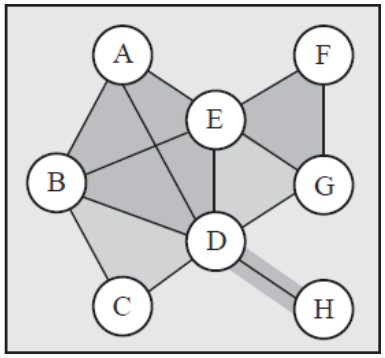




Conglomerados

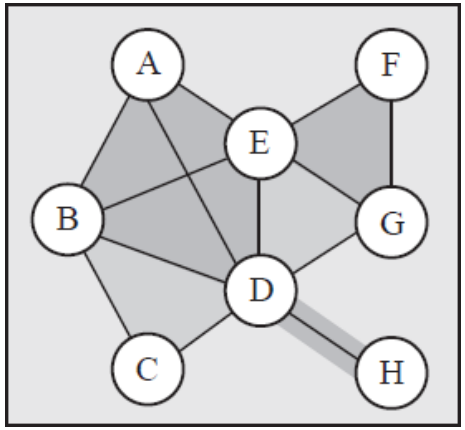
- **Definición Conglomerado.** *Un conjunto completo de nodos C se denomina un conglomerado si no es subconjunto propio de otro conjunto completo, es decir, si es maximal.*
- El grafo mostrado en la 1ª figura contiene los siguientes conglomerados: C1 = {A,B}, C2 = {B,C}, C3 = {C,D}, C4 = {D,H}, C5 = {D,E,G}, C6 = {E,F,G} y C7 = {A,E}. (En color más oscuro)
- Sin embargo, si se añade alguna arista al grafo, alguno de estos conglomerados ya no será un conjunto maximal y el conjunto de conglomerados del nuevo grafo será distinto.
- Por ejemplo, en el grafo de la 2ª figura, obtenido añadiendo tres aristas al grafo de la 1ª, los conjuntos C1, C2, C3 y C7 ya no son completos. El nuevo grafo contiene solamente cinco conglomerados: C1 = {A,B,D,E}, C2 = {B,C,D}, C3 = {D,H}, C4 = {D,E,G}, y C5 = {E,F,G}.





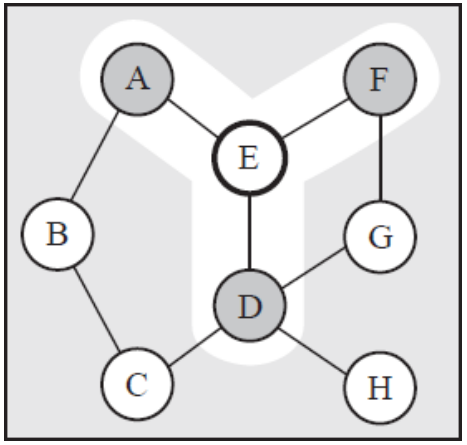
Bucle

- **Bucle.**
  - *Un bucle es un camino cerrado en un grafo no dirigido.*
- **Ejemplo.**
  - Sea el grafo no dirigido mostrado en la figura.
  - El camino cerrado $A - B - C - D - E - A$ es un bucle de longitud 5.
  - Si en un bucle se reemplaza un camino entre dos nodos por un camino alternativo, se obtiene un nuevo bucle. Por ejemplo, si se reemplaza la arista *D−E* por el camino *D−G−F−E* en el bucle anterior, se obtiene un nuevo bucle de longitud 7: *A−B−C−D−G−F −E−A*.



Vecindad

- **Vecinos de un nodo.**
  - *El conjunto de nodos adyacentes a un nodo Xi en un grafo no dirigido se denomina conjunto de vecinos de Xi, Vec(Xi) = {Xj | Xj ∈ Ady(Xi)}.*
- En el caso de grafos no dirigidos, el conjunto de nodos adyacentes a un nodo dado coincide con el conjunto de vecinos de dicho nodo.
- Por ejemplo, los nodos sombreados en la figura, *{A,D, F}*, son los vecinos del nodo *E*.

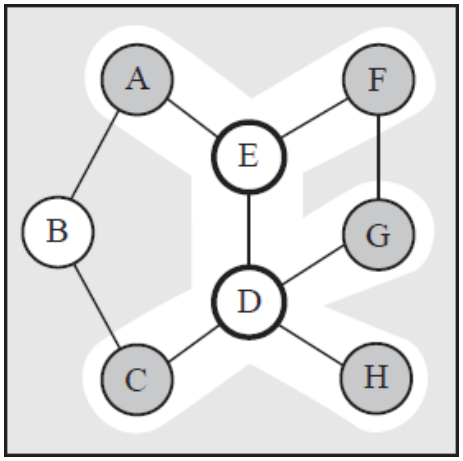


Frontera

- **Frontera de un conjunto de nodos.**
  - *La unión de los conjuntos de vecinos de los nodos de un conjunto dado, S, excluyendo los nodos de S, se denomina la frontera de S y se denota por Frn(S), donde X \ S es el conjunto de nodos de X excluyendo los de S.*

$$Frn(S) = \left(\bigcup_{X_i \in S} Vec(X_i) \right) \setminus S,$$

  - Por ejemplo, los nodos sombreados en la figura, *{A,C, F, G,H}*, son la frontera del conjunto *{D,E}*.
  - Si *S* contiene 1 nodo, la frontera es la vecindad.

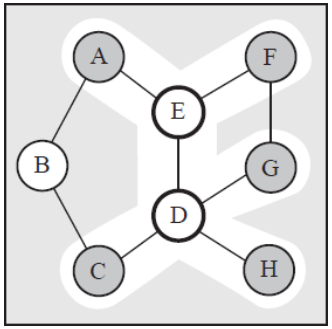


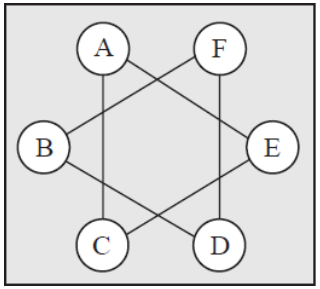
Tipos de Grafos no Dirigidos

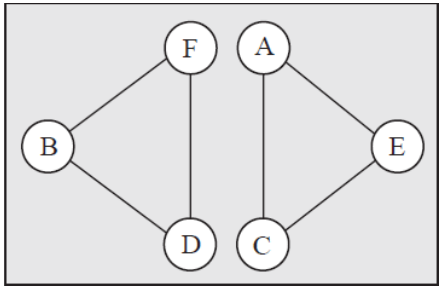
- En muchas situaciones prácticas es importante conocer si existe un camino entre un par de nodos dados.
- Por ejemplo, en el campo de los **sistemas expertos**, los grafos se utilizan para representar relaciones de dependencia entre las variables que componen el sistema.
  - En estos casos, es muy útil conocer el ***nº de posibles caminos entre dos nodos***, a efectos de **entender la estructura de dependencia** contenida en el grafo.
  - Desde este punto de vista, una clasificación útil de los grafos debe tener en cuenta el nº de caminos distintos existentes entre cada par de nodos.

Grafos conexos no dirigidos

- *Un grafo no dirigido se denomina conexo si existe al menos un camino entre cada par de nodos. En caso contrario, el grafo se denomina inconexo.*
- Por ejemplo, el grafo de la 1ª figura es un grafo conexo. Sin embargo, el grafo representado en la 2ª figura es inconexo pues, por ejemplo, no existe ningún camino entre los nodos *A* y *F*.
- El grafo de la 2ª figura parece conexo a primera vista, pues las aristas se cruzan. En la 3ª figura queda más patente que no es así.
- La representación gráfica de un grafo se analiza en detalle más adelante.





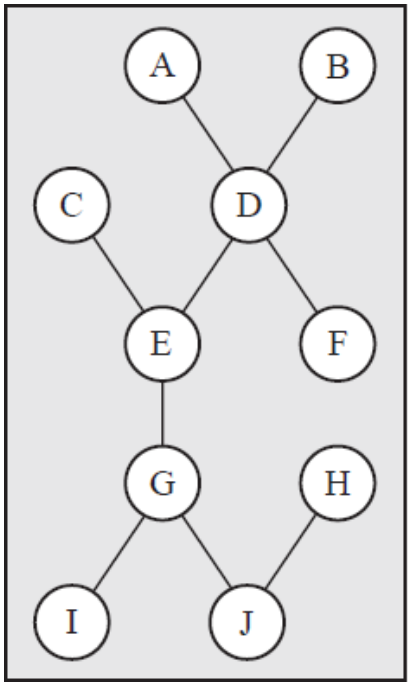


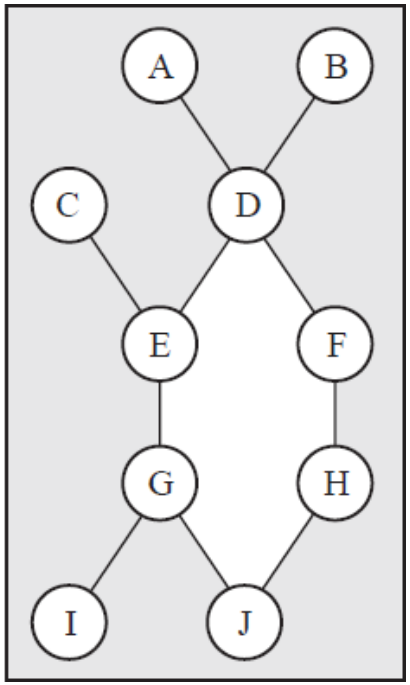
Componentes conexas

- Un grafo inconexo puede dividirse en un conjunto de grafos conexos llamados ***componentes conexas***.
  - Por ejemplo, el grafo inconexo anterior contiene las componentes conexas *{A,C,E}* y *{B,D,F}*. ➔ En la práctica, se suponen los grafos conexos (en caso contrario, podría argumentarse sobre cada una de las componentes conexas de forma análoga)
- Más adelante se muestra un algoritmo para determinar si un grafo es conexo, y calcular sus componentes conexas, caso de no serlo.
- La complejidad topológica de un grafo aumenta con el nº de caminos distintos entre dos nodos. ➔ Además de considerar la existencia de un camino entre dos nodos, se ***ha de considerar también el nº de caminos posibles***.

Árbol

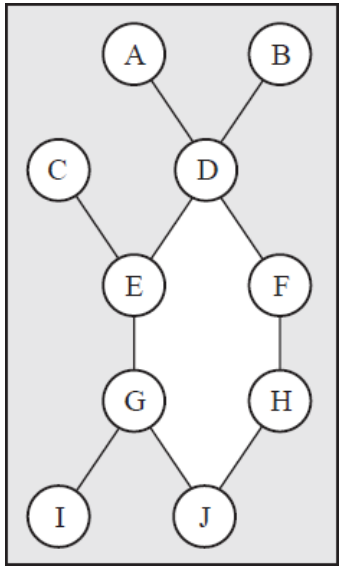
- ***Un grafo conexo no dirigido se denomina un árbol si existe un único camino entre cada par de nodos***.
- Así, un árbol es un grafo conexo, en el que si se elimina una cualquiera de sus aristas, el grafo se vuelve inconexo.
- Además, un árbol no contiene bucles, pero si se añade una arista cualquiera al grafo se forma un bucle.
- La 1ª figura muestra un ejemplo de árbol.
  - La eliminación de una cualquiera de sus aristas divide al grafo en dos partes inconexas.
  - Si se añade al grafo una arista cualquiera, como en la 2ª figura, se crear un bucle en el grafo, que deja de ser un árbol.



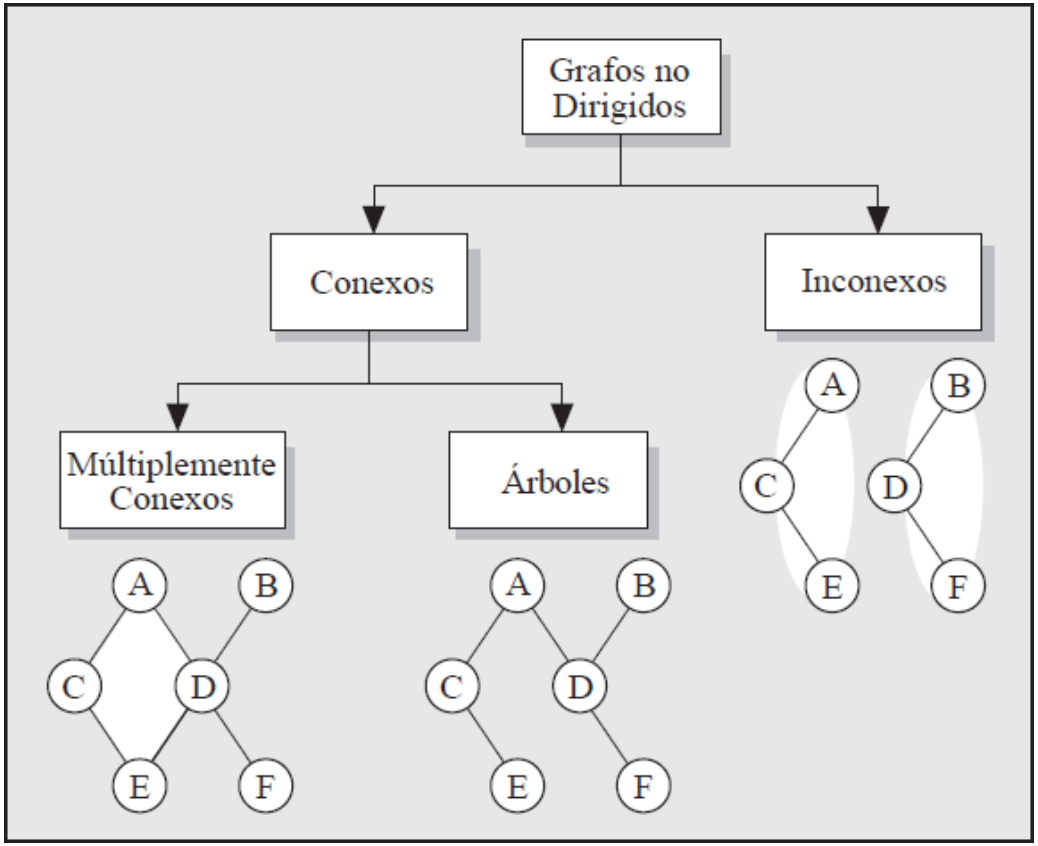


Grafo múltiplemente conexo

- ***Un grafo conexo se denomina múltiplemente conexo si contiene al menos un par de nodos unidos por más de un camino o, equivalentemente, si contiene al menos un bucle.***.
- Si un grafo contiene dos caminos distintos entre un par de nodos, pueden combinarse para formar un bucle.
  - Por tanto, las dos definiciones anteriores son efectivamente equivalentes.
  - Por ejemplo, el grafo de la figura es múltiplemente conexo, pues existen los caminos *D−E −G−J* y *D −F −H −J* que unen los nodos *D* y *J*. Estos dos caminos forman el bucle *D − E − G − J − H − F − D*.

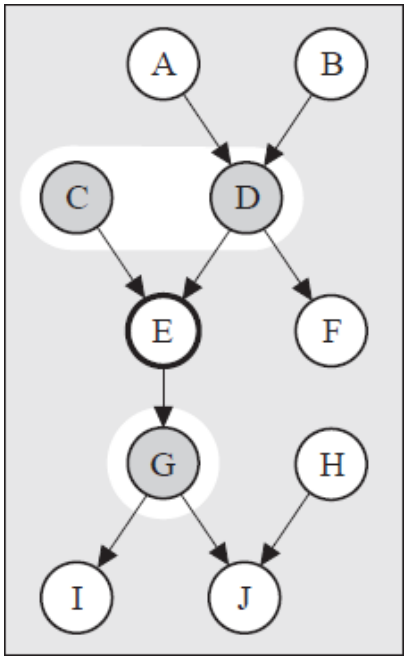


Resumen Grafos No Dirigidos



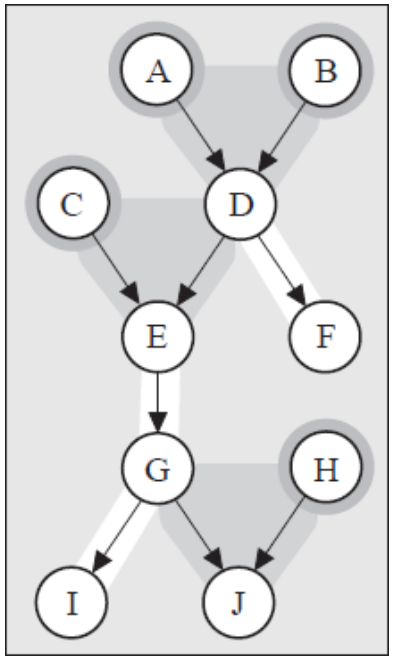
Características de los Grafos Dirigidos

- Se describen las características prinipales de los grafos dirigidos.
- **Padre e hijo.**
  - *Cuando existe una arista dirigida, $Xi \rightarrow Xj$, del nodo Xi al nodo Xj , entonces se dice que el nodo Xi es un padre del nodo Xj , y que el nodo Xj es un hijo de Xi.*
- El conjunto de padres de un nodo *Xi* se denota mediante ΠXi o simplemente Πi.
  - Por ejemplo, los nodos *C* y *D* son los padres del nodo *E* en el grafo de la figura.
- En un grafo dirigido, el conjunto de hijos de un nodo coincide con el conjunto de nodos adyacentes.



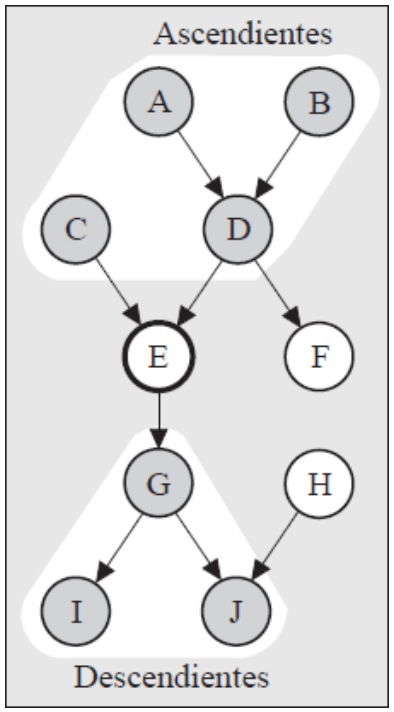
Familia de un nodo

- **Familia de un nodo.**
  - *Conjunto formado por un nodo y sus padres.*
- Las zonas sombreadas en el grafo de la figura muestran las familias asociadas a este grafo.
- En este ejemplo se pueden observar familias con uno, dos y tres nodos.
- Las familias de un grafo juegan un papel muy importante, pues la estructura de dependencias codificada en un grafo dirigido podrá trasladarse a una función de probabilidad definiendo ***distribuciones de probabilidad locales sobre cada familia del grafo***.



Ascendientes y descendientes de un nodo

- **Ascendientes de un nodo.**
  - *Un nodo Xj es ascendiente del nodo Xi si existe un camino de Xj a Xi.*
- **Conjunto ancestral.**
  - *Conjunto de nodos S que contiene los ascendientes de todos sus nodos.*
- **Descendientes de un nodo**.
  - *Un nodo Xj se denomina descendiente del nodo Xi si existe un camino de Xi a Xj (es decir, Xi es ascendiente de Xj).*

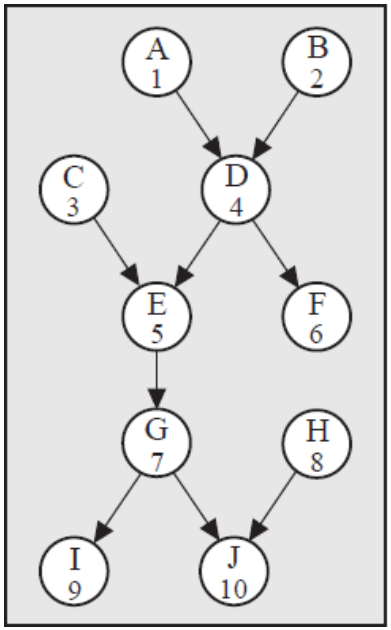


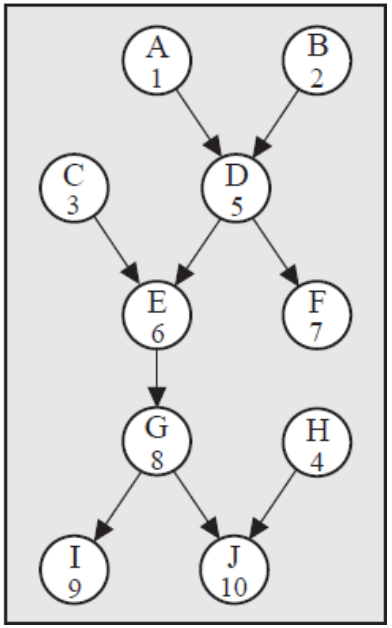
Ordenación

- Se han analizado atributos de los nodos de un grafo referidos a su relación de dependencia con el resto de los nodos (padres, hijos, familia, etc.).
- En ocasiones esta estructura de dependencia, u otras propiedades topológicas del grafo, pueden plasmarse de forma global en una ordenación de los nodos $X = \{X1, \ldots, Xn\}$.
- **Ordenación.**
  - *Dado un conjunto* $X = \{X1, \ldots, Xn\}$ *de nodos, una ordenación,* a*, es una biyección que asigna un n° del conjunto* $\{1, \ldots, n\}$ *a cada nodo:*
    - $a : \{1, \ldots, n\} \longrightarrow \{X1, \ldots, Xn\}$.
  - *Por tanto,* $a(i)$ *denota el i-ésimo nodo de la numeración. Una numeración puede representarse mediante la sucesión ordenada de nodos* $(a(1), \ldots, a(n))$.

Numeración ancestral

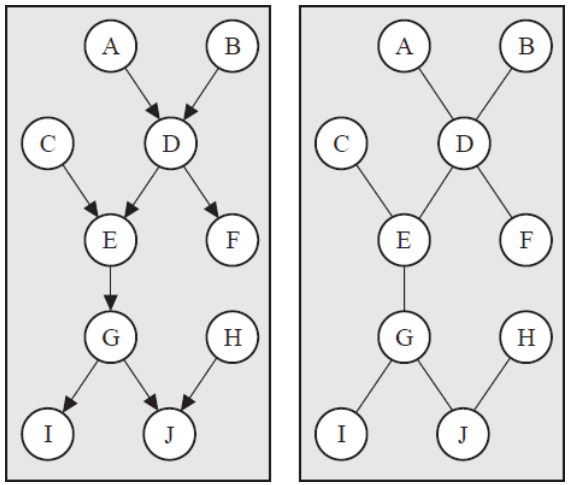
- Muestra la estructura de ascendientes/descendientes de forma global.
- **Numeración ancestral.**
  - *Una numeración de los nodos de un grafo dirigido se denomina ancestral si el número correspondiente a cada nodo es menor que los correspondientes a sus hijos.*
- Por ejemplo, las dos numeraciones mostradas en las figuras son dos numeraciones ancestrales distintas del mismo grafo.
  - Por tanto, este tipo de numeración no es única.
- Existen grafos dirigidos que no admiten numeración ancestral. (Se analizará más adelante, desde un punto de vista teórico y algorítmico).





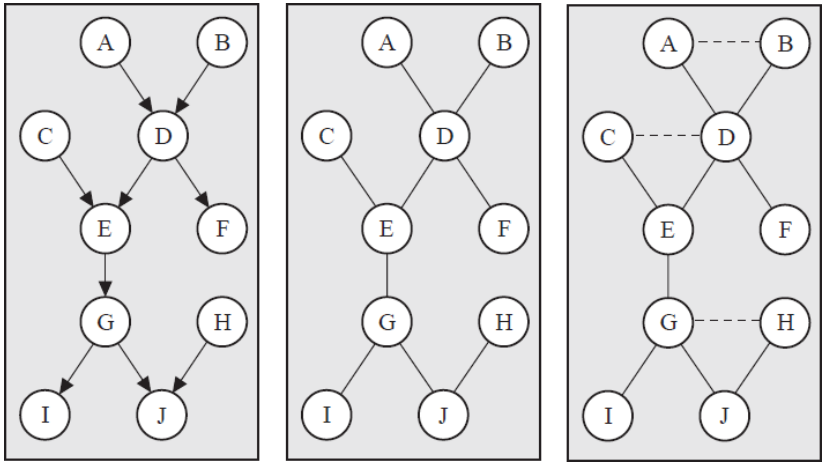
Grafo no dirigido asociado a un grafo dirigido

- Un grafo dirigido puede convertirse a no dirigido, eliminando la direccionalidad de sus aristas.
  - El inverso es más complejo, al existir dos alternativas para orientar una arista $X_i - X_j$: $X_i \rightarrow X_j$ o $X_j \rightarrow X_i$. Por tanto, se pueden definir varios grafos dirigidos asociados a un mismo grafo no dirigido.
- **Grafo no dirigido asociado a un grafo dirigido.**
  - *Dado un grafo dirigido, el grafo no dirigido obtenido al reemplazar cada arista dirigida del grafo por la correspondiente arista no dirigida se denomina el grafo no dirigido asociado.*
  - *Simplemente, se elimina la direccionalidad de las aristas.*



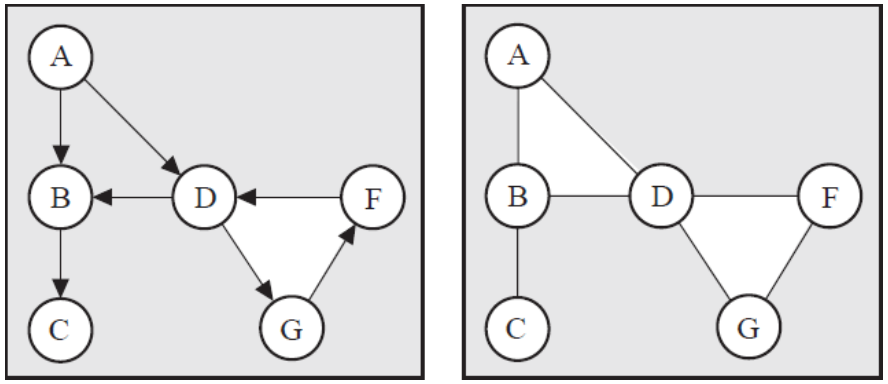
Grafo moral

- **Grafo moral.**
  - *El grafo **no dirigido** asociado al grafo dirigido que se obtiene al añadir una arista entre cada par de nodos con algún hijo común en un grafo no dirigido, se denomina el grafo moral asociado a dicho grafo.*
- La figura de la derecha muestra el grafo moral correspondiente al dirigido. Cada par de nodos (*A,B*), (*C,D*) y (*G,H*) tienen un hijo común en este grafo. Por tanto, el grafo moral asociado se forma añadiendo las aristas indicadas con línea discontinua y eliminando la direccionalidad de todas las aristas.



Caminos cerrados

- Los caminos cerrados reciben dos nombres distintos en un grafo dirigido, según se tenga en cuenta o no la direccionalidad de las aristas. Cuando un camino cerrado está definido en el grafo dirigido original se denomina un *ciclo*; en cambio, cuando se define sobre el grafo no dirigido asociado, se denomina bucle.
- **Ciclo.**
  - *Es un camino cerrado en un grafo dirigido.*
- **Bucles y ciclos.** La figura de la izquierda muestra un grafo dirigido que contiene un sólo ciclo: $D \rightarrow G \rightarrow F \rightarrow D$. Sin embargo, el grafo no dirigido asociado contiene dos bucles: $D-G-F-D$ y $A-B-D-A$.

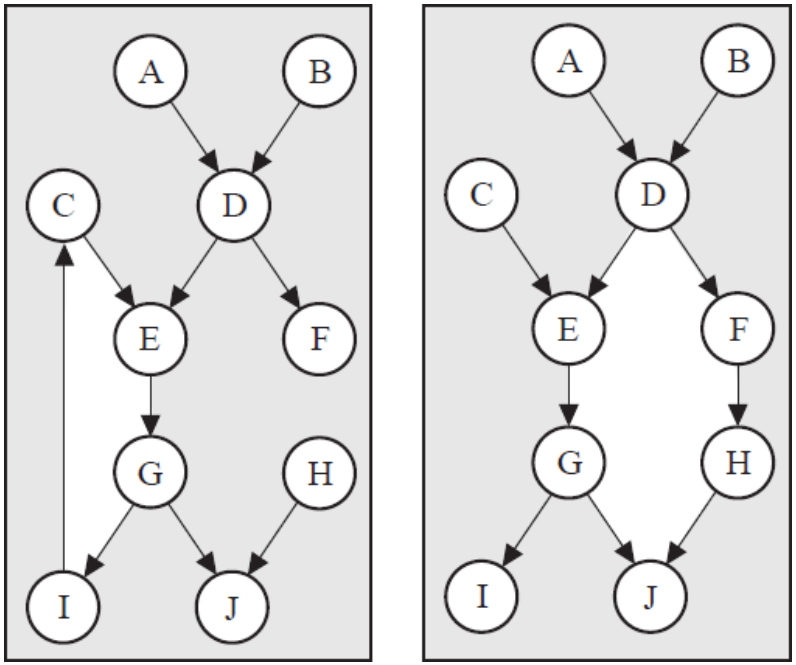


Tipos de Grafos Dirigidos

- **Grafos dirigidos conexos.**
  - *Un grafo dirigido se denomina conexo si el grafo no dirigido asociado es conexo; en caso contrario se denomina inconexo.*
- **Árboles y grafos múltiplemente conexos.**
  - *Un grafo dirigido conexo se denomina árbol si el grafo no dirigido asociado es un árbol; en caso contrario se denomina múltiplemente conexo.*
- **Grafos cíclicos y acíclicos.**
  - *Un grafo dirigido se denomina cíclico si contiene al menos un ciclo; en caso contrario se denomina grafo dirigido acíclico.*
- Los grafos dirigidos acíclicos juegan un papel muy importante, sirviendo como base para construir los modelos probabilísticos basados en *Redes Bayesianas*.

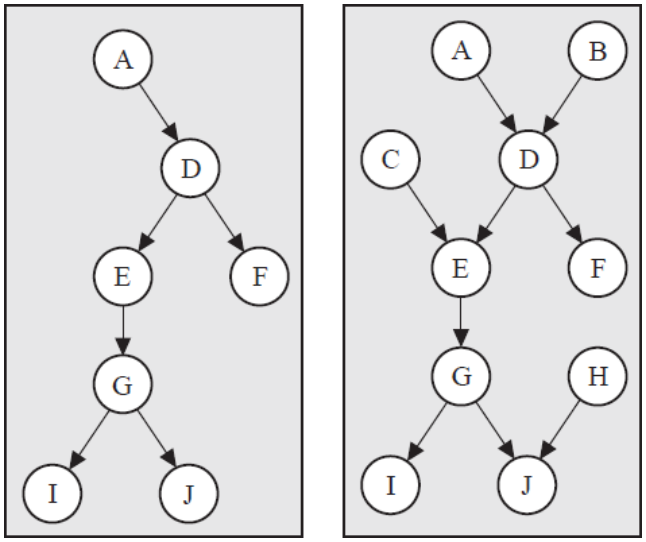
Tipos de Grafos Dirigidos

- La figura muestra un grafo cíclico y uno múltiplemente conexo.

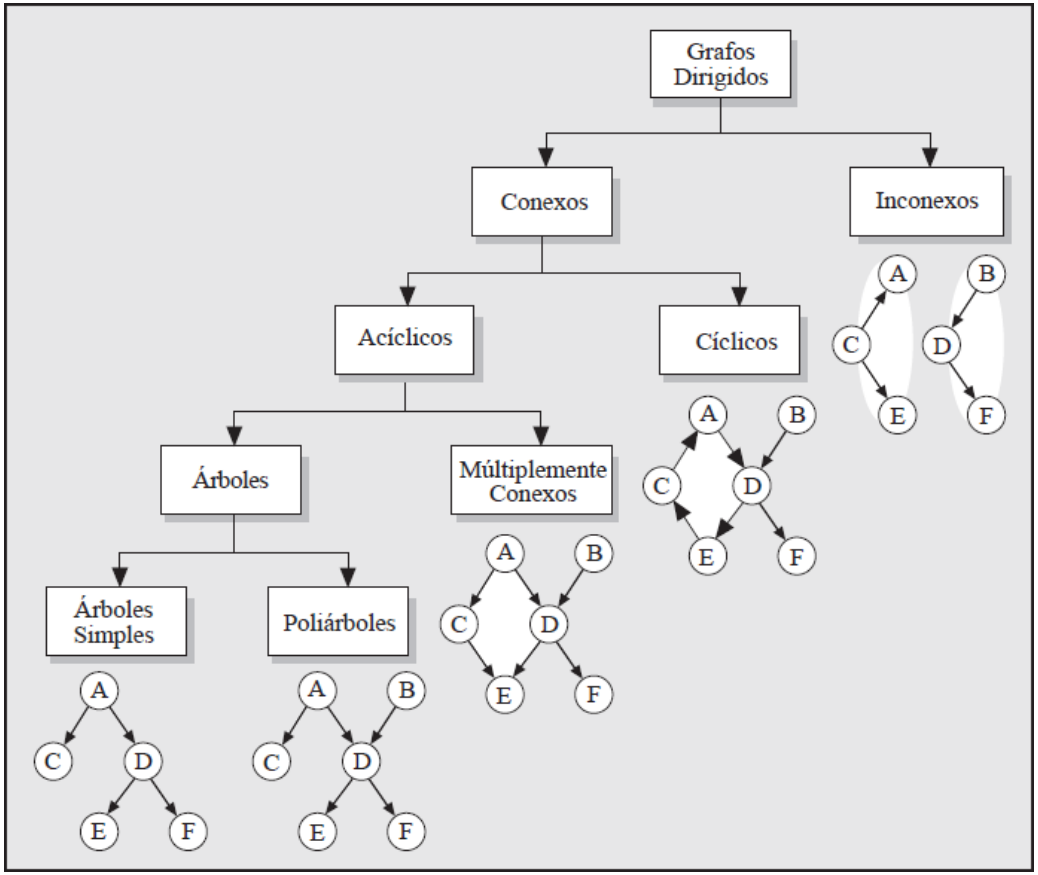


Tipos de Grafos Dirigidos

- **Grafos simples y poliárboles.**
  - *Un árbol dirigido se denomina un árbol simple si cada nodo tiene como máximo un padre; en caso contrario se denomina un poliárbol.*
- Las figuras muestran un ejemplo de árbol simple y un ejemplo de poliárbol. La figura de la izquierda muestra un grafo cíclico y la de la derecha uno múltiplemente conexo.



Resumen Grafos Dirigidos

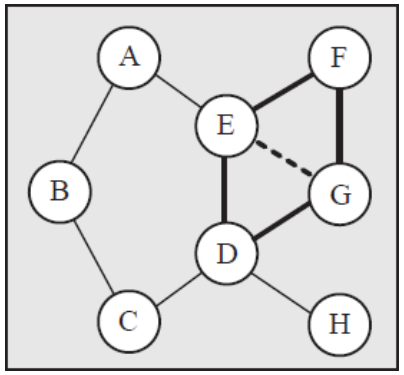


Grafos triangulados

- Los grafos triangulados son un tipo especial de grafos no dirigidos que tienen muchas aplicaciones prácticas interesantes en varios campos. Por ejemplo, veremos que este tipo de grafos constituyen la estructura gráfica del tipo de modelos probabilísticos conocidos como *modelos descomponibles* (Lauritzen, Speed y Vijayan (1984)).
- También reciben el nombre de ***circuitos rígidos*** (Dirac (1961)) y ***grafos cordales*** (Gavril (1972, 1974)).
- Introducimos los grafos triangulados, así como una serie de algoritmos para comprobar si un grafo es triangulado y cómo triangularlo en caso de que no lo sea.

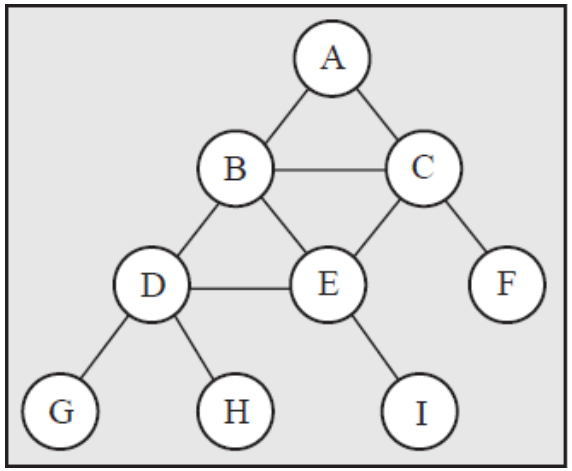
Cuerda de un bucle

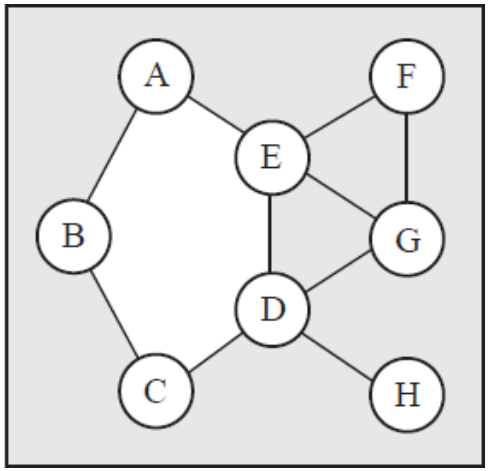
- **Cuerda de un bucle**
  - *Una cuerda es una arista que une dos nodos de un bucle y que no pertenece al bucle.*
- Por ejemplo, en el grafo de la figura, la arista *E* − *G* es una cuerda del bucle *E* − *F* − *G* − *D* − *E*. Observe que la cuerda divide el bucle en dos bucles menores: *E* − *F* − *G* − *E* y *E* − *G* − *D* − *E*.
- Por otra parte, el bucle *A* − *B* − *C* − *D* − *E* − *A* no contiene ninguna cuerda.
- Los bucles de longitud 3 son los únicos que no pueden poseer cuerdas. Por ello, estos son los menores elementos en los que puede descomponerse un bucle mediante la incorporación de cuerdas en el grafo.
- Los bucles de longitud 3 se denominan *triángulos*.



Grafos Triangulados. Definición

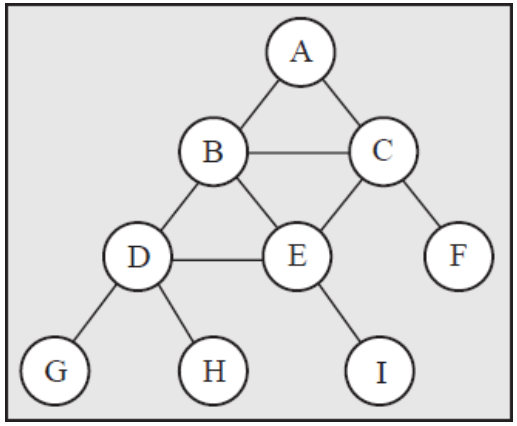
- **Grafo triangulado.**
  - *Un grafo no dirigido se denomina triangulado, o cordal, si cada bucle de longitud mayor o igual que cuatro contiene al menos una cuerda.*
- **Ejemplo.**
  - La figura de la izquierda muestra un grafo triangulado. El grafo contiene dos bucles de longitud cuatro, *A−B−E−C−A* y *B−C−E−D−B*, y un bucle de longitud cinco, *A−B−D−E−C−A*, y cada uno de ellos tiene al menos una cuerda.
  - El grafo de la derecha no es triangulado, pues contiene al bucle *A−B−C −D−E −A*, que no posee ninguna cuerda.

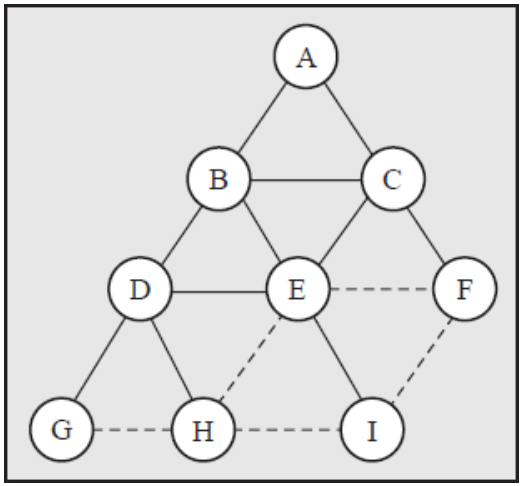




Triangulación

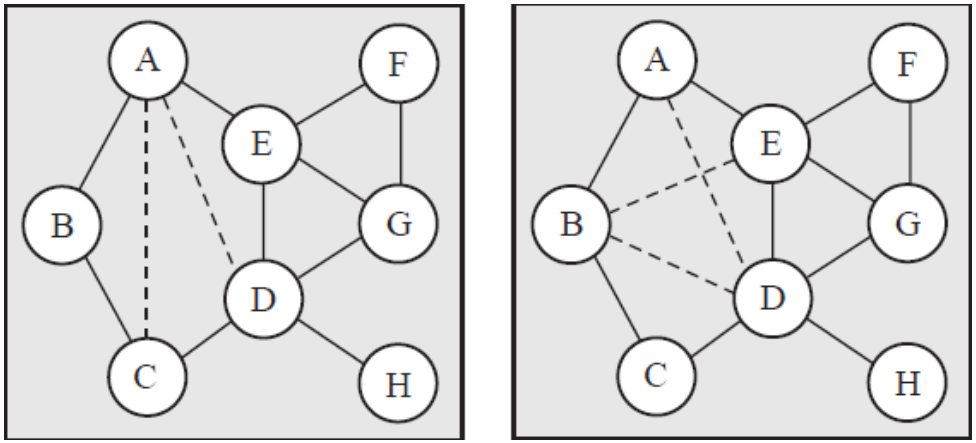
- Si un grafo no es triangulado, es posible convertirlo en triangulado añadiendo cuerdas que dividan los bucles. Este proceso se denomina ***rellenado*** o ***triangulación***.
- Es importante destacar que triangular un grafo no consiste en dividirlo en triángulos. Por ejemplo, el grafo de la izquierda es triangulado y, por tanto, no necesita la adición de aristas extra, como aquellas que se indican mediante líneas de puntos en la figura de la derecha.





Triangulaciones alternativas

- Puesto que un bucle puede romperse de varias formas distintas con una cuerda, existen varias formas distintas de triangular un grafo.
- En otras palabras, la triangulación de un grafo no tiene por qué ser única.
- Por ejemplo, los dos grafos mostrados en la figura corresponden a dos triangulaciones distintas asociadas con el grafo no triangulado anterior.

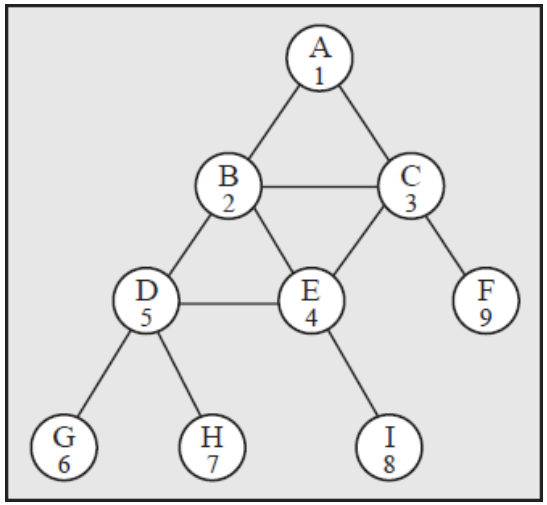


Triangulación minimal

- Para mantener lo más posible la topología original del grafo en el proceso de triangulación, es importante añadir el mínimo nº de aristas posible.
- En este sentido, una triangulación se dice ***minimal*** si contiene un nº mínimo le cuerdas por debajo del cual no es posible triangular el grafo original.
- La 1ª triangulación de la diapositiva anterior es minimal. En cambio la 2ª triangulación mostrada no lo es, pues puede eliminarse la arista $A - D$ o la $B - E$ y el grafo resultante sigue siendo triangulado.
- El problema de calcular una triangulación minimal es *NP*-complejo (NP-hard) (Yannakakis (1981)).
  - Dada la complejidad de este problema, han sido desarrollados varios algoritmos de ejecución en tiempo lineal para triangular un grafo (Rose, Tarjan y Leuker (1976), Tarjan y Yannakakis (1984)); sin embargo, ninguno de ellos garantiza que la triangulación resultante sea minimal.
- A continuación se introducirá un algoritmo simple llamado ***algoritmo de búsqueda de máxima cardinalidad*** (ver Tarjan y Yannakakis (1984)), aunque antes serán necesarias algunas definiciones.

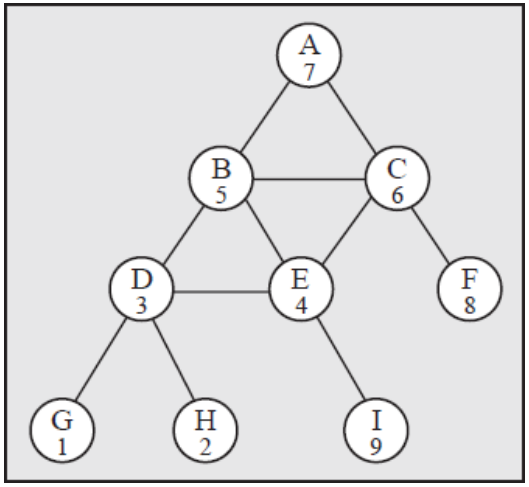
Numeración perfecta

- **Numeración perfecta.**
  - *Una numeración de los nodos de un grafo, a, se denomina perfecta si el subconjunto de nodos Frn(a(i)) ∩ {a(1), . . . , a(i − 1)} es completo para i = 2,...,n.*
- Ejemplo: la figura muestra una numeración de los nodos del grafo: $a(1) = A$, $a(2) = B$, $a(3) = C$, $a(4) = E$, etc. Se comprueba que se verifican las condiciones de numeración perfecta:
  - Para $i = 2$, $Frn(a(2)) \cap \{a(1)\} = Frn(B) \cap \{A\} = \{A,C,D,E\} \cap \{A\} = \{A\}$, que es trivialmente un conjunto completo.
  - Para $i = 3$, $Frn(a(3)) \cap \{a(1), a(2)\} = \{A,B,E, F\} \cap \{A,B\} = \{A,B\}$ es completo, pues la arista $A - B$ está contenida en el grafo.
  - Para $i = 4$, $Frn(a(4)) \cap \{a(1), a(2), a(3)\} = \{B,C,D, I\} \cap \{A,B,C\} = \{B,C\}$ tambi´en es completo.
  - Tb. se cumple para $i = 5, \ldots, 9$.
- Por tanto, *a* es una numeración perfecta.



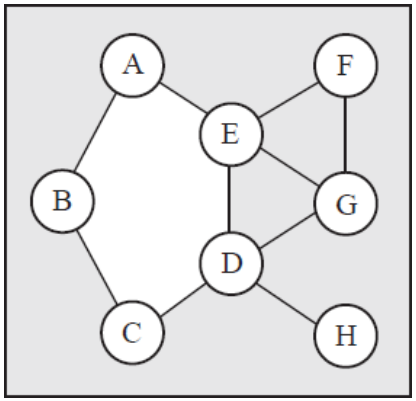
Numeración perfecta

- Ejemplo: observe la figura:



- También se trata de una numeración perfecta.
- Nótese por tanto que la numeración perfecta no es necesariamente única.

- Hemos mostrado dos numeraciones perfectas para el mismo grafo.
- Por otra parte, también existen grafos que no admiten ninguna numeración perfecta. Por ejemplo, el grafo de la figura; la presencia de bucles sin cuerdas hace imposible la numeración perfecta.



Algoritmo de búsqueda de cardinalidad máxima. Bases

- Algoritmo: *maximum cardinality search*.
- Desarrollado por Tarjan y Yannakakis (1984).
- Algoritmo rápido y conceptualmente sencillo para comprobar si un grafo no dirigido es triangulado.
- Se basa en la búsqueda de una numeración perfecta de los nodos del grafo.
- Está basado en el siguiente teorema que relaciona los conceptos de numeración perfecta y grafo triangulado (ver Fulkerson y Gross (1965), y Golumbic (1980)).
- **Triangulación y numeración perfecta.**
  - *Un grafo no dirigido admite una numeración perfecta si y sólo si es triangulado.*
- El algoritmo de máxima cardinalidad genera una numeración de los nodos del grafo que será perfecta sólo en caso de que el grafo esté triangulado.

Algoritmo de búsqueda de cardinalidad máxima. Algoritmo

- **Datos:** Un grafo no dirigido *G* = (*X,L*) y un nodo inicial *Xi*.
- **Resultado:** Una numeración *a* de los nodos de *X*.

1. *Iniciación:* Asignar el primer nº al nodo inicial, es decir, *a*(1) = *Xi*.
2. Repetir la etapa siguiente con *i* = 2, . . . , *n*.
3. *Iteración i:* En la *i*-ésima etapa de iteración, se asigna el nº *i* a un nodo que no haya sido numerado previamente y que tenga el máximo nº de vecinos numerados. Los empates se resuelven de forma arbitraria.

- Se muestra el pseudocódigo para el algoritmo de máxima cardinalidad.

Búsqueda de Máxima Cardinalidad

Datos: Un grafo $G = (X, L)$ y un nodo inicial X_i
Resultado: Una numeración α de los nodos en X

Etapa Inicial:

$\alpha(1) \leftarrow X_i$
$Numerados \leftarrow \{X_i\}$

Etapa Iterativa:

for i = 2 **to** n
  $X_k \leftarrow$ elige un nodo X_k en $X \setminus Numerados$
    con máximo $|Vec(X_k) \cap Numerados|$
  $\alpha(i) \leftarrow X_k$
  añade X_k a $Numerados$

Numeración de máxima cardinalidad

- El siguiente teorema permite reconocer si un grafo es triangulado utilizando el algoritmo de máxima cardinalidad (ver Tarjan (1983) y Tarjan y Yannakakis (1984)).
- **Numeración de máxima cardinalidad.**
  - *Cualquier numeración de los nodos de un grafo triangulado obtenida aplicando el algoritmo de máxima cardinalidad es una numeración perfecta.*
- Cuando la numeración generada no sea perfecta, significará que el grafo no será triangulado. De esta forma, se puede modificar fácilmente el algoritmo para comprobar si un grafo es triangulado.
- Cuando el grafo no sea triangulado, entonces el propio algoritmo añade las aristas necesarias para triangularlo.
- Aspectos computacionales → Tarjan y Yannakakis (1984) o Neapolitan (1990).
- Una eficiente implementación de este algoritmo se ejecutará en tiempo lineal en el tamaño de la red, $o(n+l)$, con n el nº de nodos y l el nº de aristas del grafo.

Triangulación de máxima cardinalidad

- **Datos:** Un grafo no dirigido $G = (X,L)$ y un nodo inicial Xi.
- **Resultado:** Un conjunto de nuevas aristas L', tal que, $G = (X, L \cup L')$ sea triangulado.
- *Etapa de Iniciación:*

1. Inicialmente la nueva lista de aristas es vacía, $L' = \Phi$.
2. Sea $i = 1$ y asigne el primer nº de la numeración al nodo inicial $Xi \rightarrow a(1) = Xi$.

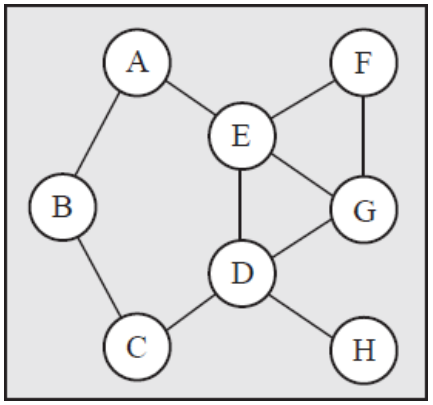
- *Etapa de Iteración:*

3. Se asigna el nº i a un nodo Xk no numerado con máximo nº de vecinos numerados, $a(i) = Xk$.
4. Si $Vec(Xk) \cap \{a(1), \ldots, a(i-1)\}$ no es un conjunto completo, añadir a L' las aristas necesarias para completar el conjunto y volver a la Etapa 2; en caso contrario, ir a la Etapa 5.
5. Si $i = n$, el algoritmo finaliza; en caso contrario, asignar $i = i + 1$ e ir a la Etapa 3.

- Utilizando el teorema anterior, puede demostrarse que cuando un grafo es triangulado, el conjunto de nuevas aristas L' necesarias para triangularlo obtenidas con el algoritmo es vacío; en caso contrario, el conjunto L' contiene las aristas necesarias para triangular el grafo.

Ejemplo Triangulación de máxima cardinalidad

- El grafo de la figura no es un grafo triangulado.



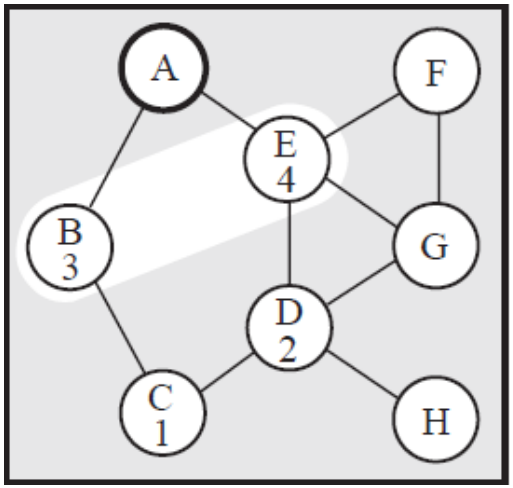
- El Algoritmo anterior permite construir una triangulación del grafo.
- A continuación analizamos la traza del algoritmo eligiendo el nodo C como el nodo inicial.

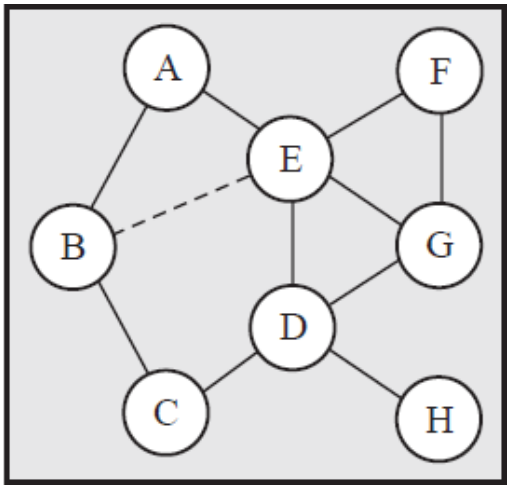
Ejemplo Triangulación de máxima cardinalidad

- Etapa 1: $L' = \Phi$.
- Etapa 2: Sean $i = 1$ y $\alpha(1) = C$.
- Etapa 3: Los nodos B y D son los únicos que tienen un vecino numerado. Deshaciendo el empate de forma arbitraria, se elige el nodo D y se numera con el nº 2, es decir, $\alpha(2) = D$.
- Etapa 4: Nótese que, en este caso, los vecinos previamente numerados forman un conjunto completo. Por tanto, no es necesario añadir ninguna arista a L' y el algoritmo continúa.
- Etapa 5: Puesto que $i = n$, se incrementa en una unidad el contador i y se va a la Etapa 3.
- Etapas 3 – 5: Siguiendo un proceso similar, los nodos B y E se numeran como 3 y 4, respectivamente.

Ejemplo Triangulación de máxima cardinalidad

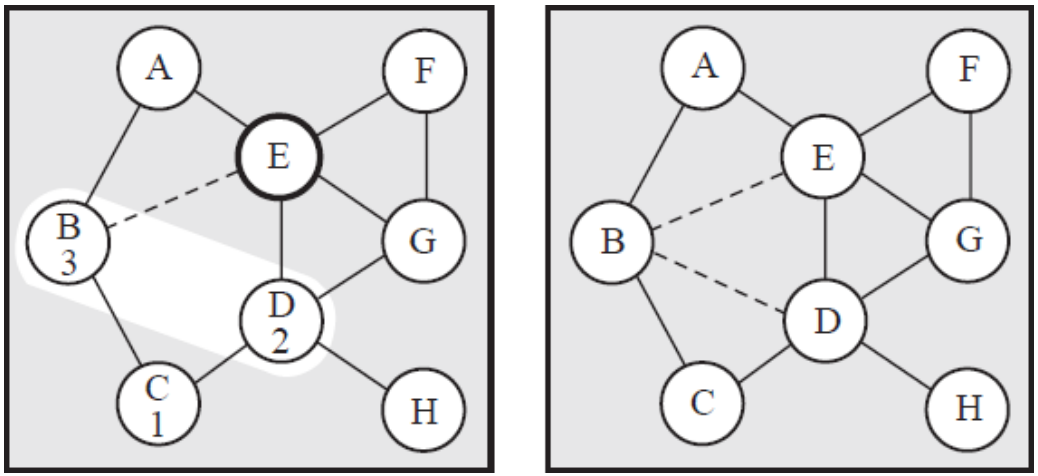
- Etapas 3 – 4: Los nodos con nº máximo de vecinos numerados son *A* y *G*. El empate se deshace eligiendo *A*. Sin embargo, como puede verse en la figura de la izquierda, el conjunto de vecinos numerados de *A*, *{B,E}*, no es un conjunto completo. Por tanto, ha de añadirse la arista *B –E* (ver la figura de la derecha) a *L'* y comenzar de nuevo con la Etapa 2. Nótese que, ahora, *L'* = *{B – E}*.





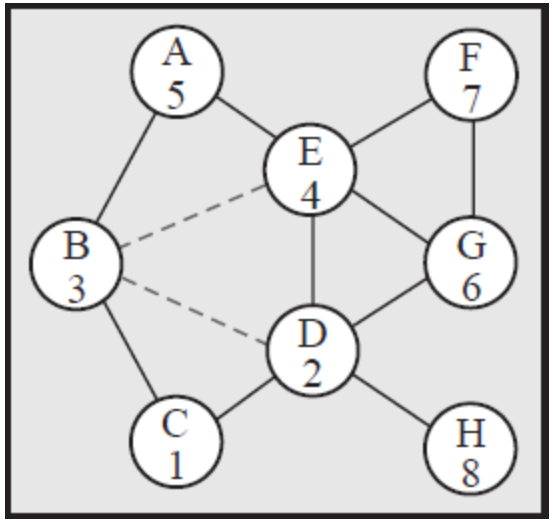
Ejemplo Triangulación de máxima cardinalidad

- Etapas 2 − 5: Los nodos *C*, *D* y *B* se numeran 1, 2 y 3, respectivamente.
- Etapas 3−4: El nodo *E* posee el máximo nº de vecinos numerados, *{B,D}*, pero este conjunto no es completo (ver figura de la izquierda). Por tanto, se añade la arista *B −D* a *L'* y se comienza de nuevo con la Etapa 2. Ahora, *L'* = *{B − E,B − D}* (ver la figura de la derecha).



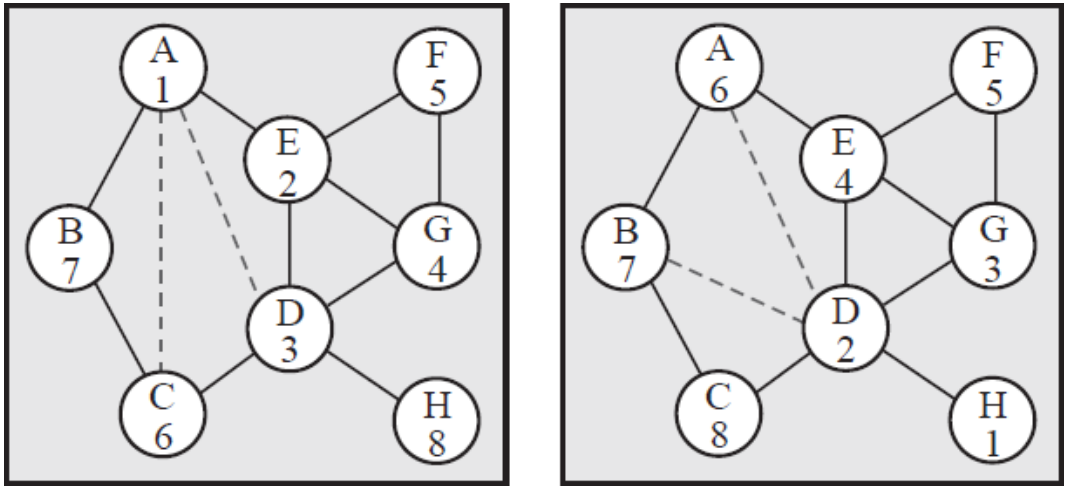
Ejemplo Triangulación de máxima cardinalidad

- Etapas 2 – 5: Los nodos *C,D,B,E, A, G, F* y *H* se numeran sucesivamente de 1 a 8. El grafo resultante $G = (X, L \cup L')$ es un grafo triangulado y la numeración final mostrada es una numeración perfecta.



Ejemplo Triangulación de máxima cardinalidad

- Dependiendo de la elección del nodo inicial y de cómo se deshacen los empates, es posible obtener varias triangulaciones del mismo grafo.
- Por ejemplo, el algoritmo de máxima cardinalidad puede producir , además de la obtenida anteriormente, las dos numeraciones perfectas mostradas en la figura.



Propiedad de intersección dinámica

- Una propiedad interesante de los grafos triangulados, especialmente útil cuando se trabaja con redes de Markov (las veremos más adelante), es la *propiedad de intersección dinámica* (*running intersection property*).
- **Propiedad de intersección dinámica.**
  - *Una numeración de los conglomerados de un grafo no dirigido* $(C1, \ldots, Cm)$ *se dice que satisface la propiedad de intersección dinámica, si el conjunto* $Ci \cap (C1 \cup \ldots \cup Ci-1)$ *está contenido en, al menos, uno de los conglomerados* $\{C1, \ldots, Ci-1\}$, *para todo* $i = 1, \ldots, m$.
- Esta propiedad establece que los conglomerados de un grafo pueden ser ordenados de forma que el conjunto de los nodos comunes a un conglomerado dado y a todos los anteriores esté contenido en alguno de los conglomerados anteriores.

Cadena de conglomerados

- Una sucesión de conglomerados que satisface la propiedad de intersección dinámica se denomina una ***cadena de conglomerados***. Se puede dar el caso de grafos no dirigidos que no poseen ninguna cadena de conglomerados y de grafos que poseen más de una. El siguiente teorema caracteriza los grafos que poseen, al menos, una cadena de conglomerados.
- **Teorema - Cadena de conglomerados.**
  - *Una grafo no dirigido tiene asociada una cadena de conglomerados si y sólo si es triangulado.*
- Se introduce un algoritmo para construir una cadena de conglomerados a partir de un grafo no dirigido.
- Está basado en el algoritmo de máxima cardinalidad y supone que el grafo es triangulado. En caso contrario, el grafo puede ser previamente triangulado utilizando el algoritmo anterior.

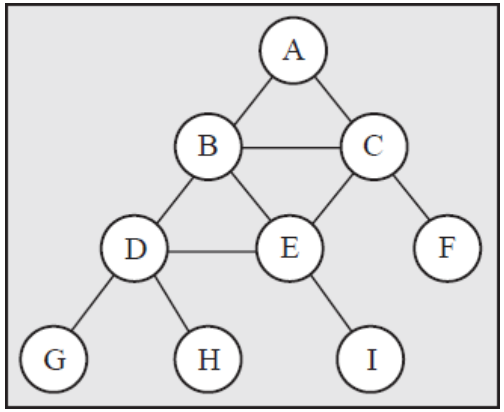
Algoritmo Generación de una cadena de conglomerados

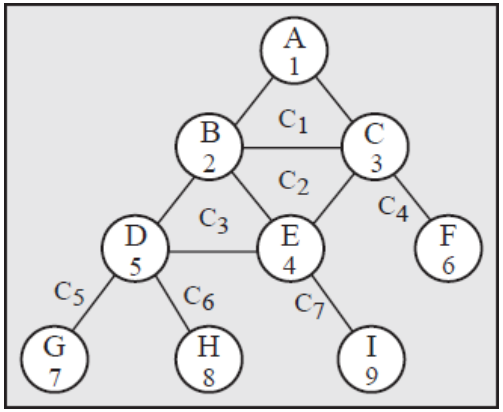
- **Datos:** Un grafo triangulado no dirigido $G = (X,L)$.
- **Resultado:** Una cadena de conglomerados $(C1, \ldots, Cm)$ asociada a G.

1. *Iniciación:* Elegir cualquier nodo como nodo inicial y utilizar el algoritmo de máxima cardinalidad para obtener una numeración perfecta de los nodos, $X1,\ldots, Xn$.
2. Determinar los conglomerados del grafo, C.
3. Asignar a cada conglomerado el máximo de los números (correspondientes a la numeración perfecta) de sus nodos.
4. Ordenar los conglomerados, $(C1, \ldots, Cm)$, en orden ascendente de acuerdo a los números asignados (deshacer empates de forma arbitraria).

Ejemplo Generación de una cadena de conglomerados

- Se aplica el algoritmo para generar una cadena de conglomerados asociada al grafo triangulado dado en la figura de la izquierda.

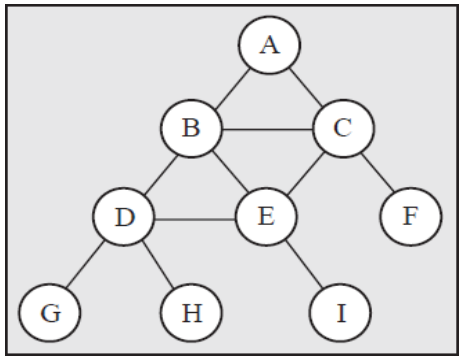


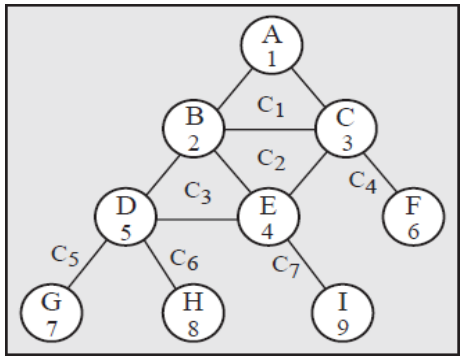


- En primer lugar se utiliza el algoritmo de máxima cardinalidad para obtener una numeración perfecta de los nodos. La figura de la derecha muestra los números obtenidos tomando el nodo A como nodo inicial.

Ejemplo Generación de una cadena de conglomerados

- Los conglomerados son: C1 = {A,B,C}, C2 = {B,C,E}, C3 = {B,D,E}, C4 = {C, F}, C5 = {D,G}, C6 = {D,H} y C7 = {E, I}.
- A continuación, se asigna a cada conglomerado el mayor de los nºs que contenga.
- Por ejemplo, para el caso del conglomerado C1, el mayor nº perfecto asociado a los nodos A, B y C es tres, que corresponde al nodo C. Por tanto, se asigna el nº 3 al conglomerado C1.
- El nº correspondiente al conglomerado C2 es 4 (que corresponde al nodo E), y así sucesivamente.
- Observe que los conglomerados ya se encuentran ordenados de forma ascendente en la ordenación natural. El conglomerado C1 es el que tiene el nº perfecto más bajo, después el C2, y así sucesivamente. Por tanto, (C1, . . . , C7) es una cadena de conglomerados para el grafo.



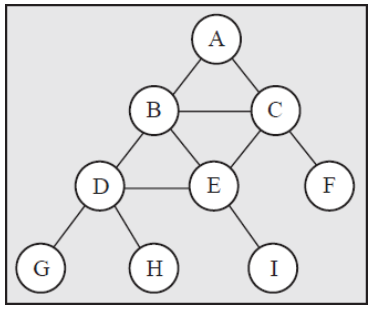


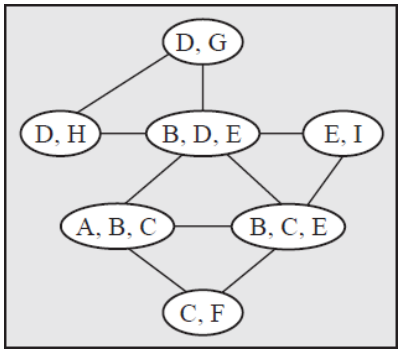
Grafos de Aglomerados

- Los grafos de aglomerados ***se forman agrupando nodos*** con ciertas características comunes de un grafo dado. Este proceso permite obtener nuevos grafos con estructuras topológicas más simples que retienen ciertas propiedades del grafo original.
- En temas posteriores se analizarán varias aplicaciones de los grafos de aglomerados.
- **Definición - Aglomerado.**
  - *Un conjunto de nodos de un grafo se denomina un aglomerado.*
- **Definición - Grafo de aglomerados de un grafo dado.**
  - *Supongamos un grafo G = (X,L) y un conjunto de aglomerados de X, C = {C1, . . . , Cm}, tal que X = C1 ∪ . . . ∪Cm. El grafo G' = (C,L') se denomina un grafo de aglomerados de G si las aristas contenidas en L' sólo unen aglomerados que contengan algún nodo común, es decir, (Ci, Cj) ∈L' ⇒ Ci ∩ Cj ≠ Φ.*
- Un análisis detallado de las propiedades de los grafos de aglomerados se presenta en los libros Beeri y otros (1983) y Jensen (1988) (y las referencias incluidas en ellos).

Grafos de conglomerados

- Los aglomerados de un grafo no son en general conjuntos arbitrarios, pues se desea ***preservar al máximo posible la estructura topológica del grafo original***.
- Se considerarán tipos especiales de grafos de aglomerados que satisfagan ciertas propiedades deseables.
- **Definición - Grafo de conglomerados.**
  - *Un grafo de aglomerados se denomina un grafo de conglomerados asociado a un grafo no dirigido G si sus aglomerados son los conglomerados de G.*
- Por ejemplo, el grafo de aglomerados mostrado en la figura de la derecha es un grafo de conglomerados asociado al grafo del ejemplo anterior (el grafo de la izquierda).





Grafo de unión

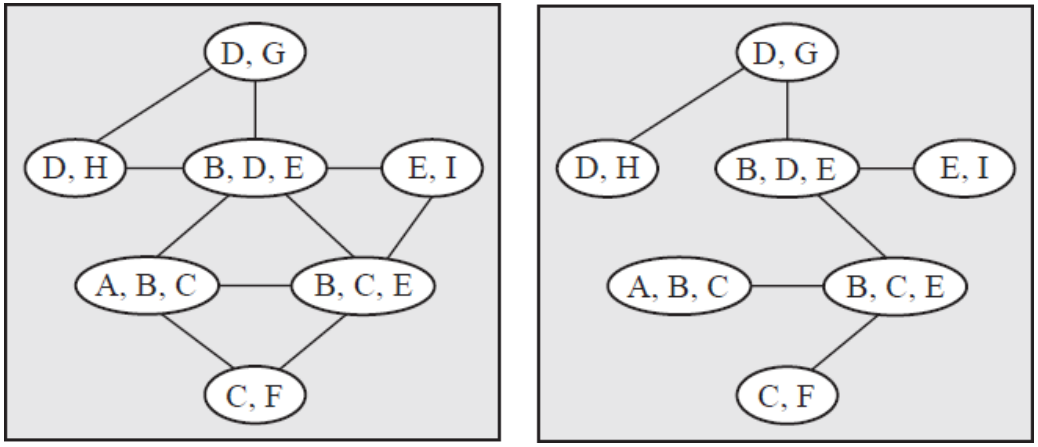
- **Definición - Grafo de unión.**
  - *Un grafo de conglomerados asociado a un grafo no dirigido se denomina un grafo de unión si* ***contiene todos las aristas posibles que unan conglomerados con algún nodo común****.*
- El grafo de unión asociado a un grafo dado es único. Por ejemplo, el grafo de conglomerados mostrado es el grafo de unión asociado al grafo del ejemplo anterior.
- Los grafos de unión tienen la propiedad de que los conglomerados con nodos comunes forman un conjunto completo.
- Por esta razón, **los grafos de unión suelen contener numerosas aristas**. Por tanto, ***sería interesante obtener algún grafo de estructura más simple*** (por ejemplo, un árbol) que retuviese la propiedad de conectar los conglomerados que tengan elementos comunes.

Árbol de unión

- **Definición - Árbol de unión.**
  - *Un grafo de conglomerados se denomina un árbol de unión si es un árbol y* ***todo nodo que pertenezca a dos conglomerados también pertenezca a todos los conglomerados contenidos en el camino que los une***.
- Nótese que en un árbol de unión existe un único camino entre cada par de conglomerados con un nodo común.

Ejemplo Árbol de Unión

- El árbol de conglomerados de la figura de la derecha es un árbol de unión obtenido eliminando cuatro aristas del grafo de unión de la figura de la izquierda.
- Se puede comprobar fácilmente que todos los conglomerados contenidos en el camino que une dos conglomerados contiene también los nodos comunes a éstos. Por ejemplo, los conglomerados *{D,H}* y *{B,D,E}* tienen un nodo común, *D*, que también pertenece al resto de los conglomerados en el camino que los une, *{D,H} − {D,G} − {B,D,E}*.

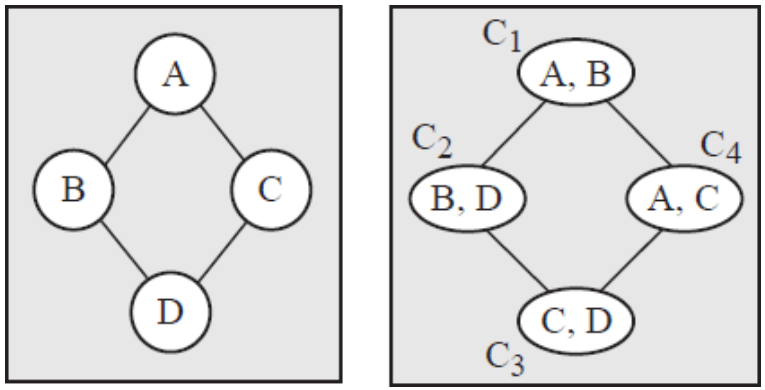


Teorema Árbol de Unión

- Más adelante se analizarán diversos métodos de propagación de incertidumbre que utilizan un árbol de unión para simplificar los **cálculos necesarios para actualizar las probabilidades**.
- El teorema siguiente indica cuándo es posible convertir un grafo de unión en un árbol de unión eliminando algunas de sus aristas (ver Jensen (1988)).
- **Teorema - Árbol de unión.**
  - *Un grafo no dirigido posee un árbol de unión si y sólo si es triangulado.*

Ejemplo
Grafo sin árbol de unión

- La figura muestra un grafo no triangulado y el grafo de unión asociado cuyos conglomerados son $C1 = \{A,B\}$, $C2 = \{B,D\}$, $C3 = \{C,D\}$ y $C4 = \{A,C\}$.
- En esta situación es imposible construir un árbol de unión a partir de este grafo, pues se trata de un grafo no triangulado. Por ejemplo, si se eliminase del grafo la arista $C1 - C4$, el árbol resultante no sería un árbol de unión pues, por ejemplo, el nodo A está contenido en $C1$ y $C4$ pero no, en los dos conglomerados restantes del camino $C1 - C2 - C3 - C4$.



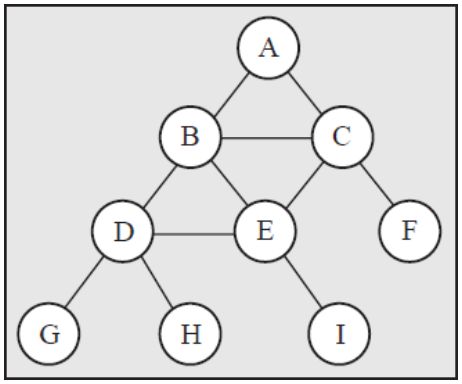
Algoritmo
Generación árbol de unión

- Se introdujo la propiedad de intersección dinámica para conglomerados. Esta propiedad permite **ordenar los conglomerados de un grafo triangulado obteniendo una *cadena de conglomerados***. El algoritmo siguiente permite construir un árbol de unión asociado a un grafo triangulado. Consiste en organizar una cadena de conglomerados en una estructura de árbol.
- **Algoritmo – Generación de un árbol de unión.**
- **Datos:** Un grafo triangulado no dirigido $G = (X,L)$.
- **Resultado:** Un árbol de unión $G' = (C,L')$ asociado a G.

1. *Iniciación:* Utilizar el algoritmo que hemos visto para obtener una cadena de conglomerados del grafo G, $(C1, \ldots, Cm)$.
2. Para cada conglomerado $Ci \in C$, escoger un conglomerado Ck en $\{C1, \ldots, Ci-1\}$ con el máximo número de nodos comunes y añadir la arista $Ci - Ck$ a L' (inicialmente vacía). Los empates se deshacen de forma arbitraria.

Ejemplo Generación de un árbol de unión

- Se aplica el algoritmo para generar un árbol de unión asociado al grafo triangulado dado en la figura. En un ejemplo anterior, se obtuvo la cadena de conglomerados $C1 = \{A,B,C\}$, $C2 = \{B,C,E\}$, $C3 = \{B,D,E\}$, $C4 = \{C, F\}$, $C5 = \{D,G\}$, $C6 = \{D,H\}$ y $C7 = \{E, I\}$.

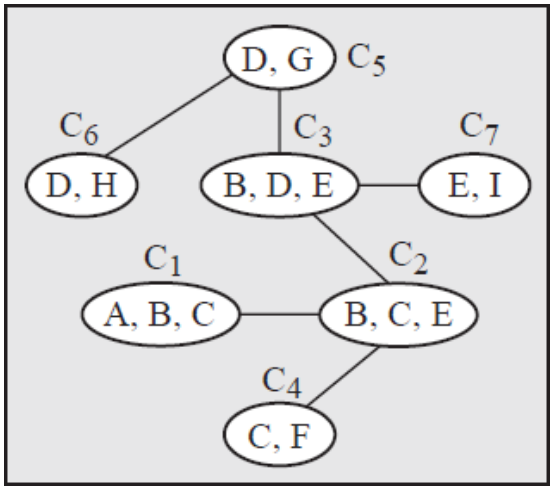


- Para generar un árbol de unión se procede a añadir las aristas necesarias a un conjunto *L'*, inicialmente vacío, de la siguiente forma:
  - Los conglomerados *C2* y *C3* tienen el máximo nº de nodos en común con el conglomerado *C7*. Deshaciendo el empate arbitrariamente, se elige el conglomerado *C3* y se añade la arista $C7 - C3$ a *L'*.

Ejemplo Generación de un árbol de unión

- C3 y C5 tienen el máximo nº de nodos coincidentes con C6. Se elige arbitrariamente uno de ellos, por ejemplo C5 y se añade la arista C6 − C5 a *L'*.
- De entre los conglomerados de {C1, C2, C3, C4}, el conglomerado C3 es el que tiene más elementos en común con C5. Por tanto, se añade la arista C5 − C3 a *L'*.
- Procediendo de forma similar, se añaden las aristas C4 −C2, C3 −C2 y C2 − C1.

- El árbol de unión resultante se muestra en la figura. Dado que muchos empates se deciden de forma arbitraria, el algoritmo podría generar varios árboles de unión distintos para un mismo grafo no dirigido.



Familias de nodos

- Se ha tratado el problema de la ***construcción de grafos de aglomerados asociados a grafos no dirigidos***. Sin embargo, ***este concepto también puede ser aplicado a los grafos dirigidos*** trabajando indirectamente ***con el grafo no dirigido asociado***.
- Como veremos más adelante, las **familias de nodos** en un grafo dirigido juegan un papel importante en los ***mecanismos de propagación de evidencia***.
- Recordemos:
  - **Familia de un nodo:** *conjunto formado por un nodo y sus padres.*
- Existe también un **tipo de redes probabilísticas** que se definen mediante ***funciones locales de probabilidad definidas en las familias de los nodos***. Por tanto, estamos interesados en el desarrollo de grafos de aglomerados tales que todas las familias del grafo dirigido original estén contenidas en, al menos, un aglomerado.

Árbol de familias

- **Definición Árbol de familias.**
  - *Un árbol de familias de un grafo dirigido D, es un árbol de unión de algún grafo no dirigido G asociado a D, en el cual la familia de cada nodo está contenida en al menos un conglomerado.*
- El proceso de **moralización** de un grafo dirigido garantiza que *la familia de cualquier nodo estará contenida en al menos un conglomerado del grafo no dirigido resultante*. Por tanto, aplicando el ***algoritmo de generación de un árbol de unión*** a cualquier versión triangulada del grafo moral se obtendrá un árbol de familias del grafo dirigido original.

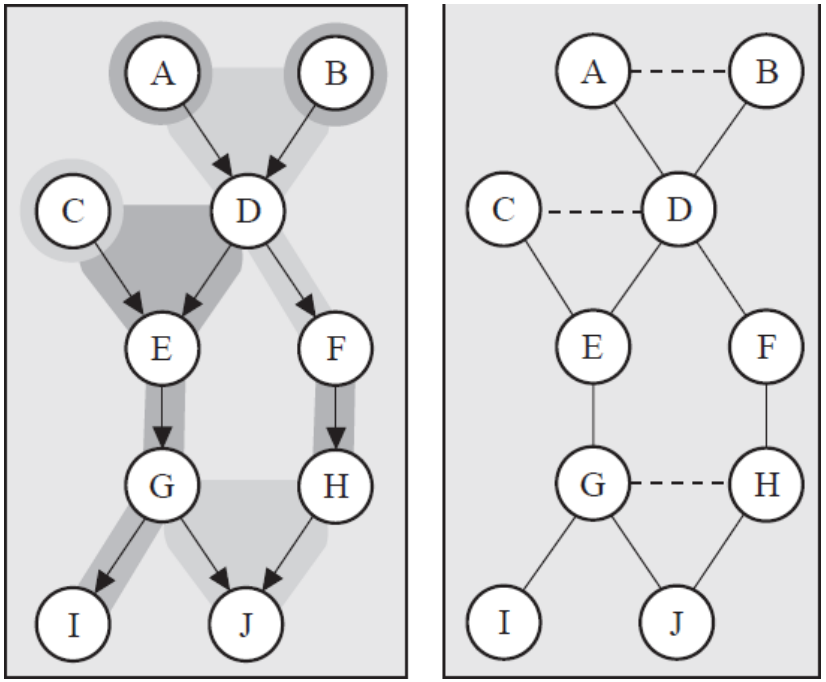
Algoritmo de Generación de un árbol de familias

- **Algoritmo de Generación de un árbol de familias.**
- • **Datos:** Un grafo dirigido $D = (X,L)$.
- • **Resultado:** Un árbol de familias $G' = (C,L')$ asociado a D.

1. **Moralizar** el grafo dirigido.
2. **Triangular** el grafo no dirigido resultante utilizando el algoritmo de triangulación de máxima cardinalidad.
3. Aplicar el algoritmo de **generación de un árbol de unión** para calcular un árbol de unión del grafo resultante.

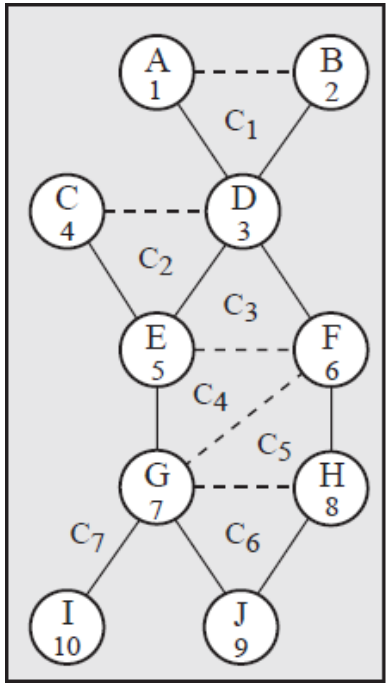
Ejemplo Generación de un árbol de familias

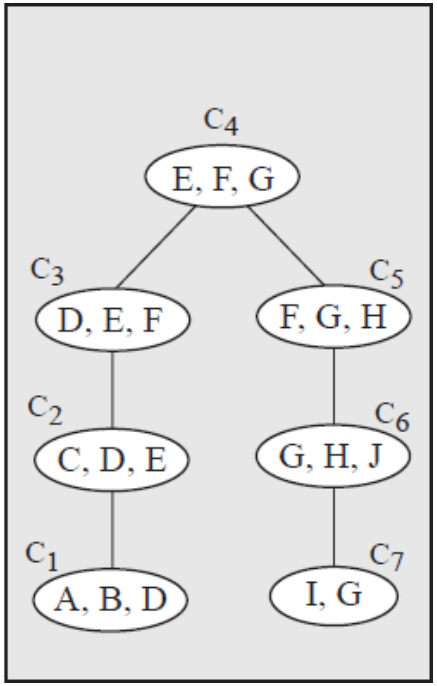
- **Ejemplo Generación de un árbol de familias.**
- Considérese el grafo dirigido de la figura de la izquierda, en la que de marcan las familias de los nodos: *{A}*, *{B}*, *{C}*, *{A,B,D}*, *{C,D,E}*, *{D,F}*, *{E,G}*, *{F,H}*, *{G, I}* y *{G,H, J}*.
- Aplicando el algoritmo se obtiene un árbol de familias asociado a este grafo.
- Para construir el ***grafo moral*** correspondiente, es necesario añadir las tres aristas mostradas con línea discontinua.



Ejemplo Generación de un árbol de familias

- El grafo moral puede triangularse utilizando el algoritmo de triangulación de máxima cardinalidad (ver figura izquierda, con dos nuevas aristas E-F y F-G).
- Finalmente, el árbol de familias asociado puede obtenerse aplicando a este grafo el algoritmo de generación de grafo de unión, como vimos en un ejemplo anterior.
- El árbol de familia resultante se muestra en la figura de la derecha.
- Todas las familias del grafo dirigido original están contenidas en al menos un conglomerado del árbol.





Representación de Grafos

Distintas formas de representación para resaltar ≠ características:

- **Simbólicamente**, como un par **(*X*,*L*)**, donde *X* es un conjunto de variables y *L* es un conjunto de aristas entre pares de variables.
  - Equivalente a la anterior, la representación simbólica dada por **(*X*, *Ady*)**, donde *Ady* es la clase de los conjuntos de adyacencia de los nodos.
- **Gráficamente**, por medio de un diagrama formado por un conjunto de nodos (uno para cada variable) y un conjunto de líneas o flechas (una para cada arista del conjunto *L*).
- **Numéricamente**, utilizando ciertos tipos de matrices.

Representación de Grafos. Ventajas e Inconvenientes

- **Representación simbólica**:
  - **Ventaja**: es conceptualmente simple (cada grafo puede ser representado por un par de conjuntos)
  - **Inconveniente**: no proporciona información directa sobre la topología del grafo.
- **Representación gráfica**:
  - **Ventaja**: permite observar globalmente las distintas relaciones que existen entre las variables
  - **Inconveniente**: se vuelve extremadamente compleja cuando el nº de aristas entre los nodos es muy elevado.
- **Representación numérica**:
  - **Ventaja**: permite obtener características de los grafos por simples manipulaciones algebraicas
  - **Inconveniente**: es muy abstracta.

Representación Gráfica de un Grafo

- Problema: ***representar*** gráficamente ***los nodos y aristas*** de un grafo, por ejemplo en un hoja de papel o en la pantalla de un ordenador.
- Principal obstáculo: puede ser representado ***de muchas formas***.
- Algunas de estas **representaciones** son **mejores que otras**:
  - sencillez
  - capacidad para mostrar características del grafo
  - …
- Estas representaciones permiten analizar visualmente ciertas ***propiedades topológicas*** del grafo de forma sencilla.
- Por ejemplo, el tipo de grafos que pueden ser dibujados en el plano sin que sus aristas se crucen se conocen como ***grafos planos*** y tiene numerosas aplicaciones interesantes.
- En Preparata y Shamos (1985) y Tamassia y Tollis (1995) se describen algunos problemas asociados con la representación de grafos.

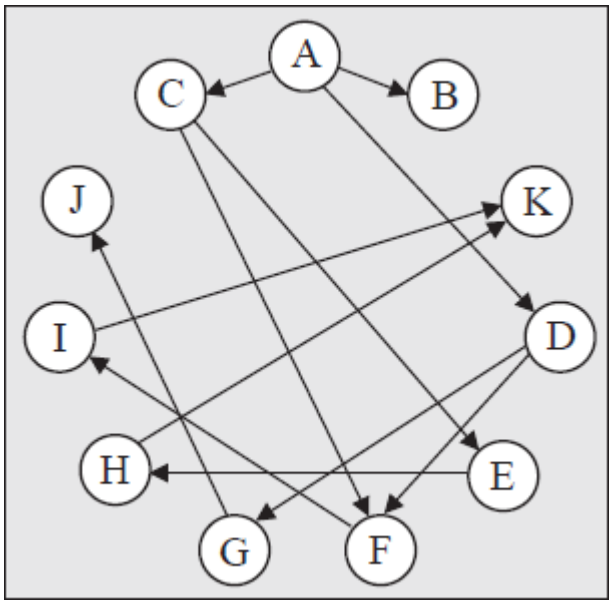
Representaciones Gráficas «buenas»

- Debe cumplir los siguientes requisitos:

1. Puede ser ***construida de forma sencilla y rápida utilizando algún algoritmo***.
2. Las ***características topológicas del grafo podrán ser analizadas mediante la representación gráfica***. Por ejemplo, la diapositiva siguiente muestra dos representaciones distintas del mismo grafo; es más fácil comprobar que el grafo es múltiplemente conexo a partir de la representación de la derecha que a partir de la izquierda. Otro ejemplo fueron las dos representaciones gráficas que vimos para un grafo inconexo, de forma que una de ellas mostraba más fácilmente esta propiedad topológica.
3. La representación será simple, teniendo un ***número mínimo de cortes de aristas***.

Representación Circular de un Grafo

- Una de las formas más ***sencillas*** de representar gráficamente un grafo.
- Consiste en dibujar los nodos sobre una ***circunferencia a distancias iguales***.
- Propiedad importante: garantiza que ***no puede haber más de dos nodos alineados***.
  - → Por tanto, si las aristas del grafo se representan mediante líneas rectas, garantiza que no habrá aristas ocultas entre los nodos.
- Representación óptima para grafos con un gran número de aristas.



Representación Circular de un Grafo. Ventajas e inconvenientes

VENTAJAS:

- Es fácil de construir.
- Todas las aristas son transparentes en el diagrama.
- Es la más conveniente para grafos completos o casi completos.

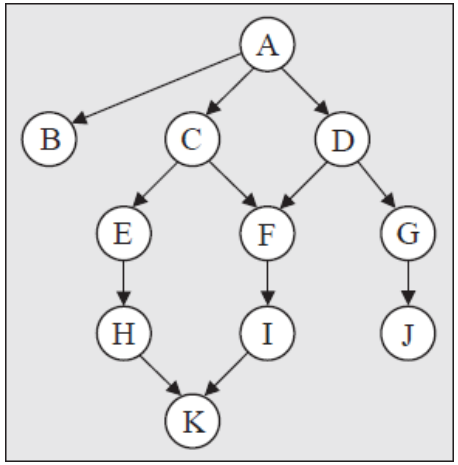
INCONVENIENTES:

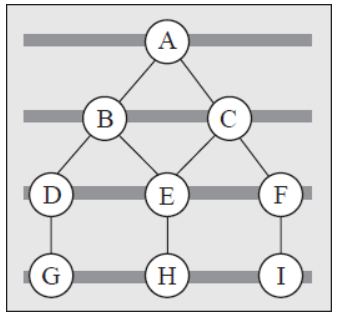
- Pueden existir numerosos cortes entre las aristas, complicando el diagrama.

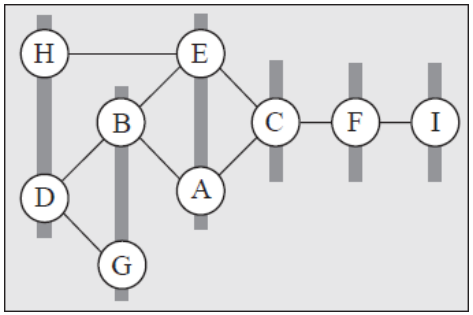
- A continuación vemos otra representación del mismo grafo, sin ningún corte entre las aristas.

Representación Multinivel de un Grafo

- Idea básica: organizar los nodos en distintos niveles, o capas, de forma que:
  - no existan aristas entre nodos del mismo nivel
  - todo nodo en un nivel esté conectado con algún nodo del nivel previo.
- Representación clara del grafo situando los nodos en niveles horizontales o verticales
- Para desarrollar esta idea en detalle, son necesarias algunas definiciones previas.







Representación Multinivel de un Grafo. Definiciones

- **Subconjunto totalmente inconexo.**
  - *Dado un grafo (X,L), un subconjunto de nodos S ⊂ X se denomina totalmente inconexo si no existe ninguna arista entre los nodos de S, es decir, si (Xi,Xj) ∈ L ⇒* $X_i \notin S \text{ o } X_j \notin S$

- **Representación multinivel.**
  - *Una representación multinivel de un grafo no dirigido (X,L) es una partición*

$$X = \bigcup_{k=1}^{m} S_k,$$

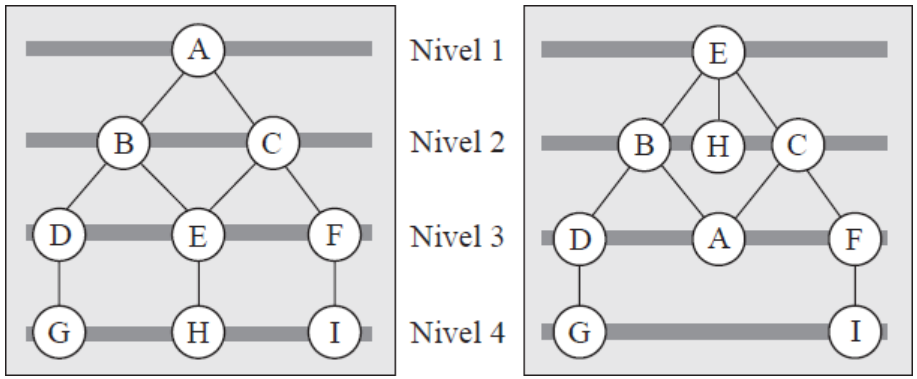
  - *donde los niveles Si, i = 1, . . . , m, son subconjuntos disjuntos y totalmente inconexos de X tales que*

$$\text{si } X_i \in S_k \Rightarrow \exists\, X_j \in S_{k-1} \text{ con } X_i \in Ady(X_j).$$

  - *Es decir, no existen aristas entre nodos del mismo nivel y los nodos de un nivel son adyacentes a, al menos, un nodo del nivel anterior.*

Representación Multinivel de un Grafo

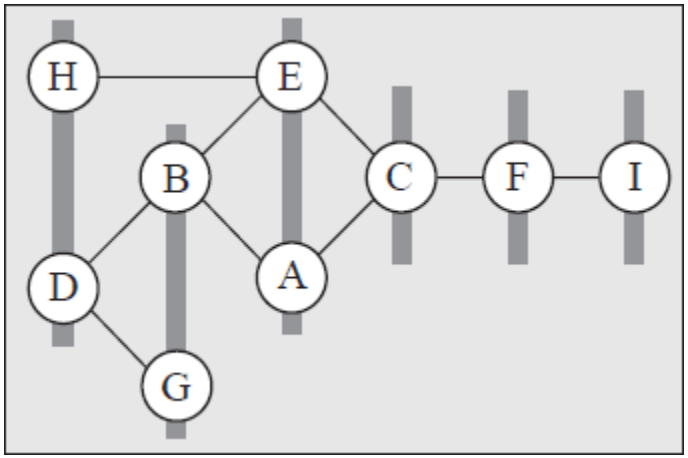
- Nota: los nodos que forman el primer nivel sólo tienen que satisfacer la propiedad de ser un subconjunto totalmente inconexo.
  - → La elección del primer nivel es bastante arbitraria y por tanto un grafo puede tener varias representaciones multinivel.
- Los nodos del primer nivel se denominan ***nodos raíz***.
- Por ejemplo, los grafos de las figuras muestran dos representaciones multinivel del mismo grafo.



Niveles: *{{A}, {B,C}, {D,E,F}, {G,H, I}}* *{{E}, {B,H,C}, {D, A, F}, {G, I}}*

Representación Multinivel de un Grafo

- Las representaciones anteriores empleaban niveles horizontales y partían de un único nodo raíz.
- La siguiente figura muestra una representación diferente, empleando niveles verticales y dos nodos raíz *{D,H}*.



Representación Multinivel de un Grafo. Ventajas

VENTAJAS:

- Es muy conveniente para árboles, o grafos con pocas aristas.
- Muestra la estructura ancestral del grafo a través de los distintos niveles de la representación.

- Siempre es posible obtener una representación multinivel de un grafo no dirigido eligiendo como conjunto de nodos raíz cualquier subconjunto totalmente inconexo de nodos del grafo.
- El segundo nivel de la representación está formado por algunos de los nodos adyacentes a este conjunto de nodos, y así sucesivamente.
- Por ejemplo, en las representaciones horizontales mostradas los únicos nodos raíz son los nodos A y E, respectivamente; mientras que en la representación vertical el conjunto de nodos raíz es *{D,H}*.
- *A continuación lo vemos con un algoritmo*.

Algoritmo Representación multinivel

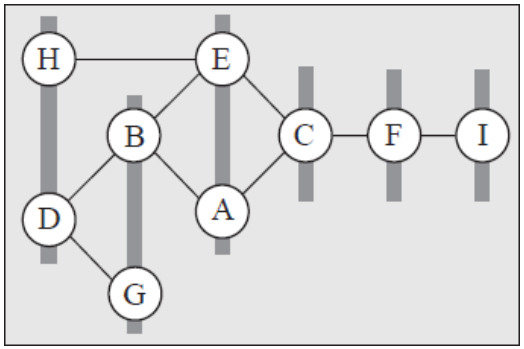
- **Datos:** Un grafo (X, Ady) de n nodos y un conjunto de nodos raíz R.
- **Resultado:** Una representación multinivel S del grafo.

1. *Iniciación:* $Asignados = R$. $Nivel(1) = R$. $Nivel(k) = \Phi$ para $k = 2, \ldots, n$. Tomar $j = 1$.
2. Si $Nivel(j) = \Phi$, devolver $S = \{Nivel(1), \ldots, Nivel(j-1)\}$, que es una representación multinivel del grafo y terminar; en caso contrario, hacer $NivelActual = Nivel(j)$ e ir a la Etapa 3.
3. Seleccionar $Xk \in NivelActual$:
   a) Añadir los elementos de $Ady(Xk) \setminus Asignados$ a $Nivel(j+1)$ y a $Asignados$.
   b) Añadir los elementos de $Ady(Xk) \cap Nivel(j)$ a $Nivel(j+1)$ y eliminar estos elementos de $Nivel(j)$ y de $NivelActual$ e ir a la Etapa 4.
4. Eliminar Xk de $NivelActual$. Si $NivelActual = \Phi$, tomar $j = j+1$ e ir a la Etapa 2; en caso contrario, ir a la Etapa 3.

- La Etapa 3(a) en el algoritmo anterior añade al nivel actual todos los vecinos no asignados de los nodos del nivel previo, mientras que la Etapa 3(b) elimina nodos vecinos del mismo nivel.
- Nótese que si el conjunto de nodos raíz R no fuese totalmente inconexo, el algoritmo eliminaría de forma automática algunos de los nodos de este nivel hasta obtener un subconjunto totalmente inconexo.

Ejemplo
Representación multinivel

- En este ejemplo se aplica el algoritmo para obtener una representación multinivel del grafo dado en la figura de abajo. Este grafo tiene asociados los conjuntos de adyacencia:
  - *Ady(A)={B,C}, Ady(B)={A,D,E}, Ady(C)={A,E,F}, Ady(D)={B,G}, Ady(E)={B,C,H}, Ady(F)={C, I}, Ady(G)={D}, Ady(H)={E}, Ady(I)={F}.*
- Considérese el conjunto de nodos raíz *{D,H}*.



- La diapositiva siguiente muestra la traza del algoritmo. Las filas de la tabla muestran el estado de las variables correspondientes al final de cada etapa.
- Como resultado del algoritmo se obtiene la representación multinivel:
  - Nivel(1)={D,H}, Nivel(2)={B,G}, Nivel(3) ={A,E}, Nivel(4)={C}, Nivel(5) ={F}, Nivel(6)={I}.

Ejemplo
Representación multinivel

Etapa	X_k	j	$Nivel(j)$	$Nivel(j+1)$	$NivelActual$
1	–	1	$\{D,H\}$	ϕ	ϕ
2	–	1	$\{D,H\}$	ϕ	$\{D,H\}$
3(a)	D	1	$\{D,H\}$	$\{B,G\}$	$\{D,H\}$
3(b)	D	1	$\{D,H\}$	$\{B,G\}$	$\{D,H\}$
4	D	1	$\{D,H\}$	$\{B,G\}$	$\{H\}$
3(a)	H	1	$\{D,H\}$	$\{B,G,E\}$	$\{H\}$
3(b)	H	1	$\{D,H\}$	$\{B,G,E\}$	$\{H\}$
4	H	2	$\{B,G,E\}$	ϕ	ϕ
2	H	2	$\{B,G,E\}$	ϕ	$\{B,G,E\}$
3(a)	B	2	$\{B,G,E\}$	$\{A\}$	$\{B,G,E\}$
3(b)	B	2	$\{B,G\}$	$\{A,E\}$	$\{B,G\}$
4	B	2	$\{B,G\}$	$\{A,E\}$	$\{G\}$
3(a)	G	2	$\{B,G\}$	$\{A,E\}$	$\{G\}$
3(b)	G	2	$\{B,G\}$	$\{A,E\}$	$\{G\}$
4	G	3	$\{A,E\}$	ϕ	ϕ
2	G	3	$\{A,E\}$	ϕ	$\{A,E\}$
3(a)	A	3	$\{A,E\}$	$\{C\}$	$\{A,E\}$
3(b)	A	3	$\{A,E\}$	$\{C\}$	$\{A,E\}$
4	A	3	$\{A,E\}$	$\{C\}$	$\{E\}$
3(a)	E	3	$\{A,E\}$	$\{C\}$	$\{E\}$
3(b)	E	3	$\{A,E\}$	$\{C\}$	$\{E\}$
4	E	4	$\{C\}$	ϕ	ϕ
2	E	4	$\{C\}$	ϕ	$\{C\}$
3(a)	C	4	$\{C\}$	$\{F\}$	$\{C\}$
3(b)	C	4	$\{C\}$	$\{F\}$	$\{C\}$
4	C	5	$\{F\}$	ϕ	ϕ
2	C	5	$\{F\}$	ϕ	$\{F\}$
3(a)	F	5	$\{F\}$	$\{I\}$	$\{F\}$
3(b)	F	5	$\{F\}$	$\{I\}$	$\{F\}$
4	F	6	$\{I\}$	ϕ	ϕ
2	F	6	$\{I\}$	ϕ	$\{I\}$
3(a)	I	6	$\{I\}$	ϕ	$\{I\}$
3(b)	I	6	$\{I\}$	ϕ	$\{I\}$
4	I	7	$\{I\}$	ϕ	ϕ

Representación Multinivel Dirigida. Definiciones

- **Representación multinivel.**
- *Una representación multinivel de un grafo dirigido (X,L) es una partición*

$$X = \bigcup_{k=1}^{m} S_k,$$

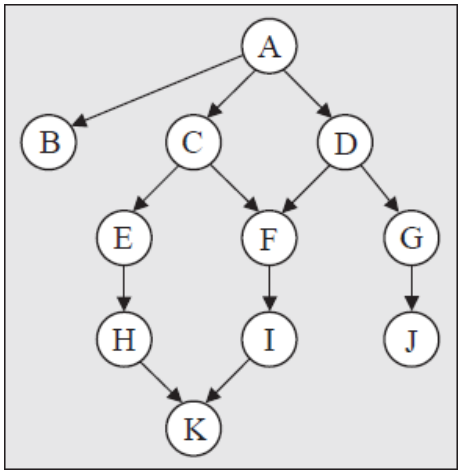
  - *donde los niveles Si, i = 1, . . . , m, son subconjuntos disjuntos y totalmente inconexos de X tales que*

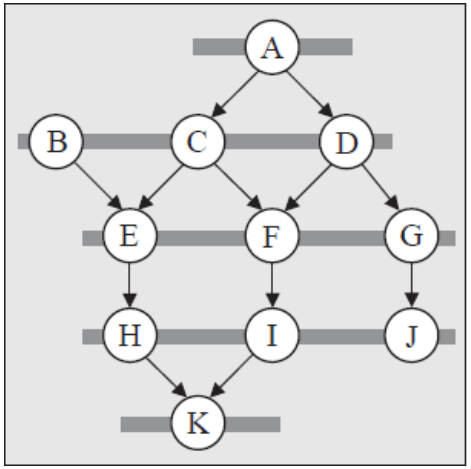
$$X_i \in S_k \text{ y } (X_i, X_j) \in L \Rightarrow X_j \in S_r \text{ con } r > k,$$

- *es decir, todos los padres de un nodo dado han de estar en niveles anteriores al nivel del nodo..*

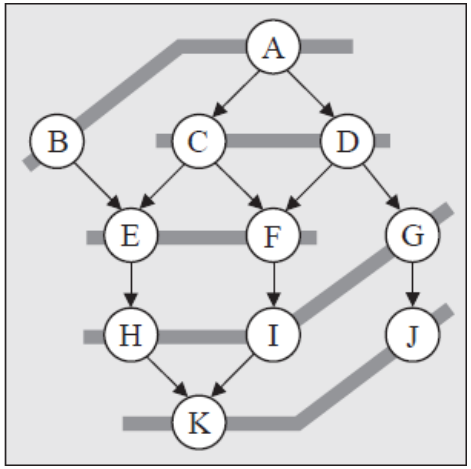
Representación Multinivel Dirigida

- Por ejemplo, la figura de abajo muestra dos representaciones multinivel distintas del grafo de la derecha.
- Los niveles (subconjuntos *Sk*, *k* = 1, . . . , 5) se distinguen en la figura mediante sombras.





$\{\{A\},\{B,C,D\},\{E,F,G\},\{H,I,J\},\{K\}\}$



$\{\{A,B\},\{C,D\},\{E,F\},\{H,I,G\},\{K,J\}\}$

- Es fácil ver que ambos cumplen la definición anteriormente dada.

Representación Multinivel Dirigida. Ventaja

VENTAJA:

- Los diagramas resultantes son fáciles de interpretar, ya que todas las aristas están orientadas en la misma dirección.

- La figura anterior muestra un diagrama multinivel en el que todas las aristas están orientadas en la dirección arriba -> abajo.
- A diferencia de los grafos no dirigidos, no todos los grafos dirigidos admiten una representación multinivel. El siguiente teorema caracteriza la clase de grafos dirigidos que admiten una representación multinivel dirigida.
- **Teorema Representación multinivel dirigida.**
  - *Un grafo dirigido (X,L) admite una representación multinivel dirigida si y sólo si (X,L) es un grafo dirigido acíclico.*
- Utilizando el conjunto de nodos sin padres $\{Xi \mid \Pi_{Xi} = \Phi\}$ de un grafo dirigido acíclico como conjunto de nodos raíz *R*, el algoritmo de representación multinivel que vimos calculará una representación multinivel del grafo, como la dada por la definición.

Representación Multinivel Dirigida. Numeración ancestral

- El siguiente teorema muestra una relación interesante entre representaciones multinivel dirigidas y numeraciones ancestrales.
- **Teorema Numeración ancestral.**
  - *Si {S1, . . . , Sm} es una representación multinivel de un grafo dirigido acíclico (X,L), entonces cualquier numeración de los nodos a que satisfaga a(Xi) > a(Xj) para Xi ∈ Sk y Xj ∈ Sr con k > r es una numeración ancestral de los nodos.*
- Por tanto, el algoritmo anterior permite obtener una numeración ancestral para un grafo dirigido acíclico.
- Además, este tipo de grafos es el único que posee este tipo de numeración.
- A continuación se desarrolla un algoritmo para dividir cualquier grafo dirigido acíclico de la forma mostrada en la definición. Para ello, se necesitan algunas definiciones adicionales.

Representación Multinivel Dirigida. Definiciones adicionales

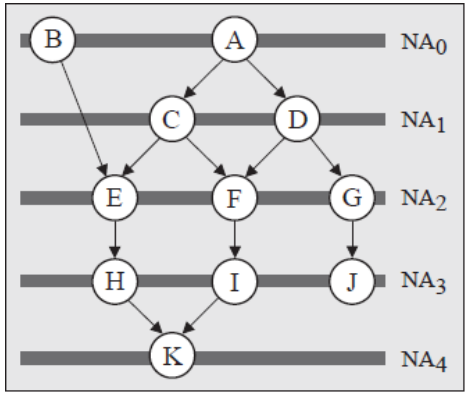
- **Definición Profundidad ascendente.**
  - *La profundidad ascendente de un nodo Xi en un grafo dirigido acíclico, PA(Xi), es la longitud máxima de los caminos del grafo que terminan en el nodo Xi.*
- **Definición Profundidad descendente.**
  - *La profundidad descendente de un nodo Xi en un grafo dirigido acíclico, PD(Xi), es la longitud máxima de los caminos del grafo que comienzan en el nodo Xi.*
- Para calcular la profundidad ascendente de un nodo basta con conocer la profundidad ascendente de sus padres.
- De la misma forma, la profundidad descendente de un nodo está determinada por la profundidad descendente de sus hijos. La profundidad descendente crece siguiendo el sentido de orientación de las aristas, mientras que la profundidad ascendente crece en el sentido contrario.
- Los conceptos de profundidad descendente y ascendente satisfacen las siguientes propiedades:
  - $0 \leq PA(Xi) \leq n - 1$ y $0 \leq PD(Xi) \leq n - 1$, donde n es el nº de nodos.
  - Si Xi no tiene padres, entonces $PA(Xi) = 0$.
  - Si Xi no tiene hijos, entonces $PD(Xi) = 0$.

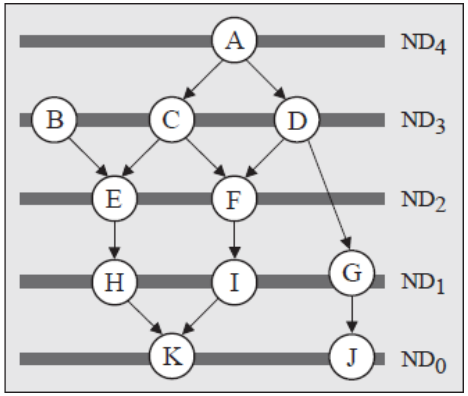
Representación Multinivel Dirigida. Definiciones adicionales

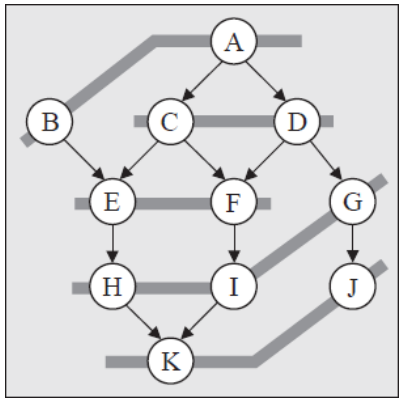
- Un grafo puede ser dividido en niveles calculando la profundidad de todos los nodos del grafo (cada nivel estará formado por los nodos con la misma profundidad).
- **Definición Niveles de profundidad de un grafo.**
  - *Dado un grafo dirigido, el k-ésimo nivel de profundidad ascendente, NAk, es el subconjunto de nodos* $\{Xi \in X \mid PA(Xi) = k\}$. *De forma similar, el k-ésimo nivel de profundidad descendente, NDk, es el subconjunto* $\{Xi \in X \mid PD(Xi) = k\}$.
- El nº de niveles no vacíos de un grafo puede ser calculado en función de la longitud máxima de sus caminos. Sea *m* la longitud del camino más largo del grafo, entonces $Nk = \Phi, \ \forall k > m$ y $Nk \neq \Phi, \ \forall k \leq m$.
- Así, se tiene un nº finito de niveles que definen una partición del grafo como la indicada en la definición de representación multinivel dirigida.
- Este resultado es confirmado por el teorema de la diapositiva siguiente.

Representación Multinivel Dirigida. Definiciones adicionales

- **Teorema Niveles de profundidad y representación multinivel dirigida.**
  - *Para cualquier grafo dirigido acíclico, los conjuntos {NDk : k = 0, . . . , m} y {NAk : k = 0, . . . , m} son dos representaciones multinivel dirigidas que cumplen la definición correspondiente.*
- La figura de abajo muestra los niveles de profundidad ascendente y descendente asociados al grafo dirigido de la figura de la derecha.







Se introduce un algoritmo para calcular los niveles de profundidad de un grafo dirigido. Permite comprobar si un grafo es acíclico.

Se ilustra para el caso de la profundidad ascendente. Con una pequeña modificación el algoritmo obtendría profundidades descendentes.

Algoritmo Profundidad ascendente para grafos dirigidos.

- **Algoritmo Profundidad ascendente para grafos dirigidos**
  - **Datos:** Un grafo (*X, Ady*) de *n* nodos.
  - **Resultado:** Los niveles de profundidad ascendente $\{NA_0, \ldots, NA_k\}$ del grafo.

1. *Iniciación:* Definir $PA(X_i) = 0$ para todos los nodos X_i que no posean padres. Si todos los nodos del grafo tienen algún padre, el algoritmo finaliza pues el grafo contiene algún ciclo. En caso contrario, tomar *profundidad* = 1 y continuar con la Etapa 2.
2. Si *profundidad* ≤ *n*, ir a la Etapa 3; en caso contrario, el algoritmo finaliza (el grafo contiene ciclos).
3. Seleccionar un nodo X_i con $PA(X_i) = profundidad - 1$. Asignar a todos los nodos X_j adyacentes a X_i la profundidad $PA(X_j) = profundidad$. Repetir este proceso con todos los nodos en el nivel *profundidad* − 1, e ir a la Etapa 4.
4. Si ningún nodo tiene profundidad *profundidad,* entonces el algoritmo finaliza y las profundidades de todos los nodos han sido calculadas. En caso contrario, incrementar *profundidad* en una unidad y volver a la Etapa 2.

- Se muestra el pseudocódigo en diapositiva siguiente. Si el algoritmo no finaliza antes de la Etapa *n*, o finaliza en la inicial, el grafo contiene algún ciclo. En caso contrario, es acíclico, y el algoritmo obtiene las profundidades de los nodos. El grafo contiene tantos niveles como etapas realizadas.

Algoritmo Profundidad ascendente para grafos dirigidos.

Algoritmo de Niveles de Profundidad Ascendente

Datos: Un grafo dirigido (X, Ady) de n nodos.

Resultado: Los niveles de profundidad ascendente asociados.

Etapa Inicial:

para $k = 1$ **hasta** n **hacer**
  si nodo X_k no tiene padres **entonces** $PA(X_k) \leftarrow 0$
si $PA(X_k) \neq 0,\ k = 1, \ldots, n$
  Terminar. El grafo contiene algún ciclo.

Etapa de Iteración i**-ésima:**

si $i > n$ **entonces**
  Terminar. El grafo contiene algún ciclo.
en otro caso
  para todo X_k tal que $PA(X_k) = i - 1$ **hacer**
    $PA(X_r) \leftarrow i$, para todo $X_r \in Ady(X_k)$
  si $PA(X_k) \neq i,\ k = 1, \ldots, n$
    Terminar. Todas las profundiades calculadas.
  en otro caso
    ir a la etapa iterativa $i + 1$-ésima

Representación Numérica de Grafos

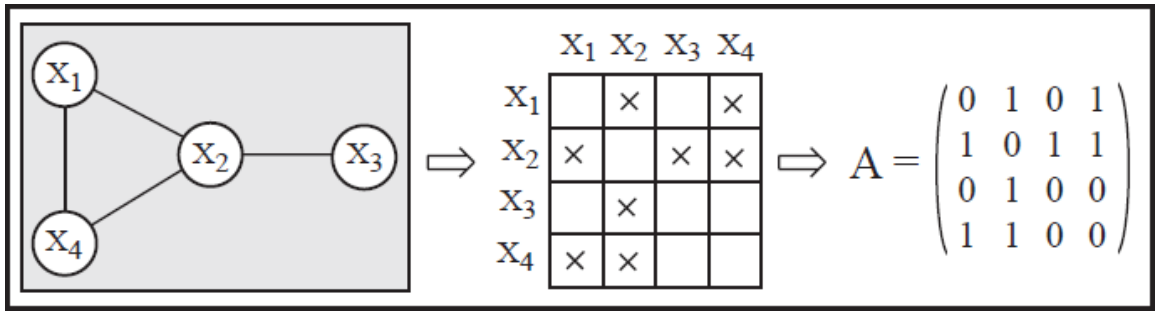
- Un grafo también puede ser representado numéricamente utilizando determinados tipos de matrices.
- La siguiente representación permite calcular de forma sencilla diversas características topológicas de un grafo.
- **Definición Matriz de adyacencia.**
  - *Sea G = (X,L) un grafo de n nodos y sea A = (aij) una matriz n × n, donde*

$$a_{ij} = \begin{cases} 1, & si\ L_{ij} \in L, \\ 0, & en\ caso\ contrario. \end{cases}$$

  - *La matriz A se denomina matriz de adyacencia del grafo G.*

Matriz de adyacencia

- Mediante sencillas manipulaciones algebraicas de la matriz de adyacencia se pueden obtener algunas características del grafo como, por ejemplo, el nº de caminos distintos que unen dos nodos, comprobar si el grafo es conexo, etc.
- La figura muestra el proceso de construcción de la matriz de adyacencia de un grafo dado. Cuando *aij* = 0, entonces no existe ninguna arista del nodo *Xi* al nodo *Xj* . En cambio, *aij* = 1 indica que el nodo *Xi* est´a conectado al nodo *Xj* , o que los nodos son adyacentes, de ah´ı el nombre de esta matriz.



- La matriz *A* contiene toda la información topológica del grafo asociado; por tanto, esta matriz caracteriza al grafo.

Propiedades de la matriz de adyacencia

- Notar que:
  - La matriz de adyacencia de un grafo no dirigido es simétrica.
  - Dado que *Lii* $\notin$ *L* para todos los valores de *i*, los elementos diagonales de *A* son nulos.
  - La matriz de adyacencia de un grafo no dirigido completo debe contener un uno en todos los elementos no diagonales.
- La matriz de adyacencia permite comprobar si existe algún camino entre cada para de nodos.
- También puede calcularse la longitud de todos los caminos que unan cada par de nodos.
- El teorema de la diapositiva siguiente muestra cómo se puede utilizar la matriz de adyacencia para esta tarea.

Teorema Potencias de la matriz de adyacencia

- **Teorema Potencias de la matriz de adyacencia.**
  - *Sea A^r la r-ésima potencia de la matriz de adyacencia asociada con el grafo G = (X,L). Entonces, el ij-ésimo elemento de A^r da el número de caminos de longitud r del nodo Xi al nodo Xj.*
- **Demostración:**
  - La demostración de este teorema puede ser fácilmente obtenida, por inducción, de la forma siguiente: El teorema se cumple para *r* = 1, ya que *aij* = 1 si existe un camino de longitud 1 (una arista) entre los nodos *i* y *j*, y *aij* = 0 en caso contrario.
  - Suponiendo que el resultado es cierto para *Ar*, para *Ar*+1 se tiene que

$$A^{r+1} = A^r A \Leftrightarrow a_{ij}^{r+1} = \sum_{k=1}^{n} a_{ik}^r a_{kj}$$

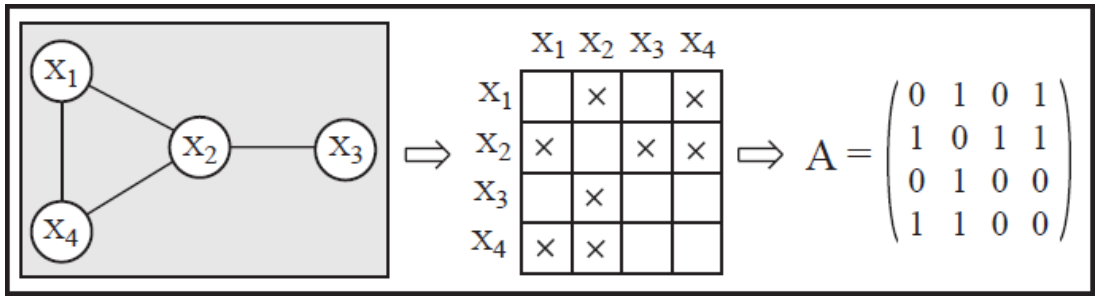
  - es decir, si hay a^r_{ik} caminos de longitud *r* del nodo *Xi* al nodo *Xk* y existe una arista del nodo *Xk* al nodo *Xj* (a_{kj} = 1), entonces se tienen a^r_{ik} caminos de longitud (*r* + 1).

Teorema Potencias de la matriz de adyacencia. Consecuencia

- El teorema anterior implica:
  - El elemento *ij*-ésimo de A^r es cero si y sólo si no existe ningún camino de longitud *r* de *Xi* a *Xj* .
  - Calculando las potencias sucesivas de la matriz de adyacencia de un grafo dado $A, A^2, A^3, \ldots$, se pueden calcular directamente el número de caminos de longitud 1, 2, 3, . . . que unen cada par de nodos.

Ejemplo Potencias de la matriz de adyacencia

- Dado el grafo de la figura:



- Las primeras tres potencias de la matriz de adyacencia son:

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

- de las cuales puede deducirse que, por ejemplo, sólo existe un camino de longitud 3 del nodo *X*1 al nodo *X*3 ($a^3_{13} = 1$). La figura muestra este camino, *X*1 − *X*4 − *X*2 − *X*3.

Matriz de alcanzabilidad

- La matriz de adyacencia también puede ser utilizada para comprobar si un grafo es conexo o inconexo. Para ello, se introduce la siguiente matriz asociada a un grafo.
- **Definición 4.48 Matriz de alcanzabilidad.**
  - *La matriz de alcanzabilidad, T = (tij), de un grafo G se define como*

$$t_{ij} = \begin{cases} 1, & si\ existe\ algun\ camino\ del\ nodo\ X_i\ al\ nodo\ X_j \\ 0, & en\ caso\ contrario. \end{cases}$$

- La matriz de alcanzabilidad está claramente relacionada con las potencias de la matriz de adyacencia.
- El siguiente resultado da una cota del número máximo de potencias de esta matriz que es necesario conocer para poder calcular la matriz de alcanzabilidad.

Teorema Acotación a la longitud de un camino

- **Teorema Acotación a la longitud de un camino.**
  - *Dado un grafo con n nodos, si existe un camino del nodo Xi al nodo Xj, entonces también existe un camino de longitud menor que n de Xi a Xj.*
- Como consecuencia, la matriz de alcanzabilidad puede ser obtenida a partir de un número finito de potencias de la matriz de adyacencia, A, A^2, A^3, . . . , A^{n-1}. El número de potencias necesario es $n - 1$. De hecho, se tiene

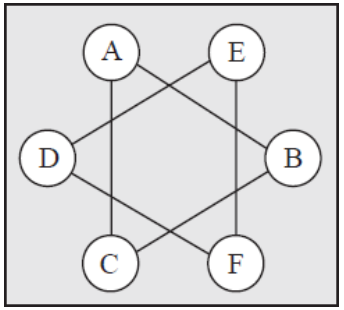
$$t_{ij} = \begin{cases} 0, & si\ a_{ij}^k = 0, \forall\, k < n \\ 1, & en\ caso\ contrario. \end{cases}$$

- En un grafo conexo, todos los elementos de la matriz de alcanzabilidad han de ser distintos de cero. Por tanto, la propiedad de conexión de un grafo se puede analizar a través de su matriz de alcanzabilidad. Además, en caso de que el grafo no sea conexo, la estructura de esta matriz permite identificar las componentes conexas del grafo.

Ejemplo
Matriz de alcanzabilidad

- Dado el grafo de la figura, es posible calcular su matriz de alcanzabilidad obteniendo las primeras $n = 5$ potencias de su matriz de adyacencia. La matriz de adyacencia de este grafo es

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



- Calculando las cinco primeras potencias, se obtiene la matriz de alcanzabilidad asociada utilizando el teorema anterior:

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

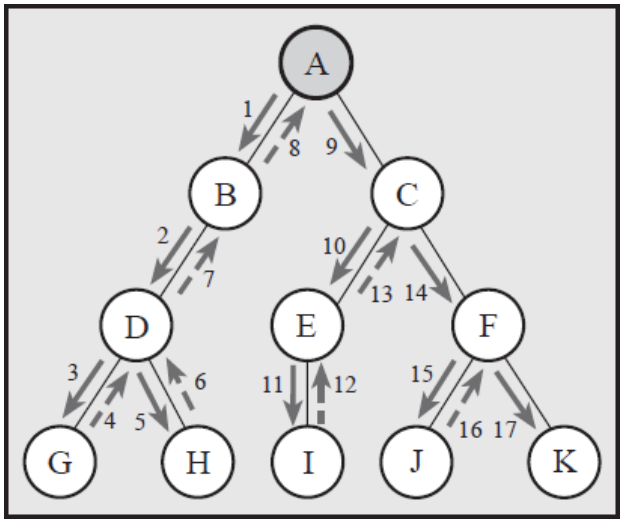
- A partir de la estructura de *T*, pueden distinguirse dos componentes conexas: *{X1,X2,X3}* y *{X4,X5,X6}*. Esta conclusión podría ser difícil de obtener con una reprentación gráfica. → En grafos complejos, las matrices de adyacencia y alcanzabilidad son herramientas útiles para investigar la estructura topológica del grafo.

Algunos Algoritmos para Grafos

- En las secciones anteriores se han introducido varias propiedades y conceptos para analizar las características de los grafos. En esta sección se introducen algunos algoritmos útiles para comprobar si un grafo posee alguna de esas propiedades. Más concretamente, dado un grafo, estamos interesados en:
  1. Obtener un camino entre dos nodos.
  2. Comprobar si el grafo es conexo y hallar sus componentes conexas.
  3. Identificar si el grafo contiene bucles o ciclos.
- No se pretende mostrar los algoritmos óptimos para resolver cada problema, sino mostrar, con la ayuda de ejemplos ilustrativos, las ideas básicas que subyacen a estos métodos.
- *Los lectores interesados en el diseño de algoritmos eficientes pueden consultar los libros de Cormen, Leiserson y Rivest (1990) y Golumbic (1980). Una descripción m´as detallada de algoritmos para grafos puede encontrarse en Gibbons (1985), McHugh (1990) y Skiena (1990).*

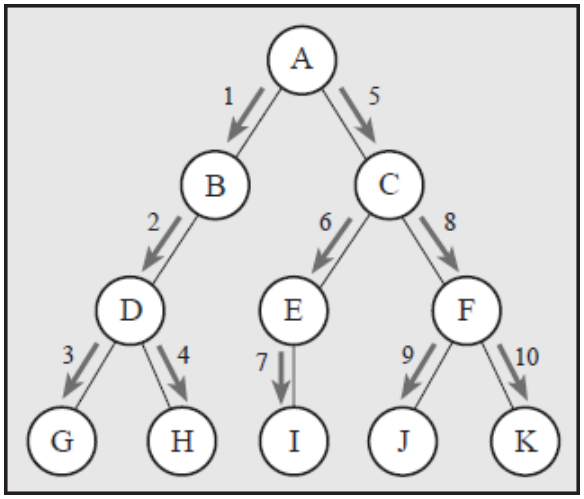
Métodos de Búsqueda

- Muchos algoritmos para grafos necesitan un mecanismo de búsqueda para explorar los nodos y aristas de un grafo. Por ejemplo, los algoritmos de búsqueda pueden ser utilizados para obtener un camino entre dos nodos, o para buscar un bucle o ciclo en un grafo. Estos métodos son la base para la construcción de los algoritmos.
- La exploración de un grafo comienza en un nodo inicial y consiste en la definición de un criterio para moverse hacia adelante y hacia atrás a través de las aristas del grafo, pasando de un nodo a un nodo vecino en cada etapa. Por tanto, la diferencia entre los distintos métodos de búsqueda radica en el criterio elegido para moverse de un nodo a otro.
- La figura muestra una búsqueda exhaustiva de un grafo comenzando en el nodo *A*. Siguiendo la secuencia indicada, y pasando de un nodo a un nodo vecino en cada etapa, se pueden visitar todos los nodos del grafo en un orden predeterminado: *A*, *B*, *D*, *G*, *H*, *C*, *E*, *I*, *F*, *J* y *K*. Además, cualquier arista es recorrida como máximo dos veces: una en la dirección de avance (líneas continuas) para alcanzar nuevos nodos y una en la dirección de retroceso (líneas discontinuas) volviendo hacia atrás, a algún nodo ya visitado.



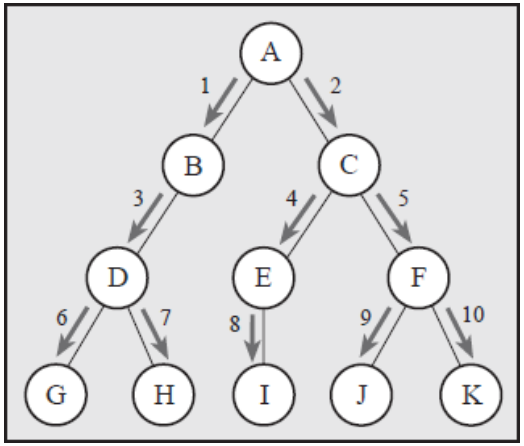
Método de búsqueda en profundidad

- En cada etapa del método de búsqueda en profundidad se visita alguno de los vecinos no visitados del nodo actual (ver la figura, donde los números indican el orden en que se visitan los nodos). En caso de que el nodo actual no tenga ningún vecino no visitado, el algoritmo vuelve atrás al nodo visitado anteriormente y el proceso de búsqueda continua hasta que todos los nodos han sido visitados.



Método de búsqueda en anchura

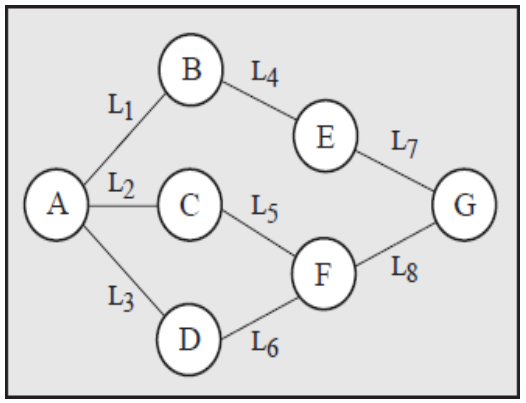
- El método de búsqueda en anchura visita los nodos del grafo capa a capa, comenzando en un nodo inicial y visitando, en la primera etapa todos los vecinos del nodo inicial. Después, se selecciona alguno de estos vecinos como nuevo nodo y se repite el proceso (ver figura, donde los números indican el orden en que se visitan los nodos).



- En las secciones siguientes se desarrollan varios algoritmos basados en estos métodos.

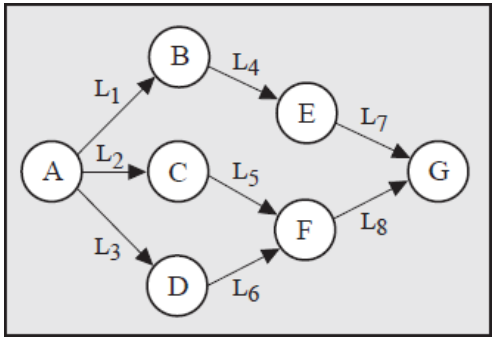
Algoritmos de Búsqueda de Caminos

- Dado un grafo $G = (X,L)$:
  - **Encontrar un camino del nodo *Xi* al nodo *Xj***.
- Se introducen dos algoritmos de búsqueda de caminos basados en las dos estrategias anteriores.
- La representación más conveniente y eficiente es la que emplea conjuntos de adyacencia.
- El grafo de la figura puede ser representado por (X,L), donde:
  - X es el conjunto de nodos $\{A,B,C,D,E,F,G\}$ y
  - L es el conjunto de aristas $\{L1, \ldots, L8\}$.
- Desde un punto de vista computacional, es más adecuada la representación:
  - $Ady(A)=\{B,C,D\}$, $Ady(B)=\{A,E\}$, $Ady(C)=\{A,F\}$, $Ady(D)=\{A,F\}$, $Ady(E)=\{B,G\}$, $Ady(F)=\{C,D,G\}$, $Ady(G)=\{E,F\}$.



Algoritmos de Búsqueda de Caminos

- El grafo dirigido de la figura tiene los conjuntos siguientes de adyacencia:
  - *Ady(A)={B,C,D}, Ady(B)={E}, Ady(C)={F}, Ady(D)={F}, Ady(E)={G}, Ady(F)={G}, Ady(G)=Φ.*



- Los conjuntos de adyacencia proporcionan una representación independiente del carácter dirigido o no dirigido del grafo.
  - Por ejemplo, si nos diesen el grafo dirigido de la figura y se quisiese realizar alguna operación de carácter no dirigido (obtener bucles, caminos no dirigidos, etc.), bastaría con considerar los conjuntos de adyacencia correspondientes al grafo no dirigido asociado.
- A partir de las técnicas de búsqueda anteriores, se pueden definir de forma sencilla los siguientes algoritmos de búsqueda de caminos: ***búsqueda de caminos en profundidad*** y ***búsqueda de caminos en anchura***.

Algoritmo Búsqueda de caminos en profundidad

- **Datos:** Un grafo arbitrario (*X, Ady*) y dos nodos *Xi* y *Xj* .
- **Resultado:** Un camino *Camino* = {*Xi*1,. . . ,*Xir*} del nodo *Xi* = *Xi*1 al nodo *Xj* = *Xir* . Si no existe tal camino, entonces *Camino* = *Φ*.

1. *Iniciación:* Tomar *Xk* = *Xi*, *Camino* = {*Xi*} y *Visitados* = {*Xi*}.
2. *Iteración:* Si todos los nodos de *Ady*(*Xk*) han sido ya visitados, o si *Ady*(*Xk*) = *Φ*, ir a la Etapa 4; en caso contrario, ir a la Etapa 3.
3. *Etapa de avance:* Elegir un nodo *Xr* $\in$ *Ady*(*Xk*), tal que *Xr no* $\notin$ *Visitados*, y añadir *Xr* a *Camino* y a *Visitados*. Si *Xr* = *Xj*, entonces el algoritmo finaliza con resultado *Camino*; **en caso contrario, tomar *Xk* = *Xr* e ir a la Etapa 2**.
4. *Etapa de retroceso:* Si *Xk* = *Xi*, el algoritmo finaliza pues no hay ningún camino de *Xi* a *Xj*; en caso contrario, **eliminar *Xk* de *Camino*, asignar a *Xk* el último nodo de *Camino***, e ir a la Etapa 2.

Algoritmo Búsqueda de caminos en profundidad

- Se muestra el pseudocódigo para este algoritmo.
- En cada etapa del algoritmo se actualizan las listas:
  - ***Camino***, que contiene un camino de *Xi* al último nodo visitado.
  - ***Visitados***, que contiene los nodos que ya han sido visitados.

Algoritmo de Búsqueda de Caminos en Profundidad

Datos: Un grafo (X, Ady) y dos nodos X_i y X_j.
Resultado: Un camino de X_i a X_j, o ϕ si no existe ningún camino.

Etapa Inicial:

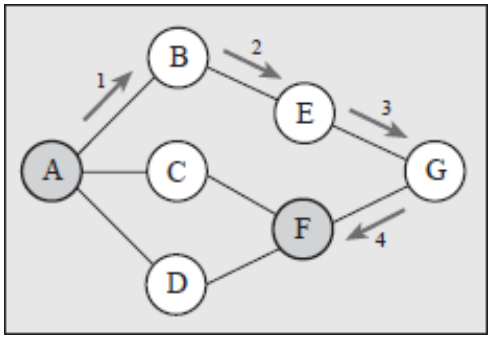
$X_k \leftarrow X_i$
$Camino \leftarrow \{X_i\}$
$Visitados \leftarrow \{X_i\}$

Etapa de Iteración:

si existe $X_r \in Ady(X_k) \setminus Visitados$, **entonces**
    añadir X_r a $Visitados$ y a $Camino$
    si $X_r = X_j$, **entonces**
        Terminar. Se ha hallado un camino.
    en otro caso
        Tomar $X_k \leftarrow X_r$
        repetir la etapa de iteración.
en otro caso
    si $X_k = X_i$, **entonces**
        Terminar. No existe camino entre los nodos.
    en otro caso
        eliminar X_k de $Camino$
        $X_k \leftarrow$ último nodo en $Camino$
        repetir la etapa de iteración.

Ejemplo Búsqueda de caminos en profundidad

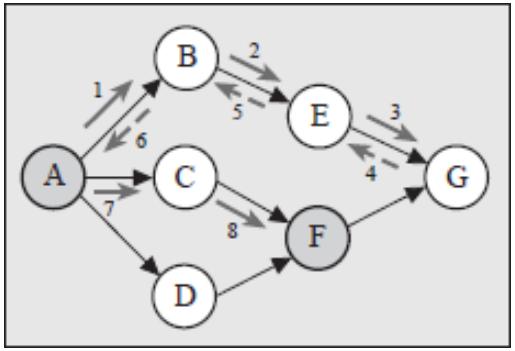
- Dado el grafo no dirigido de la figura de, se desea obtener un camino entre los nodos *A* y *F*.
- En la tabla se recoge el resultado de aplicar el algoritmo anterior a este grafo.
  - Muestra los valores del nodo actual *Xk*, sus nodos adyacentes aún no visitados *Ady(Xk)* \ *Visitados*, y las listas *Camino* y *Visitados* al final de la etapa indicada.
- En este caso no se realiza ninguna etapa de retroceso para obtener el camino *A−B−E−G−F*.



Etapas	X_k	$Ady(X_k) \setminus Visitados$	Visitados	Camino
1	A	$\{B, C, D\}$	$\{A\}$	$\{A\}$
2,3	B	$\{E\}$	$\{A, B\}$	$\{A, B\}$
2,3	E	$\{G\}$	$\{A, B, E\}$	$\{A, B, E\}$
2,3	G	$\{F\}$	$\{A, B, E, G\}$	$\{A, B, E, G\}$
2,3	F	$\{C, D, G\}$	$\{A, B, E, G, F\}$	$\{A, B, E, G, F\}$

Ejemplo Búsqueda de caminos en profundidad (II)

- Si se considera el grafo dirigido de la figura, el algoritmo de búsqueda de caminos en profundidad realiza diversas etapas de avance y retroceso hasta encontrar el camino $A \rightarrow C \rightarrow F$.
- La tabla recoge las etapas de este ejemplo.
- El proceso de búsqueda llega en algún momento al nodo G, pero $Ady(G) = \Phi$. Por tanto, el algoritmo vuelve al nodo anterior para poder continuar el proceso de búsqueda.



Etapas	X_k	$Ady(X_k) \setminus Visitados$	Visitados	Camino
1	A	$\{B, C, D\}$	$\{A\}$	$\{A\}$
2, 3	B	$\{E\}$	$\{A, B\}$	$\{A, B\}$
2, 3	E	$\{G\}$	$\{A, B, E\}$	$\{A, B, E\}$
2, 3	G	ϕ	$\{A, B, E, G\}$	$\{A, B, E, G\}$
2, 4	E	ϕ	$\{A, B, E, G\}$	$\{A, B, E\}$
2, 4	B	ϕ	$\{A, B, E, G\}$	$\{A, B\}$
2, 4	A	$\{C, D\}$	$\{A, B, E, G\}$	$\{A\}$
2, 3	C	$\{F\}$	$\{A, B, E, G, C\}$	$\{A, C\}$
2, 3	F	ϕ	$\{A, B, E, G, C, F\}$	$\{A, C, F\}$

Notas sobre el algoritmo anterior

- El algoritmo anterior siempre obtiene caminos simples, es decir, caminos que no contienen dos veces el mismo nodo (no contienen bucles ni ciclos).
  - Sin embargo, como veremos más adelante, el algoritmo puede modificarse fácilmente para hallar bucles y ciclos.
- **El camino encontrado por este algoritmo no es, generalmente, el camino más corto entre los nodos.**
  - A continuación se considera la estrategia de *búsqueda en anchura* describiendo un algoritmo para comprobar si existe un camino entre un par de nodos dados.

Algoritmo Búsqueda de caminos en anchura

- **Datos:** Un grafo arbitrario (*X, Ady*) y dos nodos *Xi* y *Xj*.
- **Resultado:** Existencia o no de un camino entre *Xi* y *Xj*.

1. *Iniciación:* Definir *Visitados* = Φ y *Cola* = {*Xi*}.
2. *Iteración:* Seleccionar el primer nodo, *Xk*, en la lista *Cola*, eliminarlo de esta lista y añadirlo a *Visitados*.
3. Si *Xk* = *Xj*, entonces existe un camino entre *Xi* y *Xj* y el algoritmo finaliza. En caso contrario, si todos los vecinos de *Xk* han sido visitados previamente, ir a la Etapa 4; en caso contrario, ir a la Etapa 5.
4. Si *Cola* = Φ, entonces no existe ningún camino entre *Xi* y *Xj* y el algoritmo finaliza. En caso contrario, ir a la Etapa 2.
5. Añadir a la lista *Cola* todos los nodos no visitados de *Ady*(*Xk*) e ir a la Etapa 2.

Algoritmo Búsqueda de caminos en anchura

- Se muestra el pseudocódigo para este algoritmo.
- En cada etapa del algoritmo se actualizan las listas:
  - ***Visitados***, que contiene los nodos que ya han sido visitados.
  - ***Cola***, que contiene los nodos en cola pendientes de visitar.
- Si durante la ejecución del algoritmo se alcanza *Xj*, se habrá hallado un camino. En caso contrario, tras la búsqueda se concluye que el camino no existe.

Algoritmo de Búsqueda de Caminos en Anchura

Datos: Un grafo (X, Ady) y dos nodos X_i y X_j.
Resultado: Existencia de un camino de X_i a X_j.

Etapa Inicial:

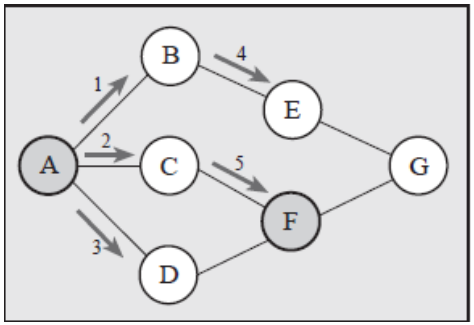
$Cola \leftarrow \{X_i\}$
$Visitados \leftarrow \phi$

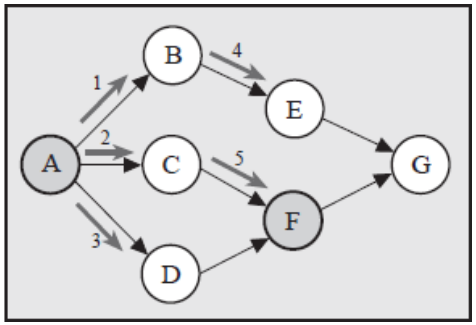
Etapa de Iteración:

$X_k \leftarrow$ primer nodo en $Cola$
Eliminar X_k de $Cola$
Añadir X_k a $Visitados$
si $X_k = X_j$, **entonces**
    Terminar (existe un camino de X_i a X_j).
en otro caso
    $S \leftarrow Ady(X_k) \setminus Visitados$
    si $S \neq \phi$ **entonces**
        Añadir S al comienzo de $Cola$
        Repetir la etapa de iteración.
    en otro caso
    si $Cola = \phi$, **entonces**
        Terminar (no existe ningún camino de X_i a X_j).
    en otro caso
        Repetir la etapa de iteración.

Ejemplo Búsqueda de caminos en anchura

- Se utiliza el algoritmo de búsqueda en anchura para comprobar si existe algún camino entre los nodos *A* y *F* en los grafos no dirigido y dirigido de la figura.
- En este caso, el algoritmo sigue las mismas etapas en ambos grafos. La tabla siguiente muestra los valores de las variables que intervienen en cada etapa del proceso. El algoritmo finaliza en la Etapa 3, concluyendo que existe un camino entre los nodos.

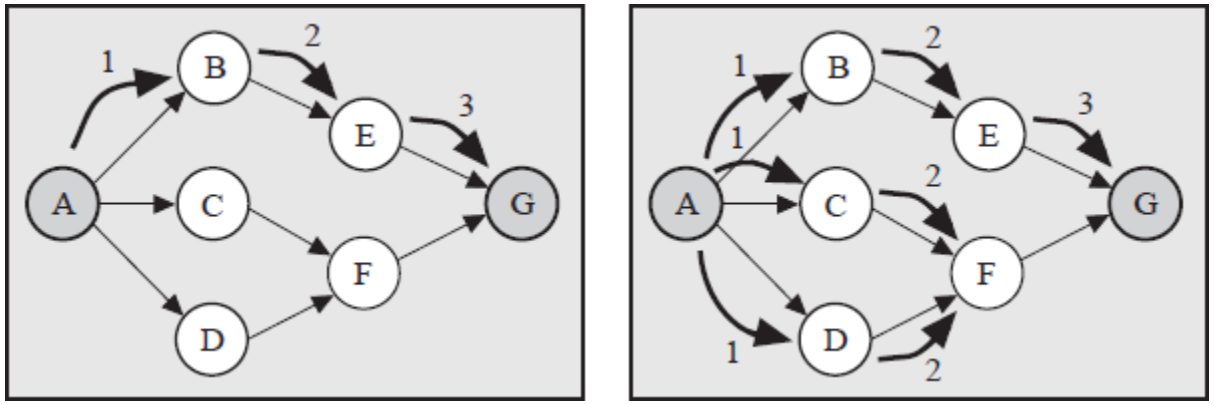




Etapas	Nodo X_k	*Visitados*	*Cola*
1	–	ϕ	$\{A\}$
2	A	$\{A\}$	$\{\}$
3,5	A	$\{A\}$	$\{B, C, D\}$
2	B	$\{A, B\}$	$\{C, D\}$
3,5	B	$\{A, B\}$	$\{C, D, E\}$
2	C	$\{A, B, C\}$	$\{D, E\}$
3,5	C	$\{A, B, C\}$	$\{D, E, F\}$
2	D	$\{A, B, C, D\}$	$\{E, F\}$
3,4	D	$\{A, B, C, D\}$	$\{E, F\}$
2	E	$\{A, B, C, D, E\}$	$\{F\}$
3,5	E	$\{A, B, C, D, E\}$	$\{F, G\}$
2	F	$\{A, B, C, D, E, F\}$	$\{G\}$

Notas sobre los algoritmos anteriores

- La complejidad de los algoritmos anteriores es lineal en el nº de aristas y nodos del grafo. La eficiencia de cada uno de estos algoritmos dependerá de la topología particular que se tenga en cada caso. ***En general, el algoritmo de búsqueda en profundidad es más eficiente*** que el de búsqueda en anchura ***cuando los caminos que unen los nodos inicial y final son largos*** (ver figura):



- En cambio, ***la situación es la contraria si los nodos están unidos por caminos cortos***.

Comprobando la Conexión de un Grafo

- Los ***métodos de búsqueda*** descritos anteriormente también ***pueden utilizarse para comprobar si un grafo es conexo***.
- La idea es realizar una **búsqueda exhaustiva** de los nodos del grafo, **obteniendo el conjunto *S* de nodos** que son **alcanzables desde un nodo inicial**.
  - Si el grafo es conexo, entonces el conjunto *S* contendrá todos los nodos del grafo;
  - en caso contrario, el subconjunto de nodos *S* sólo contendrá la componente conexa del grafo que contiene al nodo inicial.
- Los algoritmos anteriores pueden ser utilizados para realizar una búsqueda exhaustiva considerando el mismo nodo inicial y final, es decir, el conjunto *Visitados* resultante de la ejecución de estos algoritmos contendrá la componente conexa correspondiente a *Xi*.

Algoritmo Búsqueda de componentes conexas

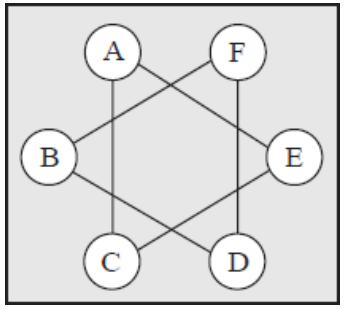
- **Datos:** Un grafo (*X, Ady*).
- **Resultado:** El conjunto de componentes conexas *C* de (*X, Ady*).

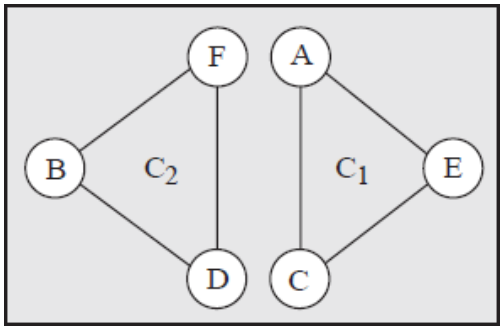
1. *Iniciación:* Definir *Visitados* = Φ, *C* = Φ.
2. Si *X* \ *Visitados* = Φ, finalizar y devolver *C*; en caso contrario, elegir un nodo de *Xi* $\in$ *X* \ *Visitados* e ir a la Etapa 3. Utilizar uno de los dos algoritmos anteriores para realizar una búsqueda exhaustiva del grafo (*X, Ady*) comenzando en el nodo *Xi* y obtener el conjunto *S* de nodos visitados.
3. Añadir *S* a *C*. Añadir a *Visitados* todos los nodos en *S*. Ir a la Etapa 2.

Si el conjunto *C* contiene una sola componente conexa, entonces el grafo es conexo; en caso contrario, el grafo es inconexo y *C* contiene todas las componentes conexas del grafo.

Ejemplo Búsqueda de Componentes Conexas

- Se ha visto que el grafo no dirigido de la figura es inconexo.
- Utilizando el algoritmo se pueden calcular sus componentes conexas.



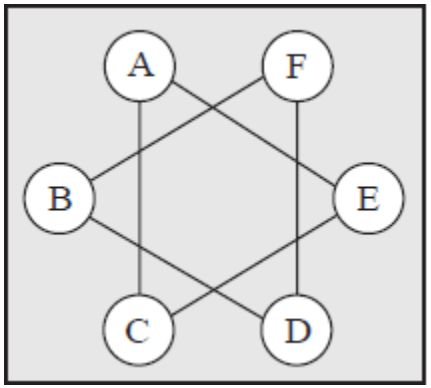


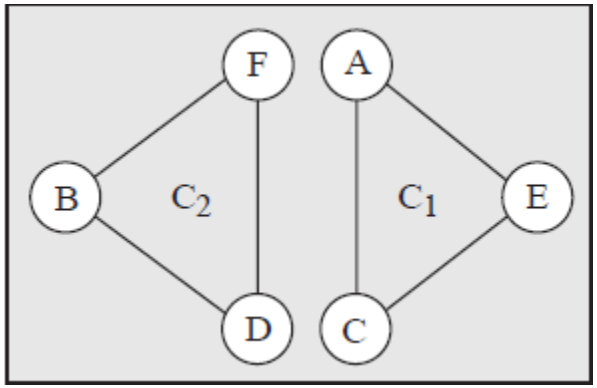
Ejemplo Búsqueda de Componentes Conexas

- Se considera *Visitados* = Φ y *C* = Φ.
- *X* \ *Visitados* = *X* = *{A,B,C,D,E, F}*. Se elige el primero de estos nodos como nodo inicial *Xk* = *A* para la primera búsqueda exhaustiva.
- Se utiliza el algoritmo de búsqueda en profundidad con *Xi* = *Xj* = *A*, obteniéndose el conjunto de nodos visitados *S* = C1 = *{A,C,E}*.
- Por tanto, se tiene *C* = *{C1}* y *Visitados* = *{A,C,E}*. *X* \ *Visitados* = *{B,D,F}*. Se toma *Xk* = *B*.
- Utilizando de nuevo el algoritmo de búsqueda en profundidad con *Xi* = *Xj* = *B*, se obtiene el conjunto de nodos visitados C2 = *{B,D,F}*.
- Ahora se tiene *Visitados* = *{A,C,E,B,D, F}*, *C* = *{C1, C2}*.
- Dado que *X* \ *Visitados* = Φ, el algoritmo finaliza obteniendo *C*.

Ejemplo Búsqueda de Componentes Conexas

- Entre las componentes conexas están los subconjuntos C1 = *{A,C,E}* y C2 = *{B,D,F}*. Por tanto, el grafo de la Figura de la izquierda es inconexo y contiene las dos componentes conexas, C1 y C2, tal y como se muestra en la figura de la derecha.





Búsqueda de Bucles y Ciclos

- Como ya se mencionó al final de un ejemplo anterior, **los algoritmos de búsqueda de caminos pueden modificarse fácilmente para hallar bucles o ciclos en un grafo**.
- Se muestran las modificaciones necesarias para *adaptar el algoritmo de búsqueda en profundidad* para esta tarea.
- Dado que el objetivo de este algoritmo es hallar un camino cerrado (un bucle o un ciclo), ***se puede utilizar*** el algoritmo de búsqueda en profundidad ***comprobando en cada etapa si hay algún nodo*** contenido ***en el camino*** que ***también*** lo esté ***en la lista de nodos adyacentes del*** nodo ***actual***.
- **Los caminos cerrados resultantes serán bucles** (si el grafo es no dirigido) **o ciclos** (si el grafo es dirigido).
- El algoritmo selecciona un nodo inicial arbitrario y busca de forma exhaustiva un camino cerrado en el grafo.

Algoritmo Búsqueda de caminos cerrados en profundidad

- **Datos:** Un grafo (*X, Ady*).
- **Resultado:** Un camino cerrado, *Camino*. Si el grafo no contiene ningún camino cerrado, entonces *Camino* = Φ.

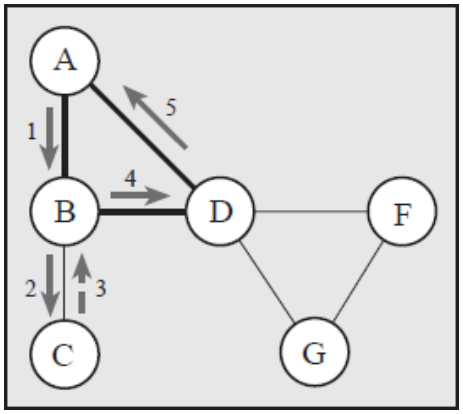
1. *Iniciación:* Definir *Camino* = Φ y *Visitados* = Φ.
2. Si existe algún nodo $Xi \in X \setminus Visitados$, ir a la Etapa 3; en caso contrario, el algoritmo finaliza (no existe ningún camino cerrado en el grafo).
3. Añadir *Xi* a *Visitados* y tomar *Camino* = {*Xi*}, tomar *Xk* = *Xi* y *Previo* = *Xi*.
4. *Iteración:* Si existe algún nodo $Xr \in Ady(Xk) \cap Camino$, con $Xr \neq Previo$, entonces añadir *Xr* a *Camino* y finalizar (se ha encontrado un camino cerrado); en caso contrario, ir a la Etapa 5.
5. Si todos los nodos de *Ady*(*Xk*) han sido ya visitados, o *Ady*(*Xk*) = Φ, ir a la Etapa 7; en caso contrario, ir a la Etapa 6.
6. *Etapa de Avance:* Elegir algún nodo $Xr \in Ady(Xk)$, tal que $Xr \notin Visitados$. Definir *Previo* = *Xk*, añadir *Xr* a *Camino* y *Visitados*, tomar *Xk* = *Xr*, e ir a la Etapa 4.
7. *Etapa de Retroceso:* Eliminar *Xk* de *Camino*. Si *Xk* = *Xi*, ir a la Etapa 2; en caso contrario, asignar a *Xk* el ´ultimo nodo en *Camino*, e ir a la Etapa 5.

Nota sobre el algoritmo

- El algoritmo anterior considera un nodo arbitrario del grafo, *Xi*, como nodo inicial.
  - Si no se encuentra ningún camino cerrado (el algoritmo vuelve al nodo original), entonces se comprueba si todos los nodos han sido visitados concluyéndose, en ese caso, que no existe ningún camino cerrado en el grafo;
  - en caso contrario, el algoritmo elige alguno de los nodos no visitados como nodo inicial y repite el proceso.
- La forma en que este algoritmo está concebido hace que no sólo sea válido para grafos dirigidos y no dirigidos, sino también para grafos conexos e inconexos.
- El siguiente ejemplo ilustra la aplicación de este algoritmo.

Ejemplo Búsqueda de Bucles y Ciclos

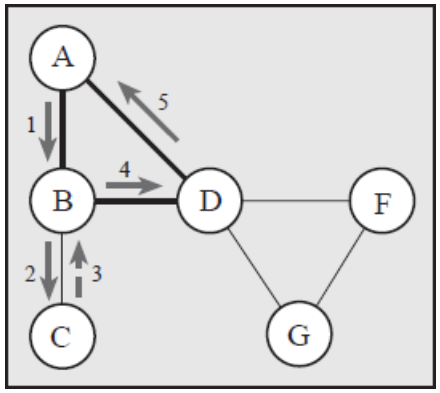
- Considere el grafo no dirigido dado en la figura que contiene dos bucles, *A−B−D−A* y *D−G−F −D*.
- Suponga que se aplica el algoritmo comenzando en el nodo *A*.
- La tabla muestra las etapas seguidas por el algoritmo. Muestra, en cada etapa, el nodo *Xk*, el nodo *Previo* asociado, el *Camino* actual, el conjunto *Ady*(*Xk*) ∩ *Camino*, que se utiliza para indicar si existe algún camino cerrado, y el conjunto *Visitados* que contiene los nodos que han sido visitados.
- Las etapas seguidas por el algoritmo se ilustran en la figura.



Etapa	X_k	*Previo*	*Camino*	*Visitados*
1	–	–	ϕ	ϕ
2, 3	A	A	$\{A\}$	$\{A\}$
4, 5, 6	B	A	$\{A, B\}$	$\{A, B\}$
4, 5, 6	C	B	$\{A, B, C\}$	$\{A, B, C\}$
4, 5, 7	B	B	$\{A, B\}$	$\{A, B, C\}$
5, 6	D	B	$\{A, B, D\}$	$\{A, B, C, D\}$
4	A	–	$\{A, B, D, A\}$	$\{A, B, C, D\}$

Ejemplo Búsqueda de Bucles y Ciclos

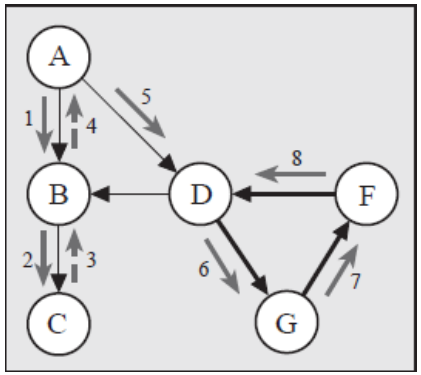
- Estas etapas se resumen:
  - Inicialmente se viaja de *A* a *B* y de *B* a *C*. Al alcanzar el nodo *C* ya no es posible avanzar, luego se vuelve un paso atrás, al nodo *B* y se viaja al único vecino aún no visitado, *D*.
  - El conjunto *Ady(D)* ∩ *Camino* contiene al nodo *A*, que no es el nodo *Previo* a *D*. Por tanto, se ha encontrado el bucle *A* − *B* − *D* − *A*. Si se ignorase el nodo *A* se podría continuar viajando, buscando un bucle distinto. De esta forma se pueden obtener todos los bucles contenidos en el grafo.
- Por ejemplo, si en la Etapa 5 se eligiese el nodo *G* o *F* en lugar del nodo *A*, se obtendría un bucle distinto: *D* − *G* − *F* − *D*.



Etapa	X_k	*Previo*	*Camino*	*Visitados*
1	–	–	ϕ	ϕ
2, 3	A	A	$\{A\}$	$\{A\}$
4, 5, 6	B	A	$\{A, B\}$	$\{A, B\}$
4, 5, 6	C	B	$\{A, B, C\}$	$\{A, B, C\}$
4, 5, 7	B	B	$\{A, B\}$	$\{A, B, C\}$
5, 6	D	B	$\{A, B, D\}$	$\{A, B, C, D\}$
4	A	–	$\{A, B, D, A\}$	$\{A, B, C, D\}$

Ejemplo Búsqueda de Bucles y Ciclos

- Considere ahora el grafo dirigido cíclico de la figura.
- Procediendo de la misma forma que en el caso anterior, y comenzando en el nodo *A*, el algoritmo realiza las etapas indicadas en la tabla y la figura.
- En este caso, el algoritmo termina hallando el ciclo $D \rightarrow G \rightarrow F \rightarrow D$.
- Observe que el grafo de la figura del ejemplo anterior es el grafo no dirigido asociado a este grafo dirigido.
- Por tanto, sin más que cambiar los conjuntos de adyacencia se pueden obtener los ciclos de un grafo dirigido, o los bucles del grafo no dirigido asociado.



Etapa	X_k	*Previo*	*Camino*	*Visitados*
1	–	–	ϕ	ϕ
2, 3	A	A	$\{A\}$	$\{A\}$
4, 5, 6	B	A	$\{A, B\}$	$\{A, B\}$
4, 5, 6	C	B	$\{A, B, C\}$	$\{A, B, C\}$
4, 5, 7	B	B	$\{A, B\}$	$\{A, B, C\}$
5, 7	A	B	$\{A\}$	$\{A, B, C\}$
5, 6	D	A	$\{A, D\}$	$\{A, B, C, D\}$
4, 5, 6	G	D	$\{A, D, G\}$	$\{A, B, C, D, G\}$
4, 5, 6	F	G	$\{A, D, G, F\}$	$\{A, B, C, D, G, F\}$
4	F	G	$\{A, D, G, F, D\}$	$\{A, B, C, D, G, F\}$