

Algoritmos y Estructuras de Datos

Cursada 2015

Prof. Alejandra Schiavoni

Prof. Catalina Mostaccio

Facultad de Informática – UNLP



GRAFOS

Algoritmos y Estructuras de Datos



Agenda - Grafos

1. Ejemplos y terminología
2. Representaciones
3. Recorridos



Agenda - Grafos

1. Ejemplos y terminología
2. Representaciones
3. Recorridos

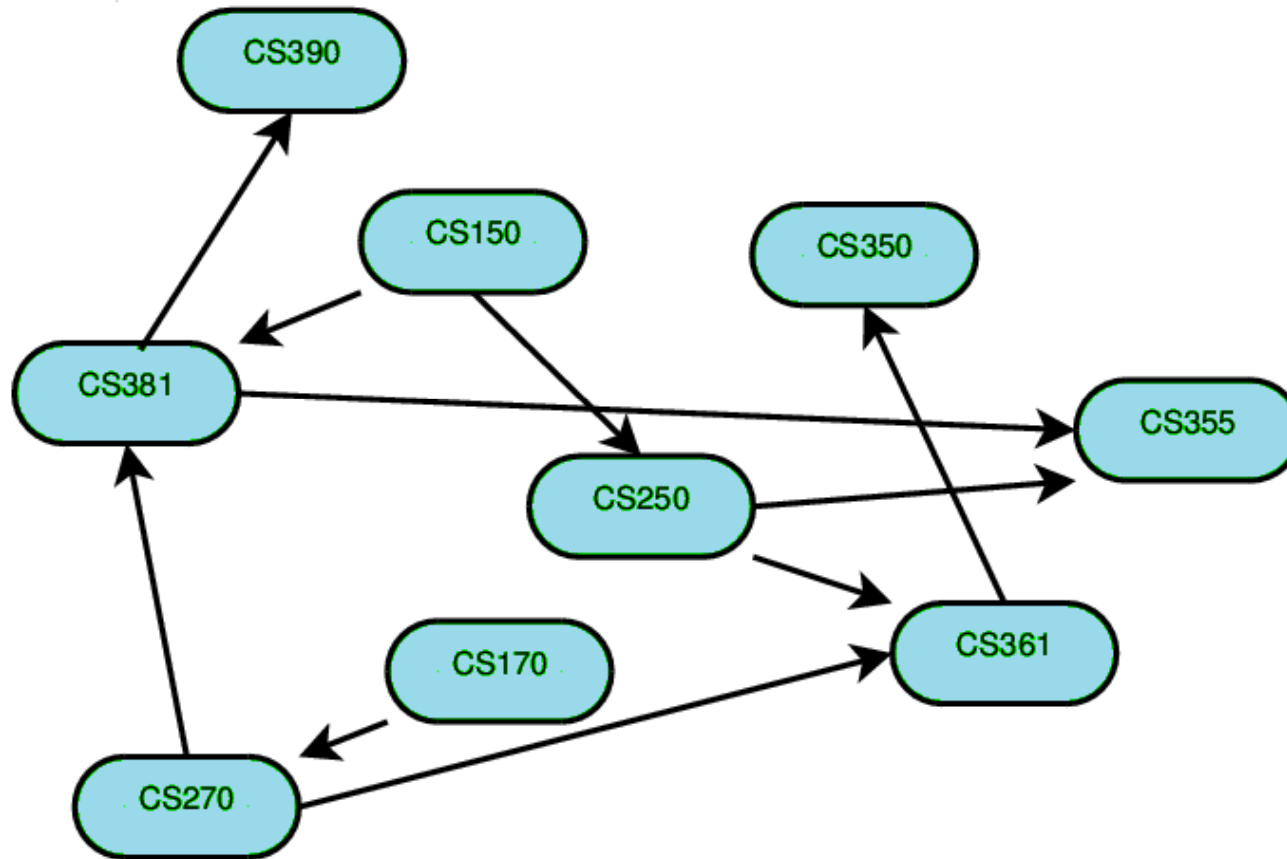
Ejemplo 1: Mapa de ciudades





Ciudades conectadas por
Rutas con *distancias* en Km

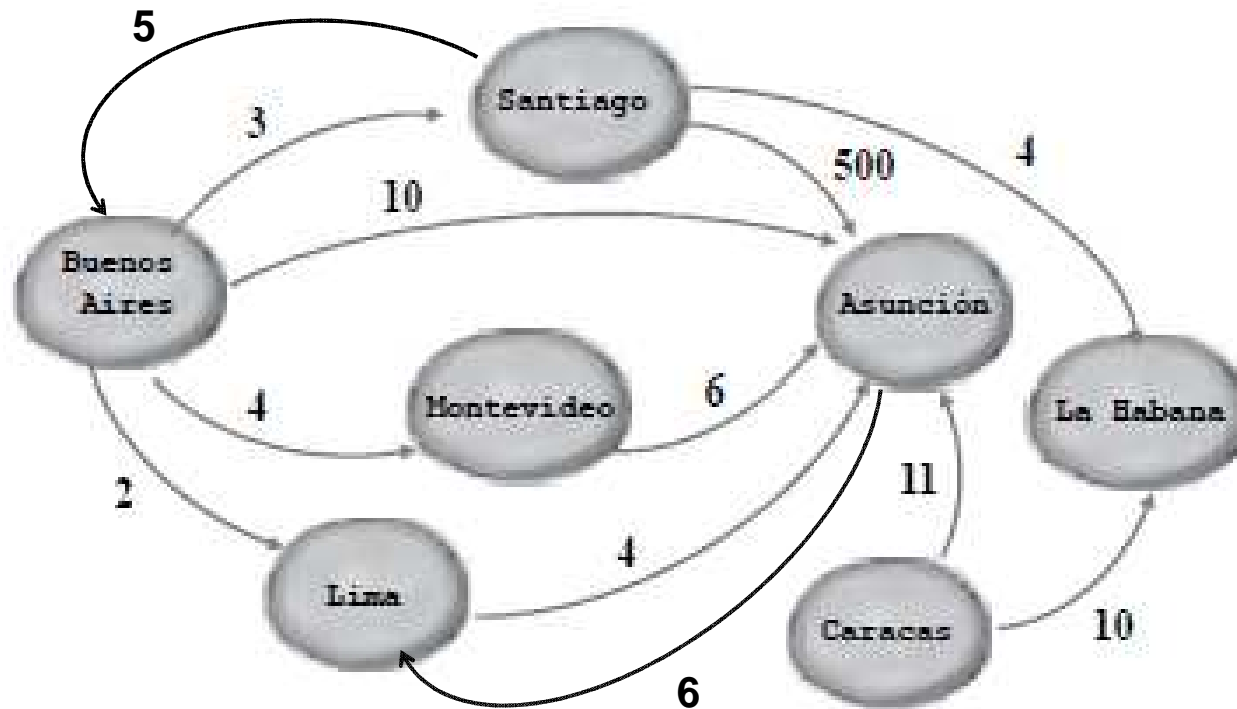
Ejemplo 3: Prerrequisitos de un curso



Cursos conectados por sus correlativas
(**relación** de “prerrequisito”)



Ejemplo 4: Mapa de rutas áreas entre ciudades



Ciudades conectadas por **Rutas aéreas** con sus respectivos *tiempos de vuelo*

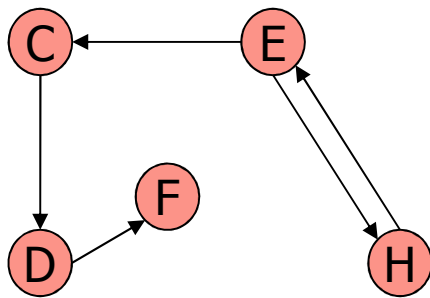


Terminología

- *Grafo* → modelo para representar relaciones entre elementos de un conjunto.
- **Grafo:** (V, E) , V es un conjunto de vértices o nodos, con una relación entre ellos; E es un conjunto de pares (u, v) , $u, v \in V$, llamados aristas o arcos.
- **Grafo dirigido:** la relación sobre V no es simétrica. Arista \equiv par ordenado (u, v) . (Ejemplo 3)
- **Grafo no dirigido:** la relación sobre V es simétrica. Arista \equiv par no ordenado $\{u, v\}$, $u, v \in V$ y $u \neq v$. (Ejemplos 1 y 2)

Terminología (cont. 1)

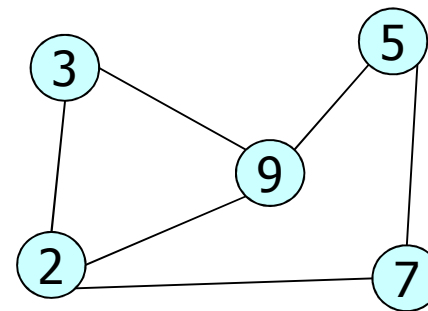
Ejemplos



Grafo dirigido $G(V,E)$.

$V = \{C,D,E,F,H\}$

$E = \{(C,D), (D,F), (E,C), (E,H), (H,E)\}$



Grafo no dirigido $G(V,E)$.

$V = \{2,3,5,7,9\}$

$E = \{\{2,3\}, \{2,7\}, \{2,9\}, \{3,9\}, \{5,7\}, \{5,9\}\}$

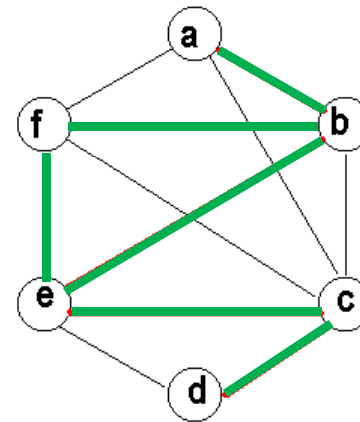
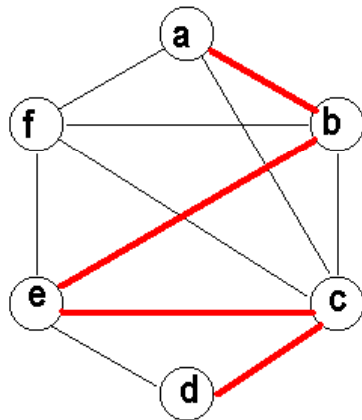


Terminología (cont. 2)

- v es **adyacente** a u si existe una arista $(u,v) \in E$.
 - en un grafo no dirigido, $(u,v) \in E$ **incide** en los nodos u, v .
 - en un grafo dirigido, $(u,v) \in E$ **incide** en v , y **parte** de u .
- En grafos no dirigidos:
 - El **grado** de un nodo: número de arcos que inciden en él.
- En grafos dirigidos:
 - existen el grado de salida (**grado_out**) y el grado de entrada (**grado_in**).
 - el **grado_out** es el número de arcos que parten de él y
 - el **grado_in** es el número de arcos que inciden en él.
 - El **grado** del vértice será la suma de los grados de entrada y de salida.
- **Grado de un grafo**: máximo grado de sus vértices.

Terminología (cont. 3)

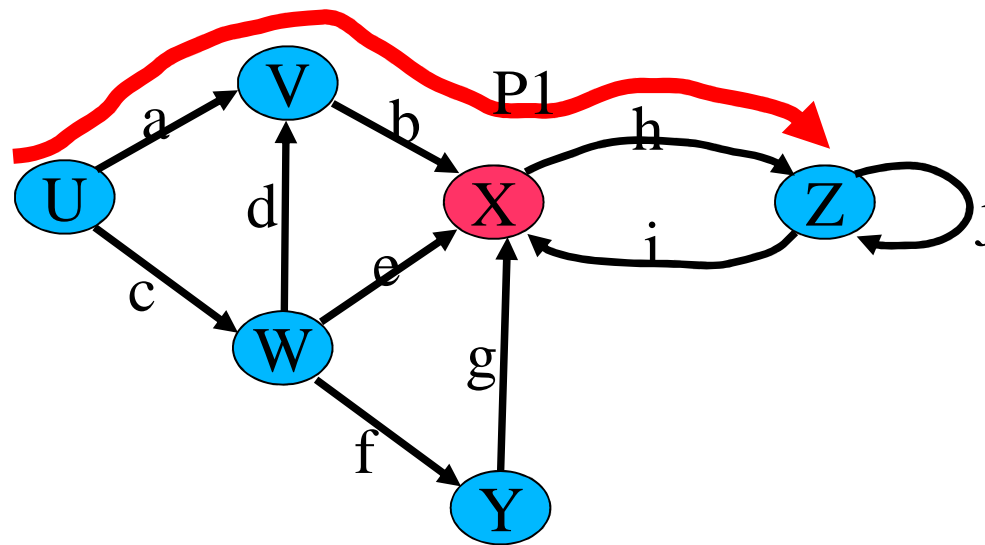
- **Camino** desde $u \in V$ a $v \in V$: secuencia v_1, v_2, \dots, v_k tal que $u=v_1, v=v_k$ y $(v_{i-1}, v_i) \in E$, para $i = 2, \dots, k$.
Ej: camino desde **a** a **d** $\rightarrow \langle a, b, e, c, d \rangle$.



- **Longitud de un camino**: número de arcos del camino.
Ejs: long. del camino desde **a** a **d** $\rightarrow \langle a, b, e, c, d \rangle$ es 4. (a)
long. del camino desde **a** a **d** $\rightarrow \langle a, b, e, f, b, e, c, d \rangle$ es 7. (b)

Terminología (cont. 4)

- **Camino simple:** camino en el que todos sus vértices, excepto, tal vez, el primero y el último, son distintos. *P1 es un camino simple desde U a Z.*

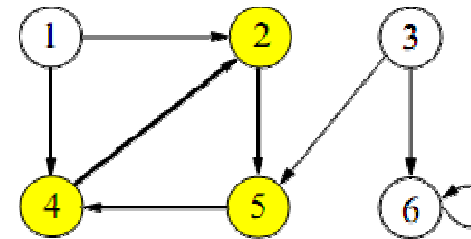


Ejemplos anteriores: (a) es camino simple , (b) no lo es.

Terminología (cont. 5)

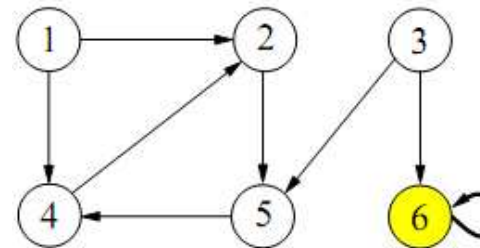
- **Ciclo:** camino desde v_1, v_2, \dots, v_k tal que $v_1 = v_k$

Ej: $\langle 2, 5, 4, 2 \rangle$ es un ciclo de longitud 3.

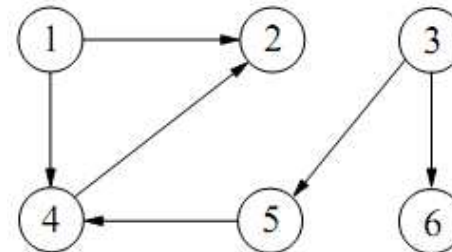


El ciclo es simple si el camino es simple.

- **Bucle:** ciclo de longitud 1.

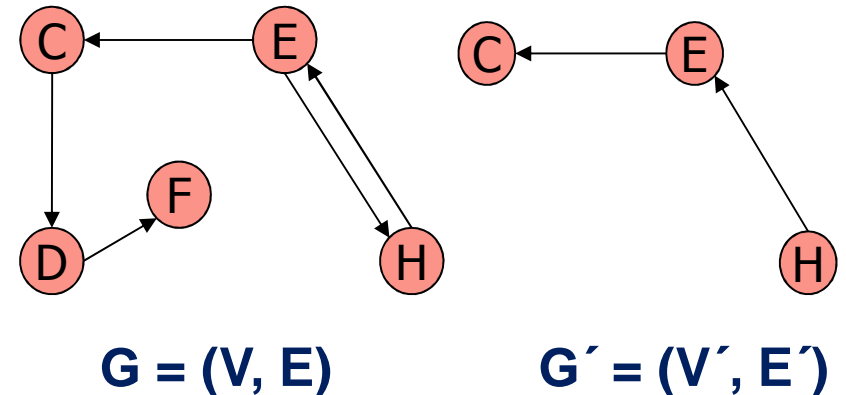
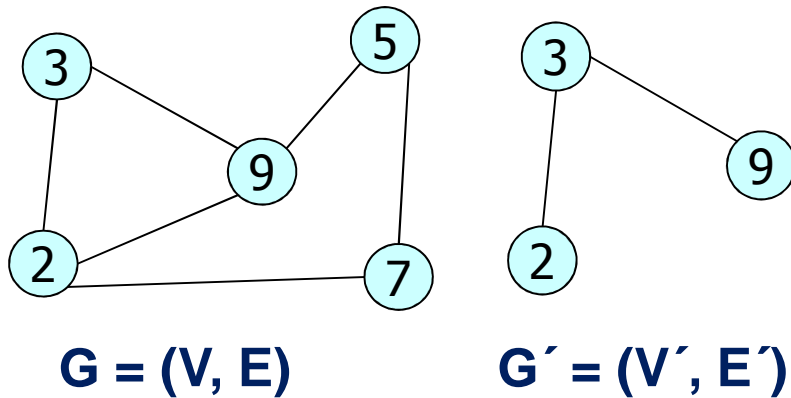


- **Grafo acíclico:** grafo sin ciclos.



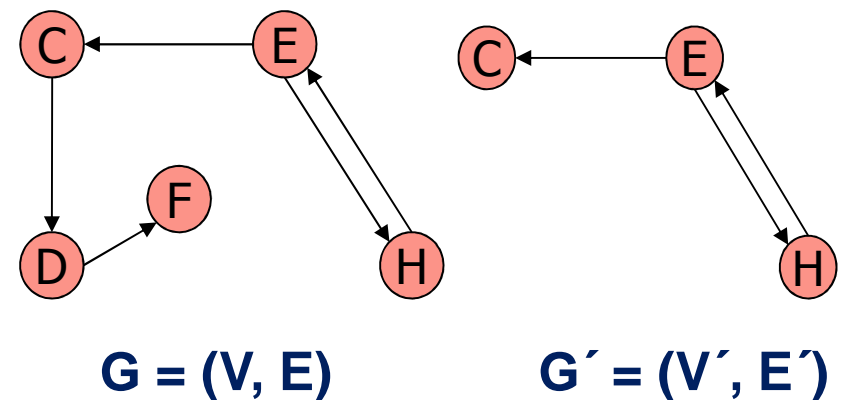
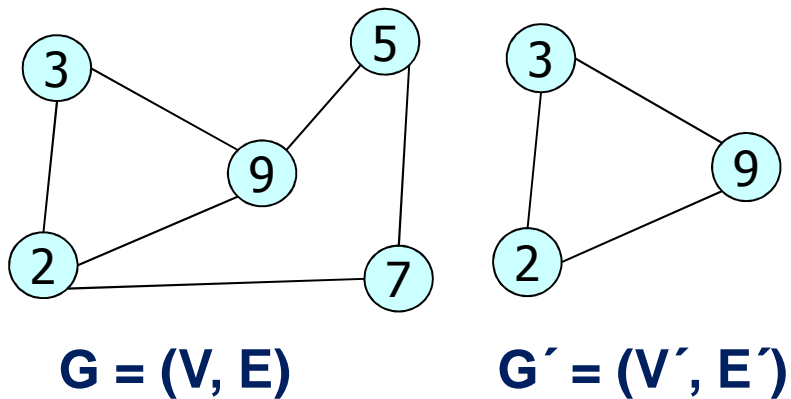
Terminología (cont. 6)

➤ Dado un grafo $G=(V, E)$, se dice que $G'=(V', E')$ es un subgrafo de G , si $V' \subseteq V$ y $E' \subseteq E$.



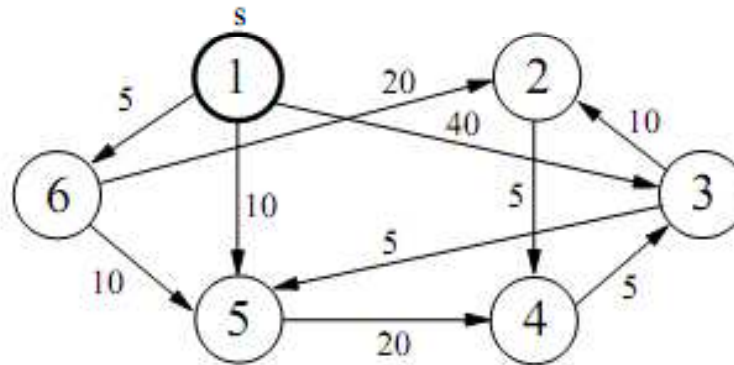
Terminología (cont. 7)

➤ *Un subgrafo inducido por $V' \subseteq V$: $G' = (V', E')$ tal que $E' = \{(u, v) \in E \mid u, v \in V'\}$.*



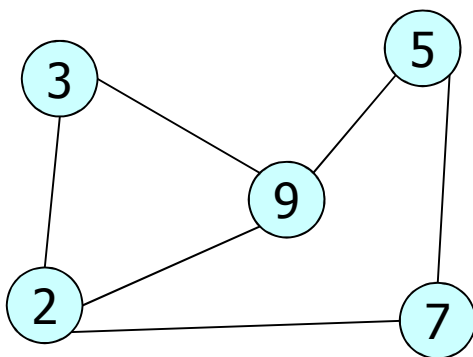
Terminología (cont. 8)

➤ *Un grafo **ponderado**, **pesado** o **con costos**: cada arco o arista tiene asociado un valor o etiqueta. (Ejemplos 2 y 4)*

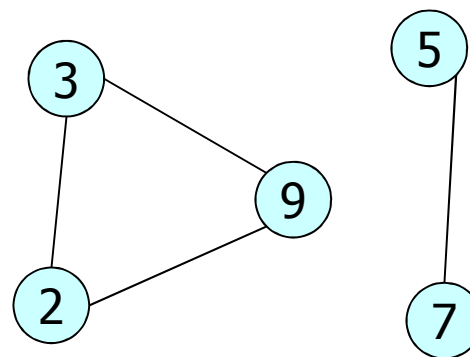


Conectividad en grafos no dirigidos

- *Un grafo no dirigido es **conexo** si hay un camino entre cada par de vértices.*



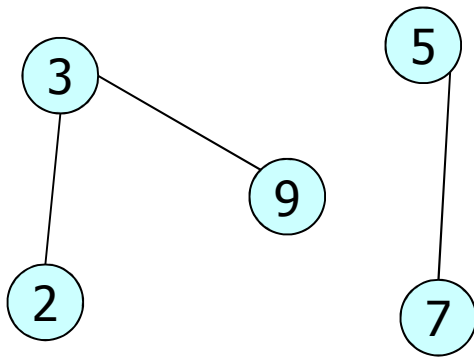
Conexo



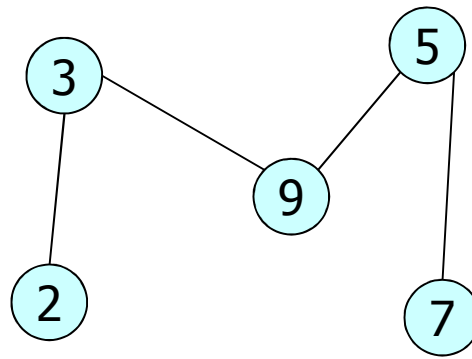
No Conexa

Conectividad: bosque y árbol

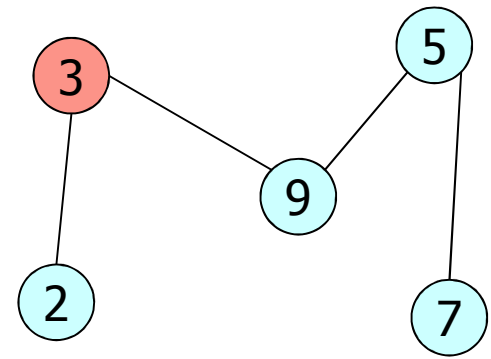
- *Un **bosque** es un grafo sin ciclos.*
- *Un **árbol libre** es un bosque conexo.*
- *Un **árbol** es un árbol libre en el que un nodo se ha designado como raíz.*



Bosque



Árbol libre



Árbol



Propiedades

➤ Sea G un grafo no dirigido con n vértices y m arcos, entonces

$$\sum_{v \in G} \deg(v) = 2 * m$$

✓ Siempre: $m \leq (n * (n-1)) / 2$

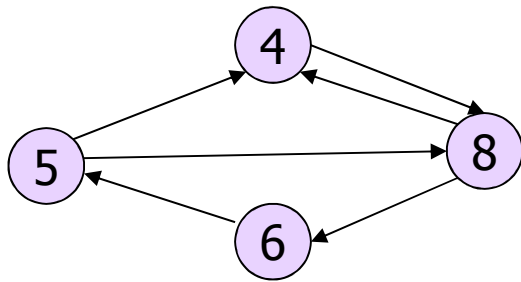
✓ Si G conexo: $m \geq n-1$

✓ Si G árbol: $m = n-1$

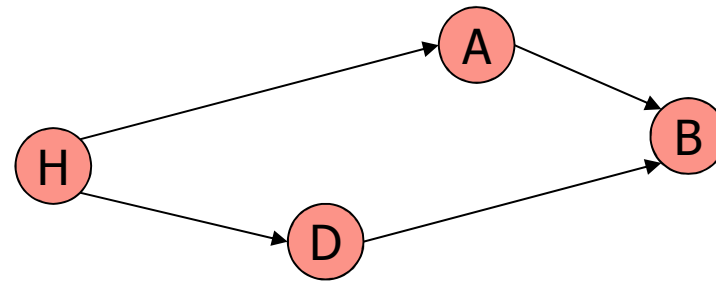
✓ Si G bosque: $m \leq n-1$

Conectividad en grafos dirigidos

- v es *alcanzable desde* u , si existe un camino de u a v .
- Un grafo dirigido se denomina **fuertemente conexo** si existe un camino desde cualquier vértice a cualquier otro vértice



Fuertemente Conexo



**No Fuertemente Conexo
Débilmente Conexos**

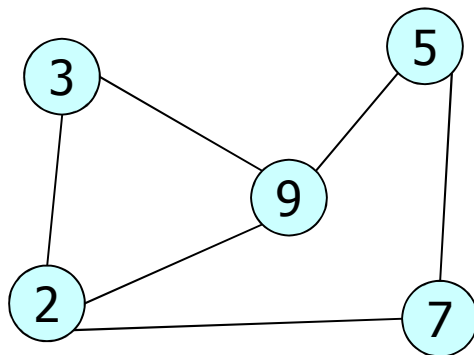
- Si un grafo dirigido no es fuertemente conexo, pero el grafo subyacente (sin sentido en los arcos) es conexo, el grafo es **débilmente conexo**.

Componentes conexas

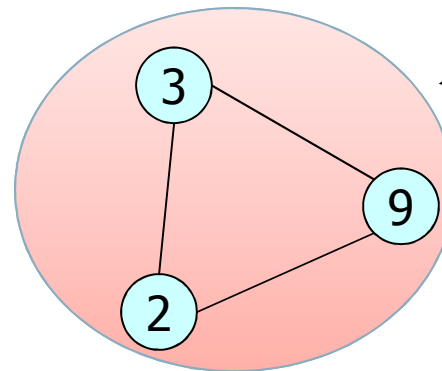
*En un grafo no dirigido, una **componente conexa** es un subgrafo conexo tal que no existe otra componente conexa que lo contenga.*

Es un subgrafo conexo maximal.

*Un grafo no dirigido es **no conexo** si está formado por varias componentes conexas.*



Conexo



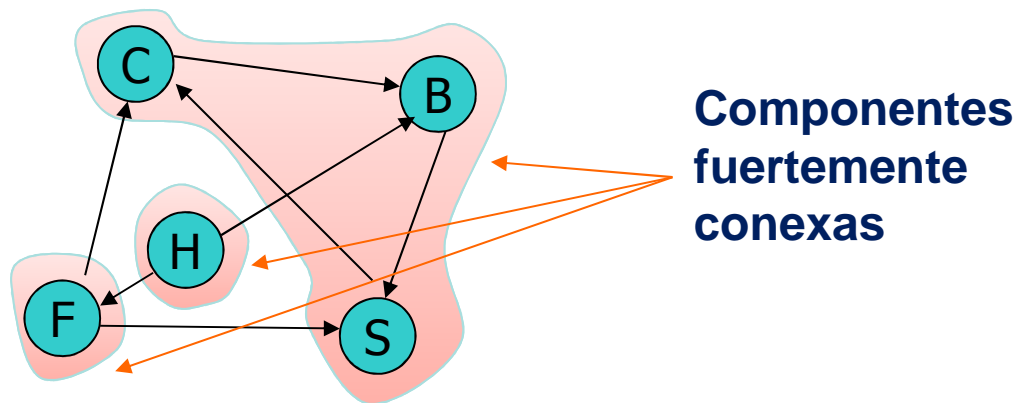
No Conexo

Componentes conexas:
entre ellas son conexas

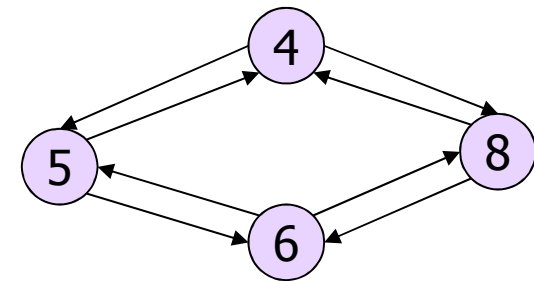
Componentes fuertemente conexas

*En un grafo dirigido, una **componente fuertemente conexa**, es el máximo subgrafo fuertemente conexo.*

*Un grafo dirigido es **no fuertemente conexo** si está formado por varias componentes fuertemente conexas.*



**No Fuertemente
Conexo**



Fuertemente Conexo



Agenda - Grafos

1. Ejemplos y terminología
2. Representaciones
3. Recorridos



Agenda - Grafos

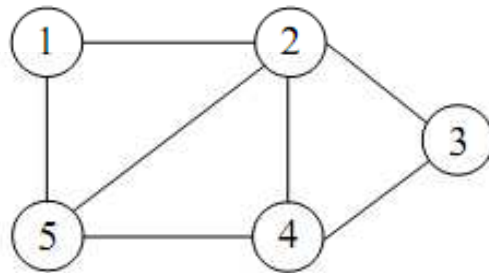
- Representaciones
 - Matriz de Adyacencias
 - Lista de Adyacencias



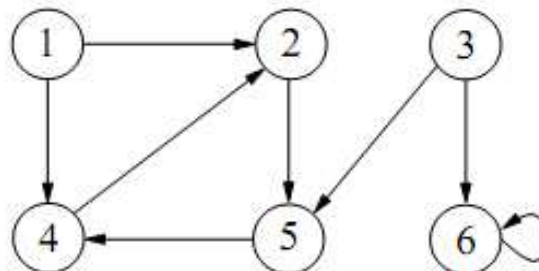
Representaciones: Matriz de Adyacencias

- $G=(V,E)$: matriz A de dimensión $|V| \times |V|$.
- Valor a_{ij} de la matriz:

$$a_{ij} = \begin{cases} 1 & \text{si } (i,j) \in E \\ 0 & \text{en cualquier otro caso} \end{cases}$$



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0



	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1



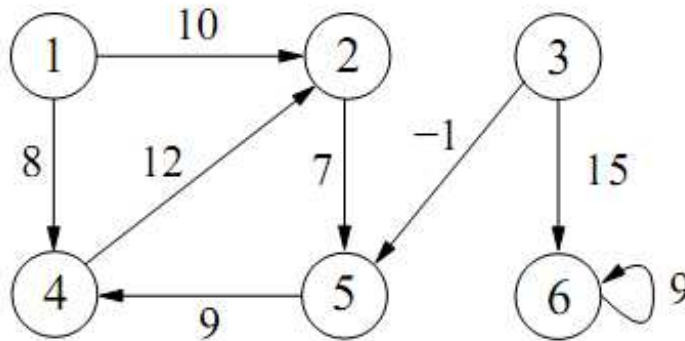
Representaciones: Matriz de Adyacencias

- *Costo espacial: $O(|V|^2)$*
- *Representación es útil para grafos con número de vértices pequeño, o grafos densos ($|E| \approx |V| \times |V|$)*
- *Comprobar si una arista (u,v) pertenece a $E \rightarrow$ consultar posición $A(u,v)$*
 - *Costo de tiempo $T(|V|, |E|) = O(1)$*

Representaciones: Matriz de Adyacencias

- Representación aplicada a Grafos pesados
- El peso de (i,j) se almacena en $A(i, j)$

$$a_{ij} = \begin{cases} w(i, j) & \text{si } (i, j) \in E \\ 0 \text{ o } \infty & \text{en cualquier otro caso} \end{cases}$$

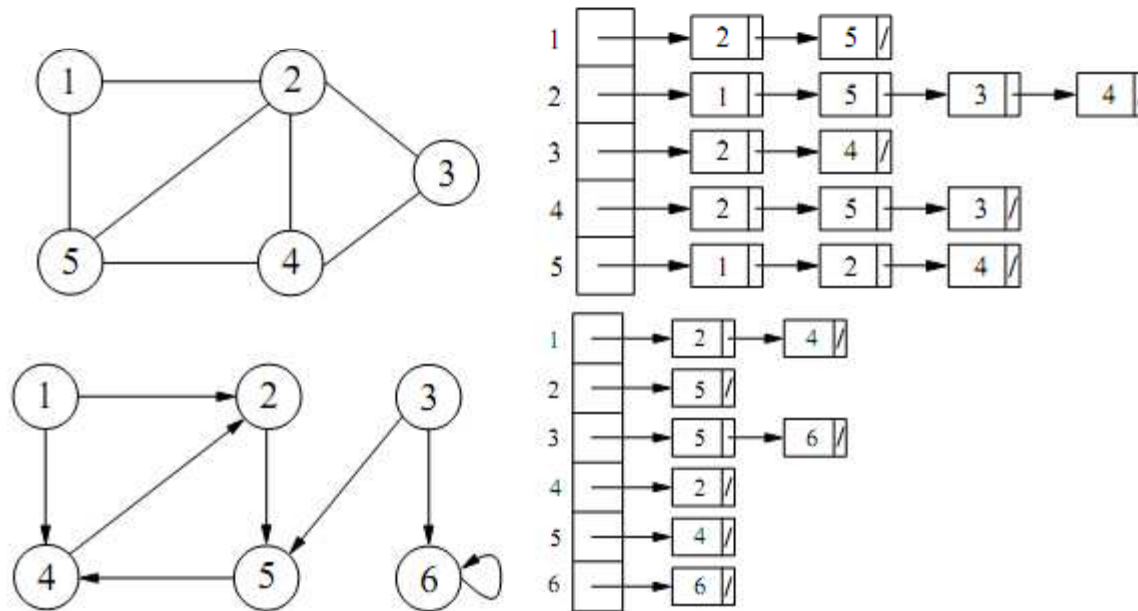


	1	2	3	4	5	6
1	0	10	0	8	0	0
2	0	0	0	0	7	0
3	0	0	0	0	-1	15
4	0	12	0	0	0	0
5	0	0	0	9	0	0
6	0	0	0	0	0	9

Representaciones: Lista de Adyacencias

- $G=(V,E)$: vector de tamaño $|V|$.
- Posición $i \rightarrow$ puntero a una lista enlazada de elementos (lista de adyacencia).

Los elementos de la lista son los vértices adyacentes a i



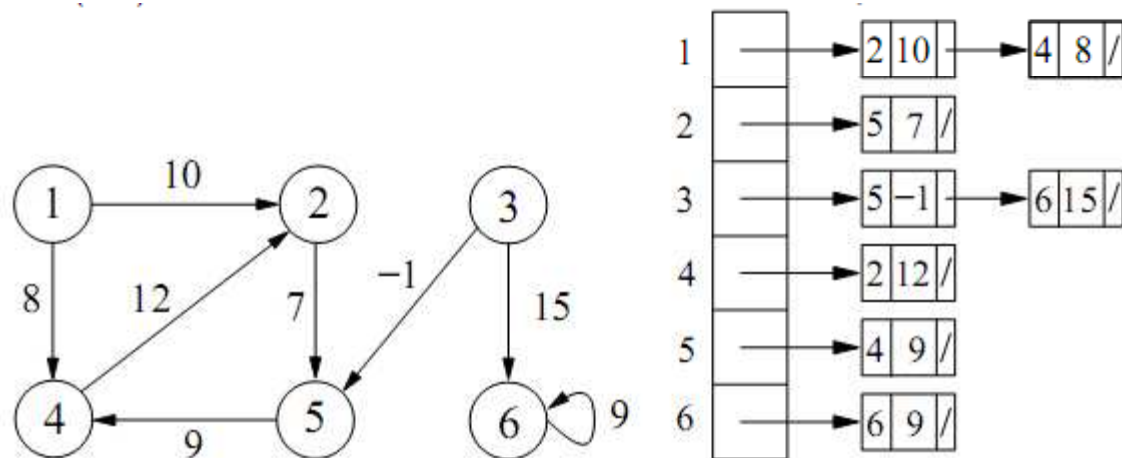


Representaciones: Lista de Adyacencias

- Si G es dirigido, la suma de las longitudes de las listas de adyacencia será $|E|$.
- Si G es no dirigido, la suma de las longitudes de las listas de adyacencia será $2|E|$.
- Costo espacial, sea dirigido o no: $O(|V| + |E|)$.
- Representación apropiada para grafos con $|E|$ menor que $|V|^2$.
- **Desventaja:** si se quiere comprobar si una arista (u, v) pertenece a $E \Rightarrow$ buscar v en la lista de adyacencia de u .
 - Costo temporal $T(|V|, |E|)$ será $O(\text{Grado } G) \subseteq O(|V|)$.

Representaciones: Lista de Adyacencias

- Representación aplicada a Grafos pesados
- El **peso de (u,v)** se almacena en el nodo de v de la lista de adyacencia de u .





Agenda - Grafos

1. Ejemplos y terminología
2. Representaciones
3. Recorridos



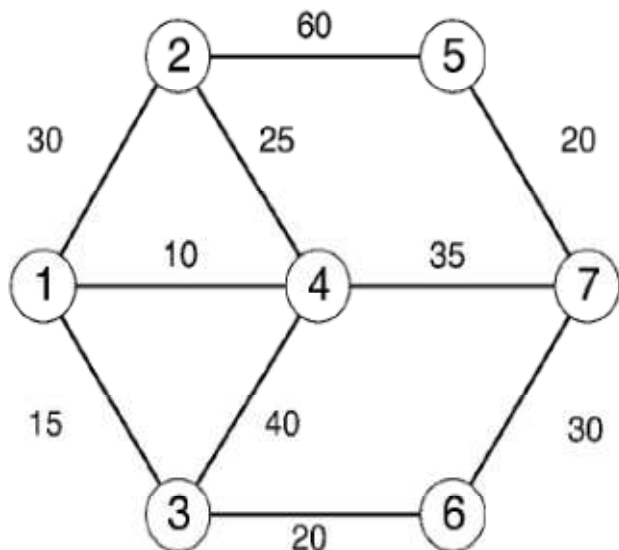
Agenda - Grafos

- Recorridos
 - en profundidad: DFS (Depth First Search)
 - en amplitud: BFS (Breath First Search)
 - Bosque de expansión DFS
 - Aplicaciones

Problema: El Guía de Turismo

El señor H es un guía de turismo de la ciudad de Buenos Aires. Su trabajo consiste en mostrar a grupos de turistas diferentes **puntos de interés** de la ciudad. Estos puntos de interés están **conectados por rutas en ambos sentidos**. Dos puntos de interés vecinos tienen un servicio de bus que los conecta, con una limitación en el **número máximo de pasajeros** que puede transportar. No es siempre posible para el señor H transportar de una única vez a todos los turistas a un destino en particular.

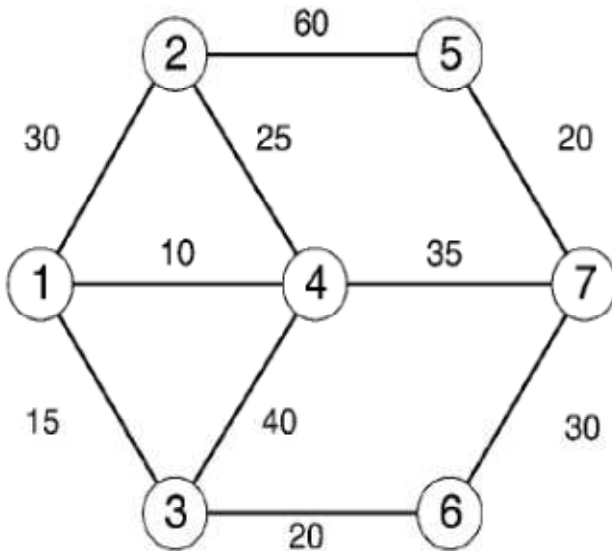
Por ejemplo, consideremos el siguiente mapa con 7 puntos de interés, donde las **aristas representan las rutas** y el **peso de ellas representa el límite máximo de pasajeros** a transportar por el servicio de bus. Su misión es indicarle al Sr. H cuál es el menor número de viajes que deberá realizar para llevar al grupo de turistas de un origen a un destino.



*En este ejemplo, el señor H debe transportar a **99 turistas** del punto 1 al punto 7.*

Veamos cuáles son los recorridos posibles y elijamos el que implique realizar el menor número de viajes.

Problema: El Guía de Turismo



vértices del recorrido	cant. turistas/viaje	cant. de viajes
1 2 4 3 6 7	20	6
1 2 4 7	25	5
1 2 5 7	20	6
1 3 4 2 5 7	15	8
1 3 4 7	15	8
1 3 6 7	15	8
1 4 2 5 7	10	11
1 4 3 6 7	10	11
1 4 7	10	11

Entonces, para transportar a los **99 turistas** del punto 1 al punto 7, se necesitarán 5 viajes, eligiendo la ruta:

1 - 2 - 4 - 7

En cada viaje el servicio de bus puede transportar como máximo a 25 pasajeros, 24 turistas + al señor H, en los cuatro primeros viajes transporta a 96 turistas y en el último a los restantes 3.



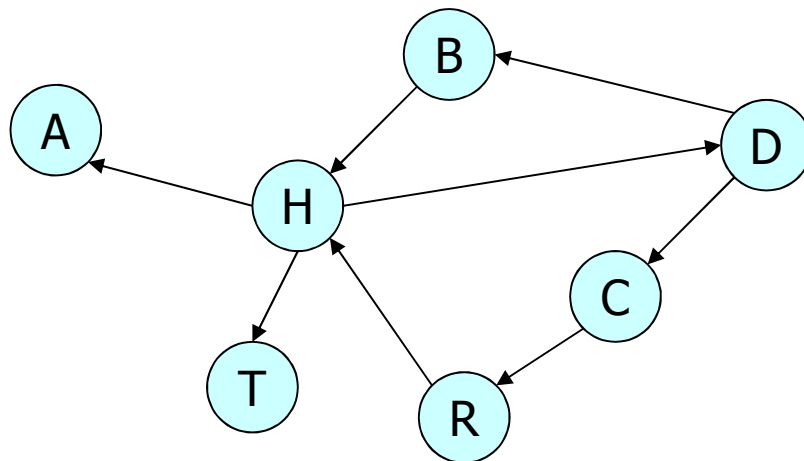
Recorrido en profundidad: DFS

→ *Generalización del recorrido preorden de un árbol.*

Estrategia:

- *Partir de un vértice determinado v .*
- *Cuando se visita un nuevo vértice, explorar cada camino que salga de él.*
- *Hasta que no se haya finalizado de explorar uno de los caminos no se comienza con el siguiente.*
- *Un camino deja de explorarse cuando se llega a un vértice ya visitado.*
- *Si existían vértices no alcanzables desde v el recorrido queda incompleto; entonces, se debe seleccionar algún vértice como nuevo vértice de partida, y repetir el proceso.*

Recorrido en profundidad: DFS



Si tomamos como vértice de partida a **D**

Se Muestra:

D C R H T A B



Recorrido en profundidad: DFS

Esquema recursivo: dado $G = (V, E)$

- 1. Marcar todos los vértices como no visitados.*
 - 2. Elegir vértice u como punto de partida.*
 - 3. Marcar u como visitado.*
 - 4. $\forall v$ adyacente a $u, (u, v) \in E$, si v no ha sido visitado, repetir recursivamente (3) y (4) para v .*
- Finalizar cuando se hayan visitado todos los nodos alcanzables desde u .*
 - Si desde u no fueran alcanzables todos los nodos del grafo: volver a (2), elegir un nuevo vértice de partida v no visitado, y repetir el proceso hasta que se hayan recorrido todos los vértices.*



Recorrido en profundidad : DFS

dfs (*v*: vértice)

marca[*v*]:= visitado;

para cada nodo *w* adyacente a *v*

si (marca[*w*] == noVisitado)

dfs(*w*);

main:dfs (grafo)

inicializar **marca** en false (arreglo de booleanos);

para cada vértice *v* del grafo

si (marca[*v*] == noVisitado)

dfs(*v*);



Recorrido DFS : Tiempo de ejecución

- *$G(V, E)$ se representa mediante listas de adyacencia.*
- *El método $dfs(v)$ se aplica únicamente sobre vértices no visitados
→ sólo una vez sobre cada vértice.*
- *$dfs(v)$ depende del número de vértices adyacentes que tenga (longitud de la lista de adyacencia).
→ el tiempo de todas las llamadas a $dfs(v)$: $O(|E|)$*
- *añadir el tiempo asociado al bucle de $main_dfs(\text{grafo})$: $O(|V|)$.
⇒ Tiempo del recorrido en profundidad es $O(|V| + |E|)$.*



Recorrido en amplitud : BFS

→ *Generalización del recorrido por niveles de un árbol.*

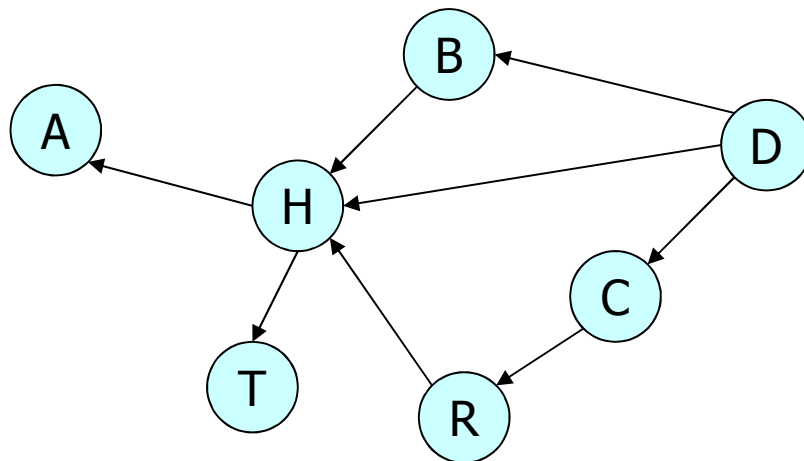
Estrategia:

- *Partir de algún vértice u , visitar u y, después, visitar cada uno de los vértices adyacentes a u .*
- *Repetir el proceso para cada nodo adyacente a u , siguiendo el orden en que fueron visitados.*



Recorrido en amplitud : BFS

Si tomamos como vértice de partida a **D**



Se Muestra:

D C H B R T A

Cola:

D C H B R T A



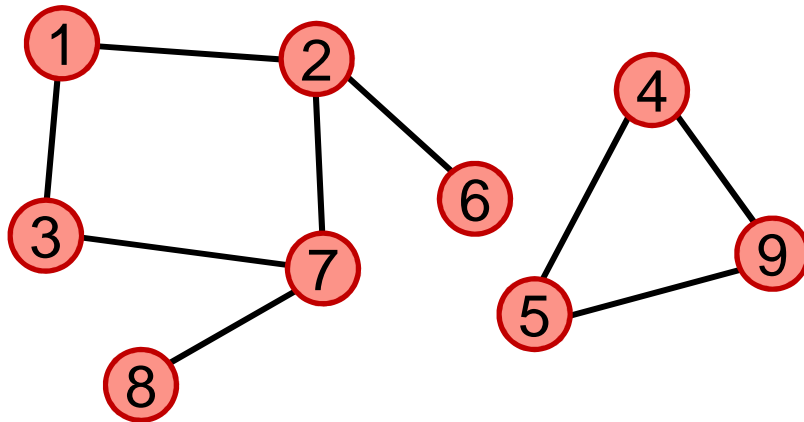
Recorrido en amplitud : BFS

Esquema iterativo: dado $G = (V, E)$

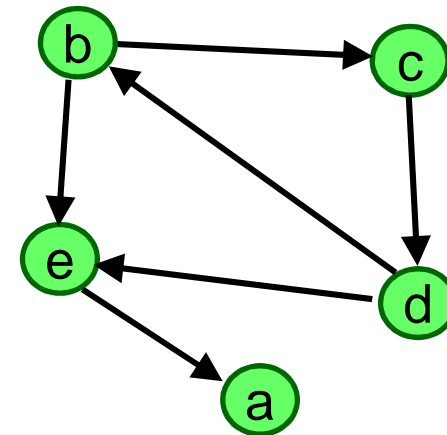
- 1. Encolar el vértice origen u .*
 - 2. Marcar el vértice u como visitado.*
 - 3. Procesar la cola.*
 - 4. Desencolar u de la cola*
 - 5. \forall adyacente a $u, (u, v) \in E$,*
 - 6. si v no ha sido visitado*
 - 7. encolar y visitar v*
- Si desde u no fueran alcanzables todos los nodos del grafo: volver a (1), elegir un nuevo vértice de partida no visitado, y repetir el proceso hasta que se hayan recorrido todos los vértices*
- Costo $T(|V|, |E|)$ es de $O(|V| + |E|)$*

Bosque de expansión del DFS

- El recorrido **no es único**: depende del nodo inicial y del orden de visita de los adyacentes.
- El orden de visita de unos nodos a partir de otros puede ser visto como un árbol: **árbol de expansión (o abarcador) en profundidad asociado al grafo**.
- Si aparecen varios árboles: **bosque de expansión (o abarcador) en profundidad**.

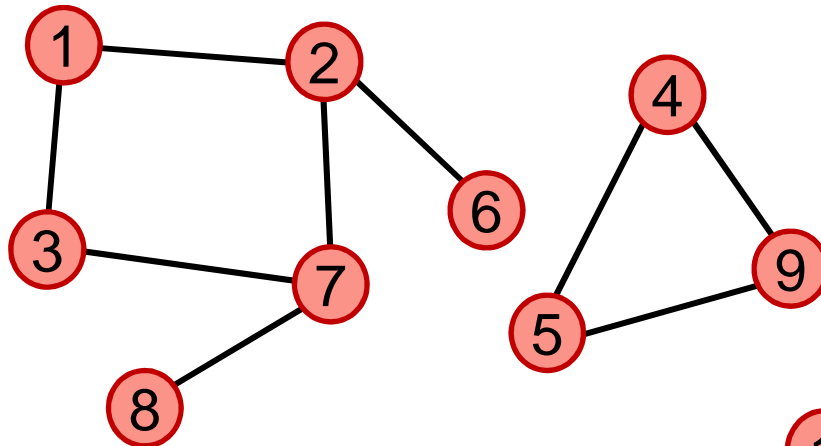


Grafo no dirigido y no Conexo

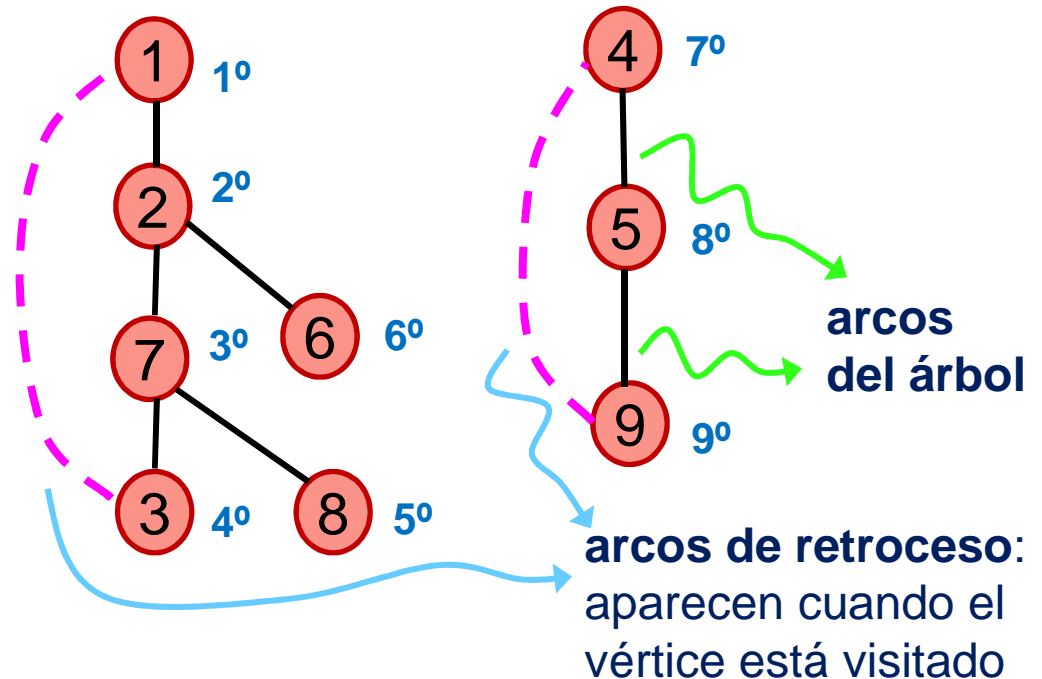


Grafo dirigido y no fuertemente Conexo

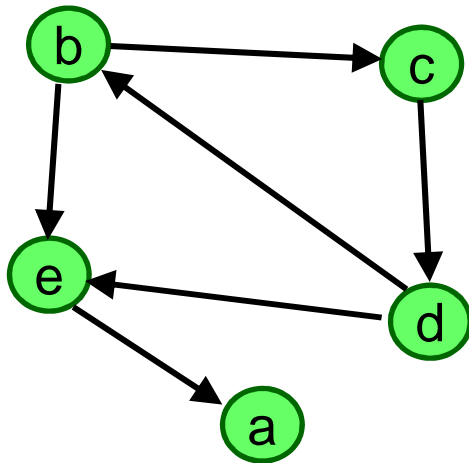
Bosque de expansión del DFS



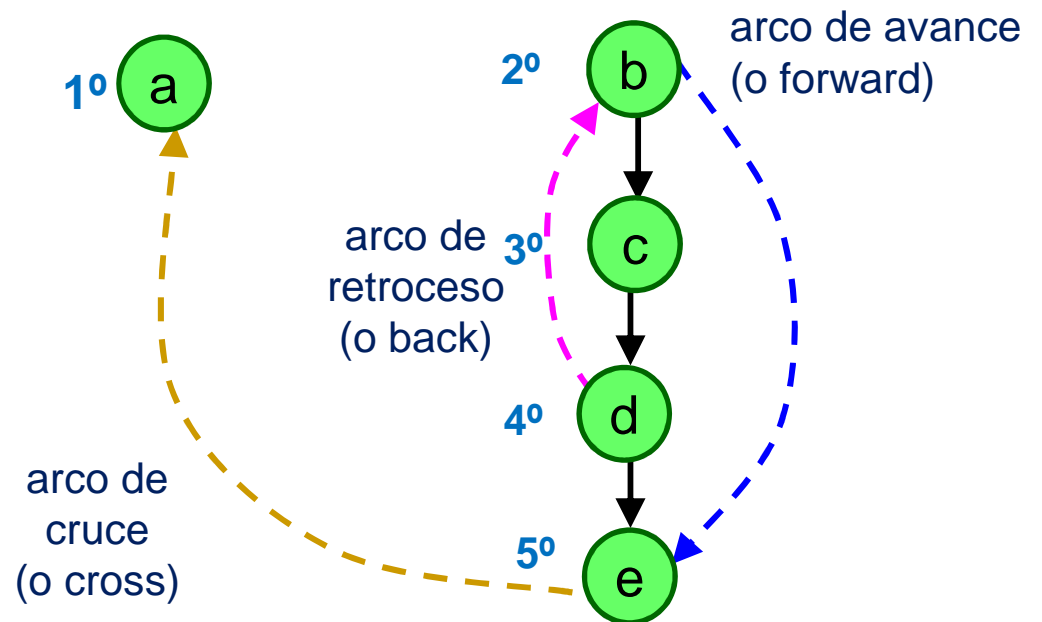
**Grafo no dirigido y no
Conexo**



Bosque de expansión del DFS



Grafo dirigido y no fuertemente Conexo



Bosque de expansión, empezando el recorrido en el vértice a



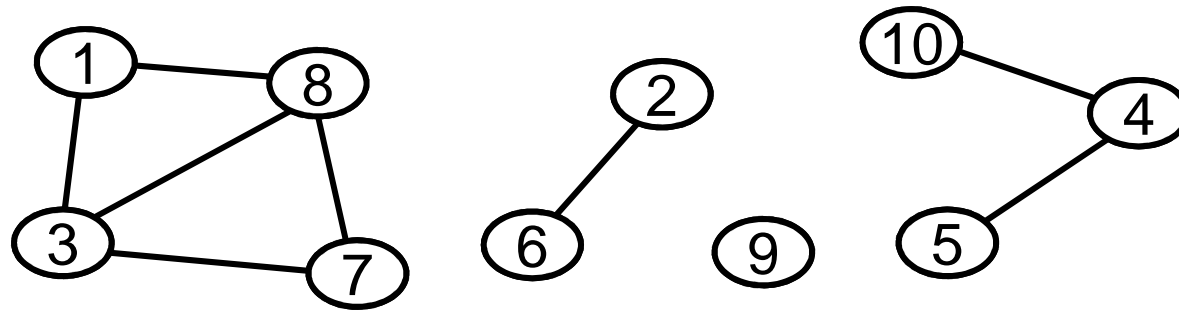
Bosque de expansión del DFS

Clasificación de los arcos de un grafo dirigido en el bosque de expansión de un DFS.

- Arcos **tree** (del **árbol**): son los arcos en el bosque depth-first-search, arcos que conducen a vértices no visitados durante la búsqueda.
- Arcos **forward**: son los arcos $u \rightarrow v$ que no están en el bosque, donde v es descendiente, pero no es hijo en el árbol.
- Arcos **backward**: son los arcos $u \rightarrow v$, donde v es antecesor en el árbol. Un arco de un vértice a si mismo es considerado un arco back.
- Arcos **cross**: son todos los otros arcos $u \rightarrow v$, donde v no es ni antecesor ni descendiente de u . Son arcos que pueden ir entre vértices del mismo árbol o entre vértices de diferentes árboles en el bosque depth-first-search

Aplicaciones del DFS

- **Problema 1:** encontrar las componentes conexas de un grafo no dirigido.



- **Problema 2: prueba de aciclicidad.** Dado un grafo (dirigido o no dirigido) comprobar si tiene algún ciclo o no.
- **Problema 3:** encontrar las componentes fuertemente conexas de un grafo dirigido.

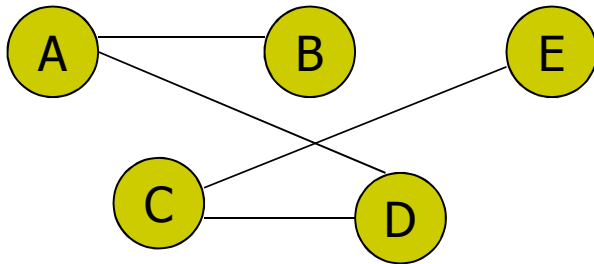


Aplicaciones del DFS

- **Problema 1: Encontrar las componentes conexas de un grafo no dirigido**
 - Si el grafo es conexo
 - Un recorrido desde cualquier vértice
 - Visitará a TODOS los vértices del grafo
 - Si no lo es
 - Partiendo de un vértice, tendremos una componente conexa
→ conjunto de vértices recorrido
 - Para descubrir otras
 - Repetir recorrido desde un vértice no visitado
 - Hasta que todos los vértices hayan sido visitados

Aplicaciones del DFS

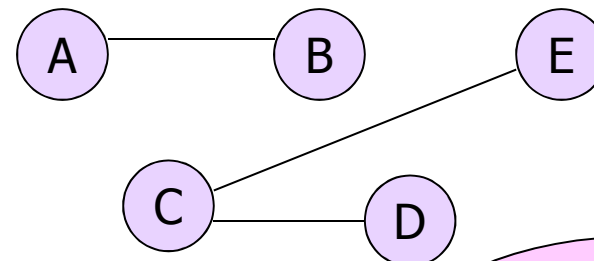
- **Problema 1: Encontrar las componentes conexas de un grafo no dirigido**



Recorrido desde E

E C D A B

Conjunto recorrido =
Conjunto de Vértices
Es CONEXO



Recorrido desde E

E C D

Componente
Conexa 1

Recorrido desde B

B A

Componente
Conexa 2



Aplicaciones del DFS

- **Problema 2: Prueba de aciclicidad**
 - **Grafo no dirigido.** Hacer un recorrido dfs. Existe algún ciclo si y sólo si aparece algún arco que no es del árbol de expansión.
 - **Grafo dirigido.** Hacer un dfs. Existe un ciclo si y sólo si aparece algún arco de retroceso.
- Orden de complejidad de la prueba de aciclicidad: igual que los recorridos.
 - Con matrices de adyacencia: $O(|V|^2)$.
 - Con listas de adyacencia: $O(|V| + |E|)$.



Aplicaciones del DFS

- **Problema 3: Encontrar las componentes fuertemente conexas**

Una aplicación clásica del depth-first search es descomponer un grafo dirigido en componentes fuertemente conexas (o conectadas).

Una *componente fuertemente conexa* de un grafo dirigido $G=(V,E)$ es el conjunto máximo de vértices $V' \subseteq V$ tal que para cada par de vértices u y v en V' , existe un camino tanto $u \rightarrow v$ como $v \rightarrow u$.



Aplicaciones del DFS

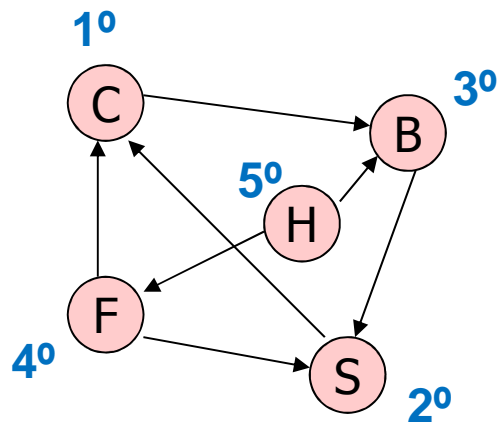
Encontrar las componentes fuertemente conexas de un grafo dirigido: Algoritmo de Kosaraju

Pasos:

1. Aplicar DFS(G) rotulando los vértices de G en post-orden (apilar).
2. Construir el grafo reverso de G , es decir G^R (invertir los arcos).
3. Aplicar DFS (G^R) comenzando por los vértices de mayor rótulo (tope de la pila).
4. Cada árbol de expansión resultante del paso 3 es una componente fuertemente conexa.

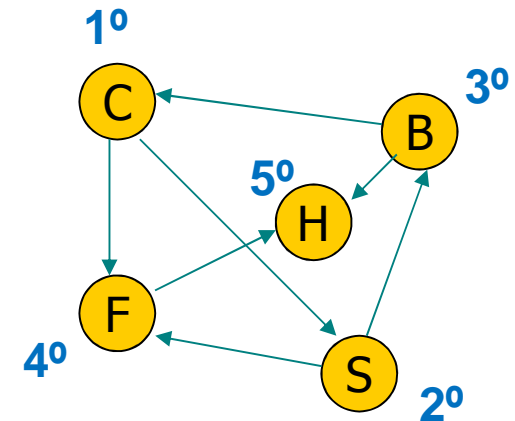
Si resulta un único árbol entonces el digrafo es fuertemente conexo.

Algoritmo de Kosaraju

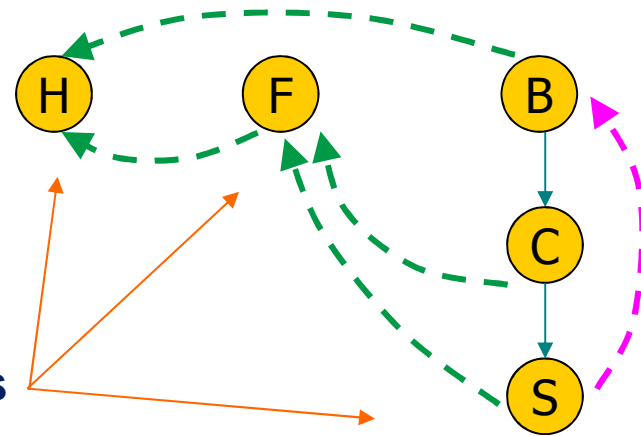


1. Aplicar el recorrido en profundidad, por ejemplo, desde **B** y rotular los vértices en post-orden

2. Construir el grafo reverso G^R
3. Aplicar DFS comenzando de los vértices de mayor rótulo en G^R



Componentes fuertemente conexos





Algoritmo de Kosaraju

Complejidad del algoritmo

- Se realizan dos DFS
- Se recorren todas las aristas una vez para crear el grafo reverso



$$O(|V| + |E|)$$