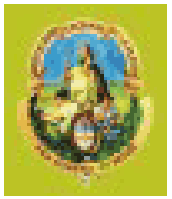


Algoritmos y Estructuras de Datos

Cursada 2011

Prof. Catalina Mostaccio
Prof. Alejandra Schiavoni

Facultad de Informática - UNLP

GRAFOS

Agenda - Grafos

1. Ejemplos y terminología
2. Representaciones
3. Recorridos
4. Sort topológico

Agenda - Grafos

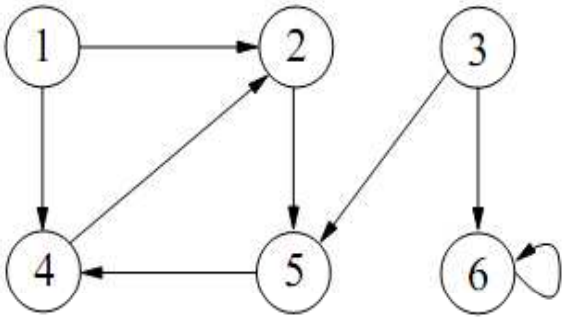
1. Ejemplos y terminología
2. Representaciones
3. Recorridos
4. Sort topológico

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 ∈ V , llamados aristas o arcos.*
- ***Grafo dirigido**: la relación sobre V no es simétrica. Arista ≡ par ordenado (u,v).*
- ***Grafo no dirigido**: la relación sobre V es simétrica. Arista ≡ par no ordenado {u,v}, u,v ∈ V y u ≠ v*

Terminología (cont. 1)

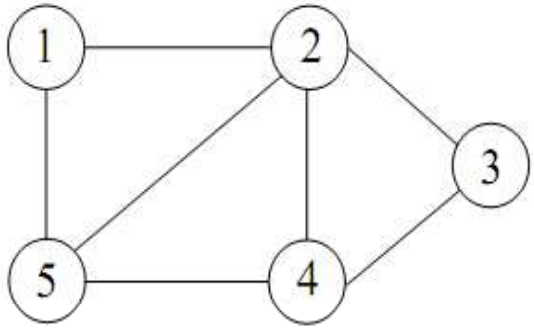
Ejemplo 1



Grafo dirigido G(V , E).

V = {1,2,3,4,5,6}

E={(1,2),(1,4),(2,5),(3,5),(3,6), (4,2), (5,4),(6,6)}



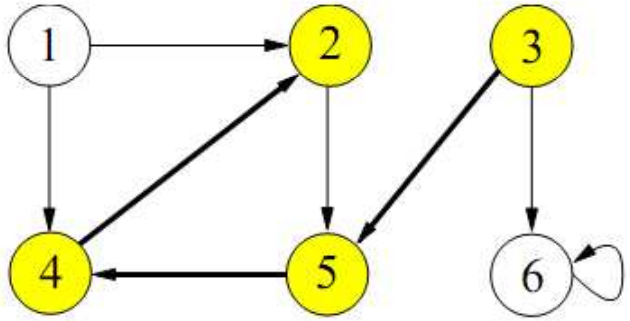
Grafo no dirigido G(V , E).

V = {1,2,3,4,5}

E= {{1,2},{1,5},{2,3},{2,4}, {2,5},{3,4},{4,5}}

Terminología (cont. 2)

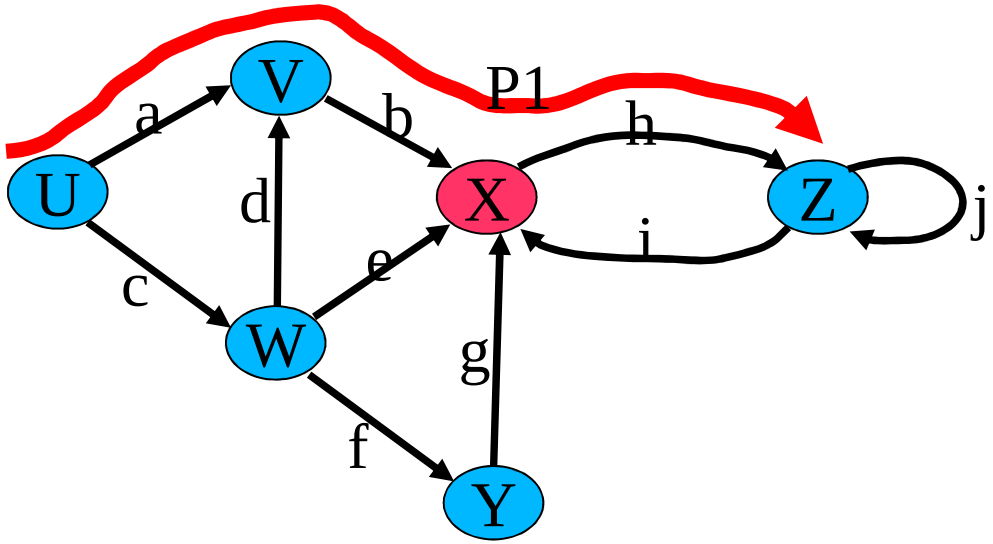
- ***Camino*** *desde u ∈ V a v ∈ V : secuencia* $v_1, v_2, \ldots, v_k$ *tal que* $u=v_1$, $v=v_k$, *y* $(v_{i-1},v_i) \in E$, *para* $i = 2, \ldots, k$.
  Ej: camino desde 3 a 2 → <3,5,4,2>.



- ***Longitud de un camino****: número de arcos del camino.*
  Ejs: long. del camino desde 3 a 2 → <3,5,4,2> es 3. (a)
  long. del camino desde 3 a 4 → <3,5,4,2,5,4> es 4. (b)

Terminología (cont. 3)

- ***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.*

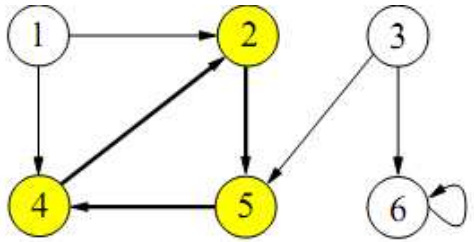


Ejemplos anteriores: (a) es camino simple , (b) no lo es.

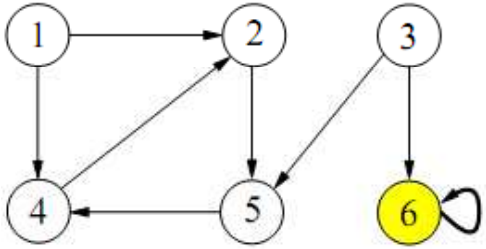
Terminología (cont. 4)

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

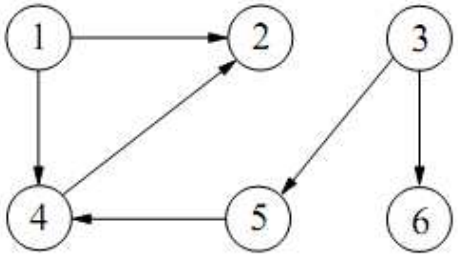
  Ej: <2,5,4,2> es un ciclo de longitud 3.



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



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



Terminología (cont. 5)

- *Un grafo es **conexo** si entre cada dos nodos hay un camino.*
- *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.*

Terminología (cont. 6)

- *v es **adyacente** a u si existe una arista (u,v) ∈ E.*
  - *en un grafo no dirigido, (u,v) ∈ E **incide** en los nodos u, v.*
  - *en un grafo dirigido, (u,v) ∈ 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. 7)

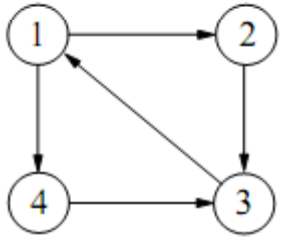
- *Sea G un grafo no dirigido con **n** vértices y **m** arcos, entonces*

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

- ✓ *Siempre:* $m \leq (n*(n-1))/2$
- ✓ *Si G conexo:* $m \geq n-1$
- ✓ *Si G árbol:* $m = n-1$
- ✓ *Si G bosque:* $m \leq n-1$

Terminología (cont. 8)

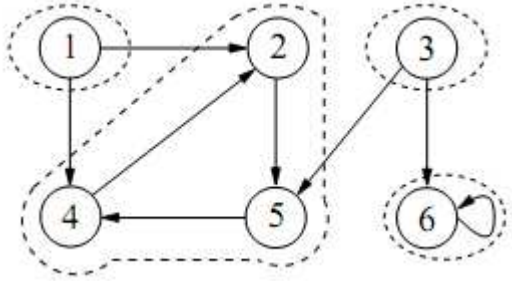
- *v es **alcanzable desde** u, si existe un camino de u a v.*
- *Un grafo no dirigido es **conexo** si existe un camino desde cualquier vértice a cualquier otro.*
- *Un grafo dirigido con esta propiedad se denomina **fuertemente conexo**:*



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

Terminología (cont. 9)

- *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.*
- *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.*



Terminología (cont. 10)

- *G' = (V', E') es un **subgrafo** de G = (V , E) si V'⊆V y E' ⊆E.*
- ***Subgrafo inducido** por V' ⊆ V : G' = (V',E') tal que E' = {(u,v) ∈ E | u,v ∈ V'}.*

Ejemplos de Subgrafos del grafo de la Fig. a

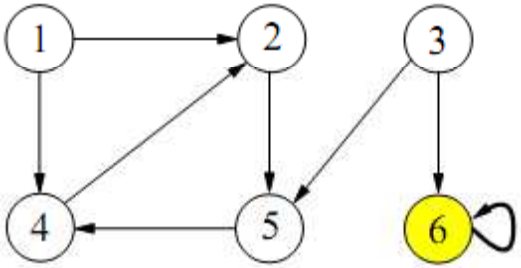


Fig. a

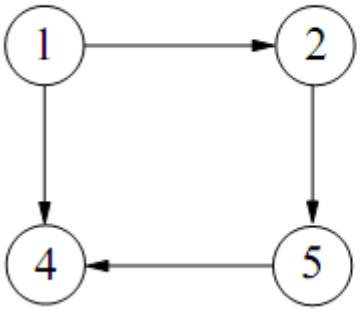


Fig. b

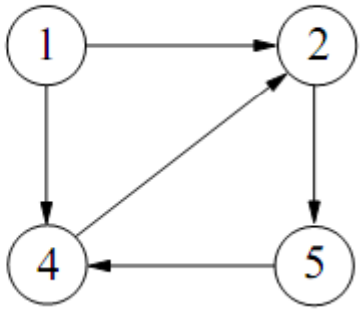


Fig. c

Fig. b : Subgrafo

V'={1,2,4,5}

E'={(1,2),(1,4),(2,5),(5,4)}

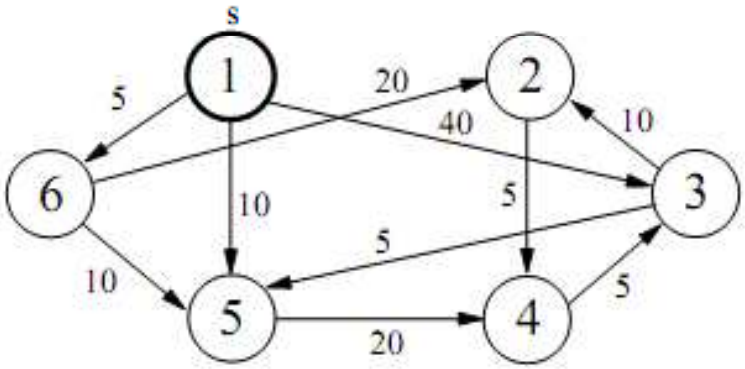
Fig. c : Subgrafo inducido por

V'={1,2,4,5}

E'={(1,2),(1,4),(2,5),(4,2),(5,4)}

Terminología (cont. 11)

- *Un grafo **ponderado, pesado o con costos**: cada arco o arista tiene asociado un valor o etiqueta.*



Agenda - Grafos

1. Ejemplos y terminología
2. Representaciones
3. Recorridos
4. Sort topológico

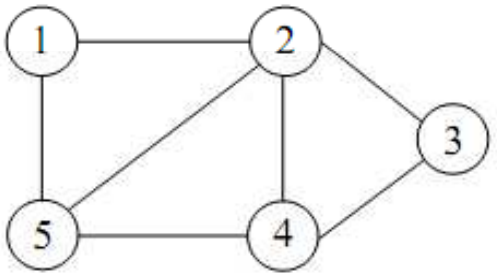
Agenda - Grafos

- Representaciones
  - Matriz de Adyacencias
  - Lista de Adyacencias

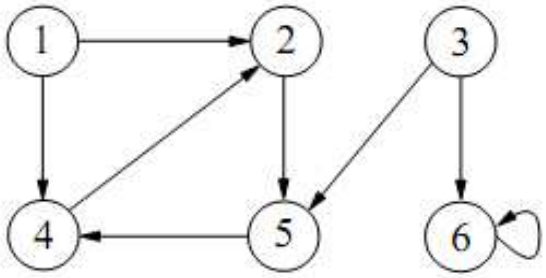
Representaciones: Matriz de Adyacencias

- *G =(V ,E): matriz A de dimensión |V |×|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

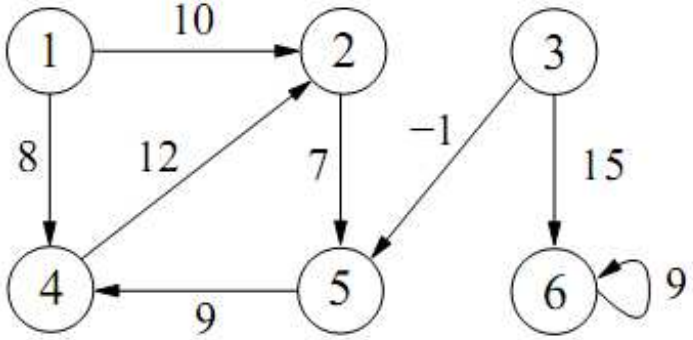
Representaciones: Matriz de Adyacencias

- *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 → consultar posición A(u,v) con 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).*

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

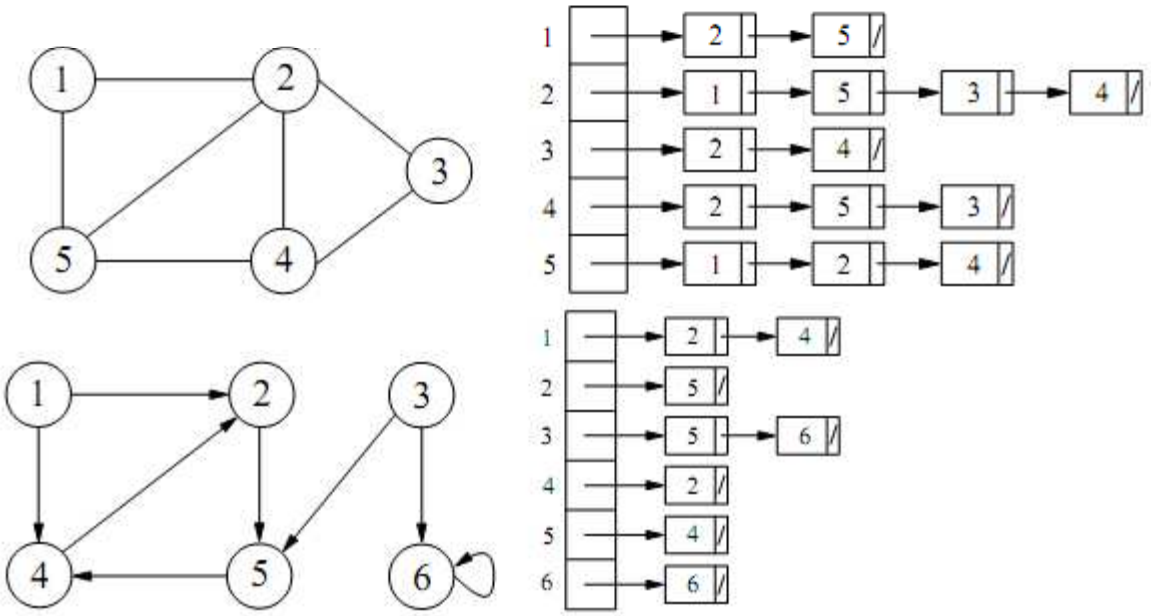


	1	2	3	4	5	6
1	0	10	0	8	0	0
2	0	0	0	0	7	0
3	0	0	0	0	−1	15
4	0	12	0	0	0	0
5	0	0	0	9	0	0
6	0	0	0	0	0	9

Representaciones: Lista de Adyacencias

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

  *Los elementos de la lista son los vértices adyacentes a **i***

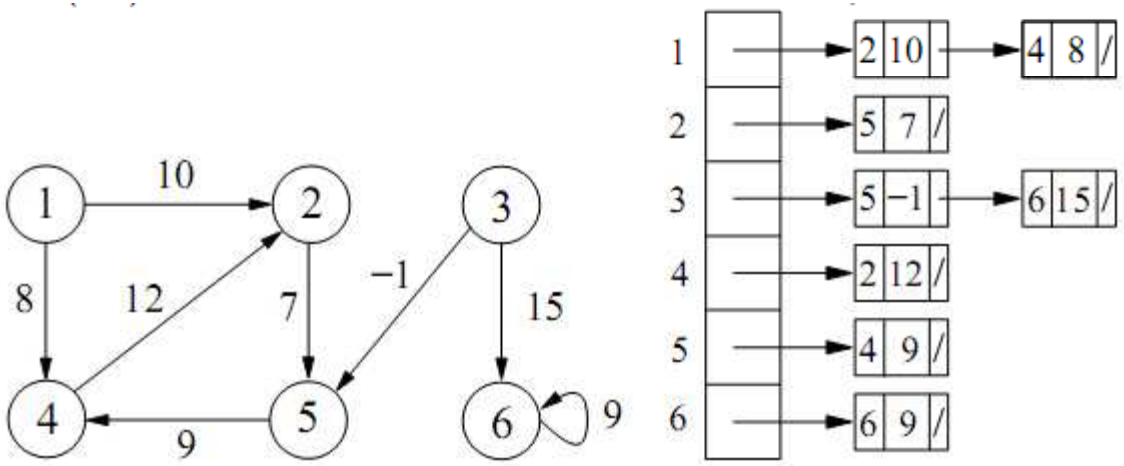


Representaciones: Lista de Adyacencias

- *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 (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
4. Sort topológico

Agenda - Grafos

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

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, seleccionar alguno como nuevo vértice de partida, y repetir el proceso.*

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. ∀ **v** adyacente a **u**,**(u,v)** ∈ E, si **v** no ha sido visitado, repetir recursivamente (3) y (4) para **v**.*

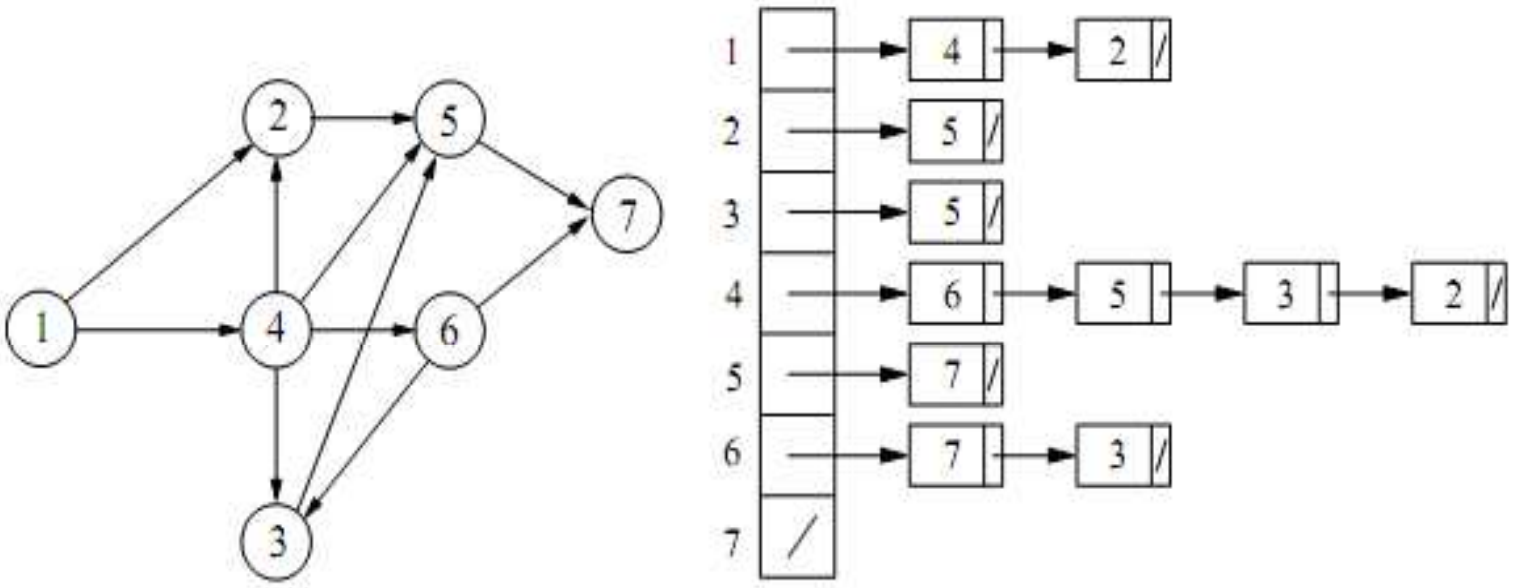
• Finalizar cuando se hayan visitado todos los nodos alcanzables desde u.

*• 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

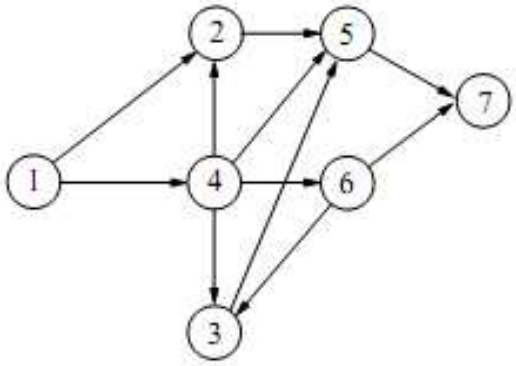
```
operación  dfs (v: nodo)
    marca[v]:= visitado
    para cada nodo w adyacente a v hacer
        si marca[w] == noVisitado entonces
                dfs(w)
    finpara

 operación main_dfs
          var  marca: array [1, ..., n] de (visitado, noVisitado)
           para i:= 1, ..., n hacer
               marca[i]:= noVisitado
           para v:= 1, ..., n hacer
               si marca[v] == noVisitado entonces
                dfs(v)
           finpara
```

Recorrido DFS : Ejemplo

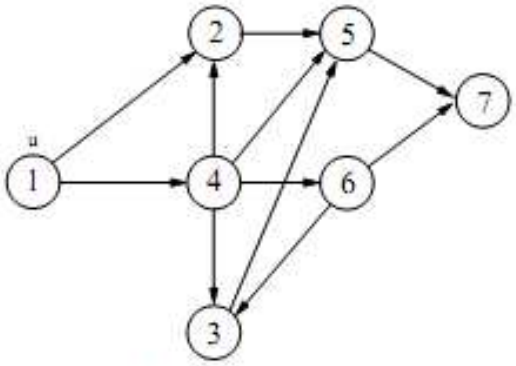


☞ *¡OJO!: el recorrido depende del orden en que aparecen los vértices en las listas de adyacencia.*

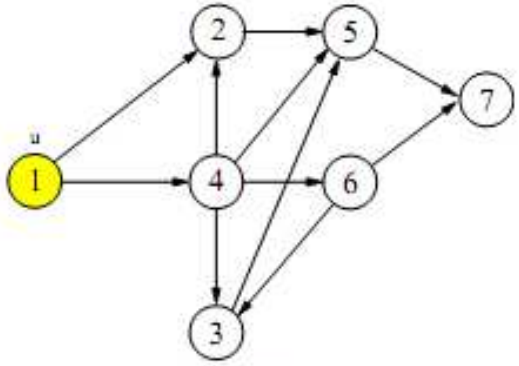
Recorrido DFS : Ejemplo (cont. 1)



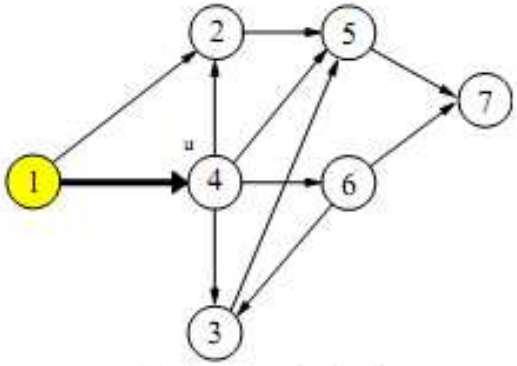
Rec_en_profund(G)



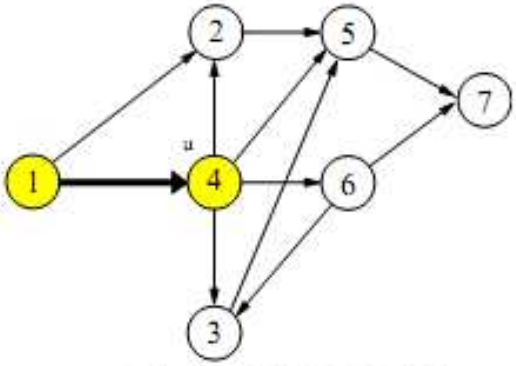
Visita_nodo(1)



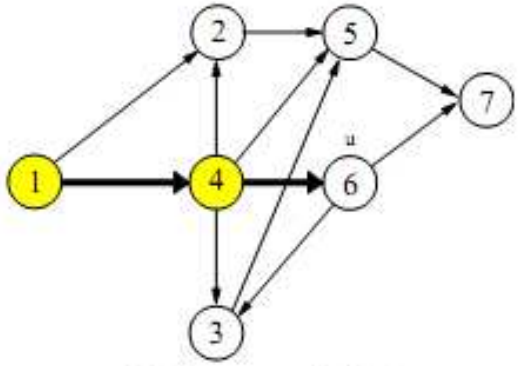
color[1]=AMARILLO



Visita_nodo(4)

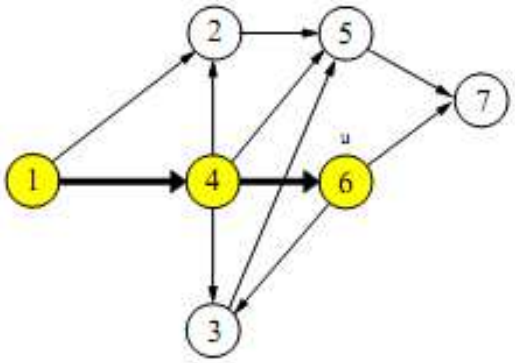


color[4]=AMARILLO

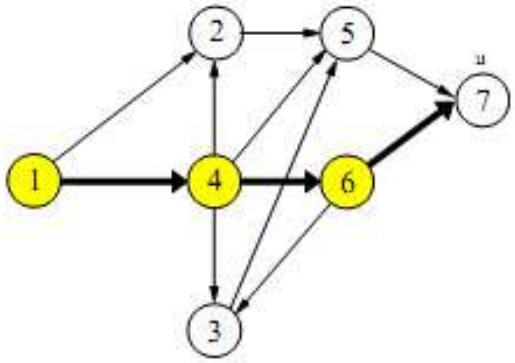


Visita_nodo(6)

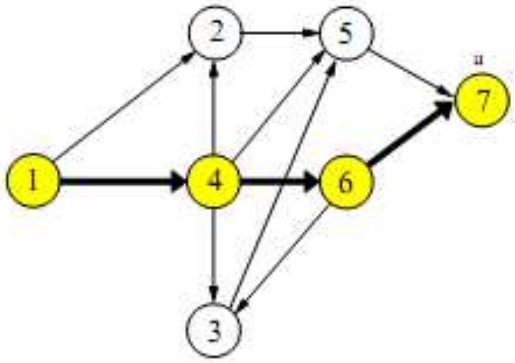
Recorrido DFS : Ejemplo (cont. 2)



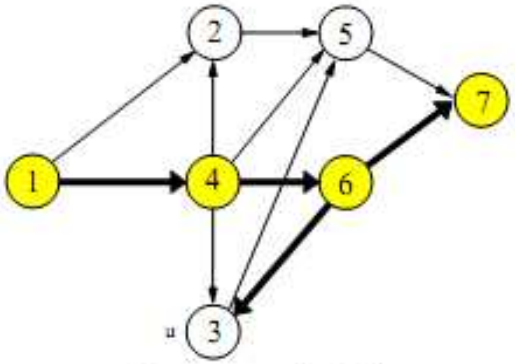
color[6]=AMARILLO



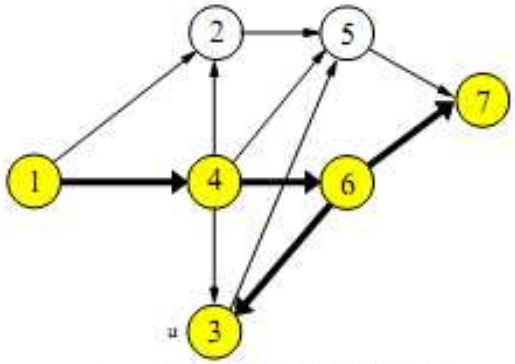
Visita_nodo(7)



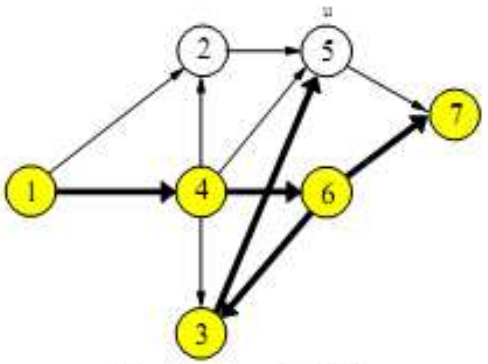
color[7]=AMARILLO



Visita_nodo(3)

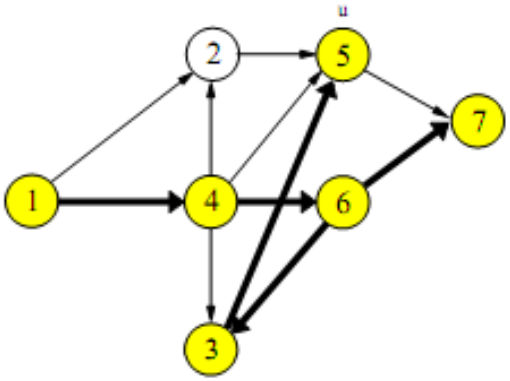


color[3]=AMARILLO

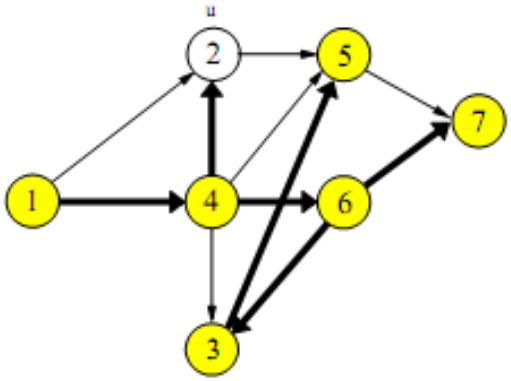


Visita_nodo(5)

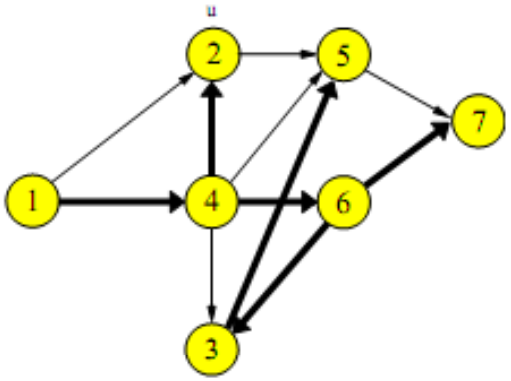
Recorrido DFS : Ejemplo (cont. 3)



color[5]=AMARILLO



Visita_nodo(2)



color[2]=AMARILLO

Recorrido DFS : Tiempo de ejecución

- *G (V , E) se representa mediante listas de adyacencia.*
- Visita_nodo *se aplica únicamente sobre vértices no visitados* → *sólo una vez sobre cada vértice.*
- Visita_nodo *depende del número de vértices adyacentes que tenga* ***u*** *(longitud de la lista de adyacencia).*
- *el tiempo de todas las llamadas a* Visita_nodo *:*

$$\sum_{v \in V} |\mathsf{ady}(v)| = \Theta(|E|)$$

- *añadir el tiempo asociado a los bucles de* Recorrido_en _profundidad*: O (|V|)).*

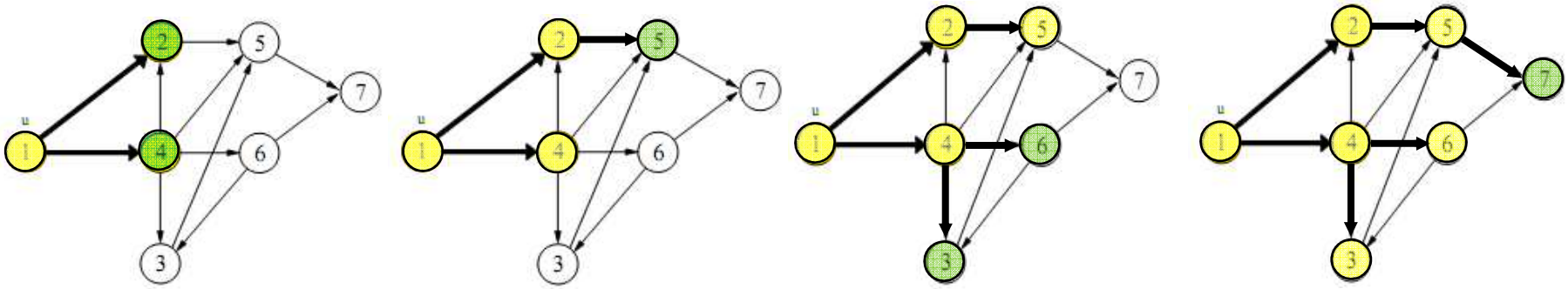
⇒ *Tiempo del recorrido en profundidad es O (|V|+|E|) .*

Recorrido en amplitud : BFS

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

***Estrategia**:*

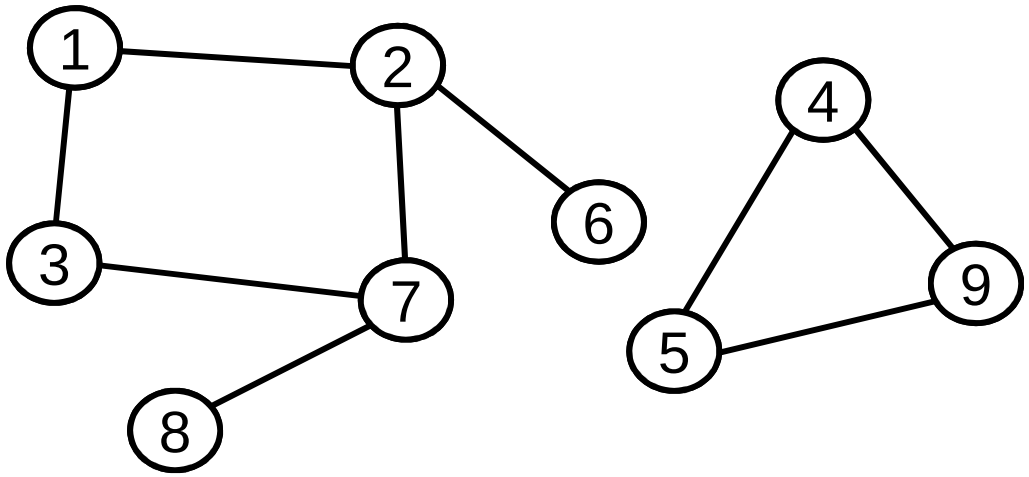
- *Partir de algún vértice **u**, visitar **u** y, después, visitar cada uno de los vértices adyacentes a **u**.*
- *Repetir el proceso para cada nodo adyacente a **u**, siguiendo el orden en que fueron visitados.*
- *Costo T(|V|,|E|) es de O(|V|+|E|).*



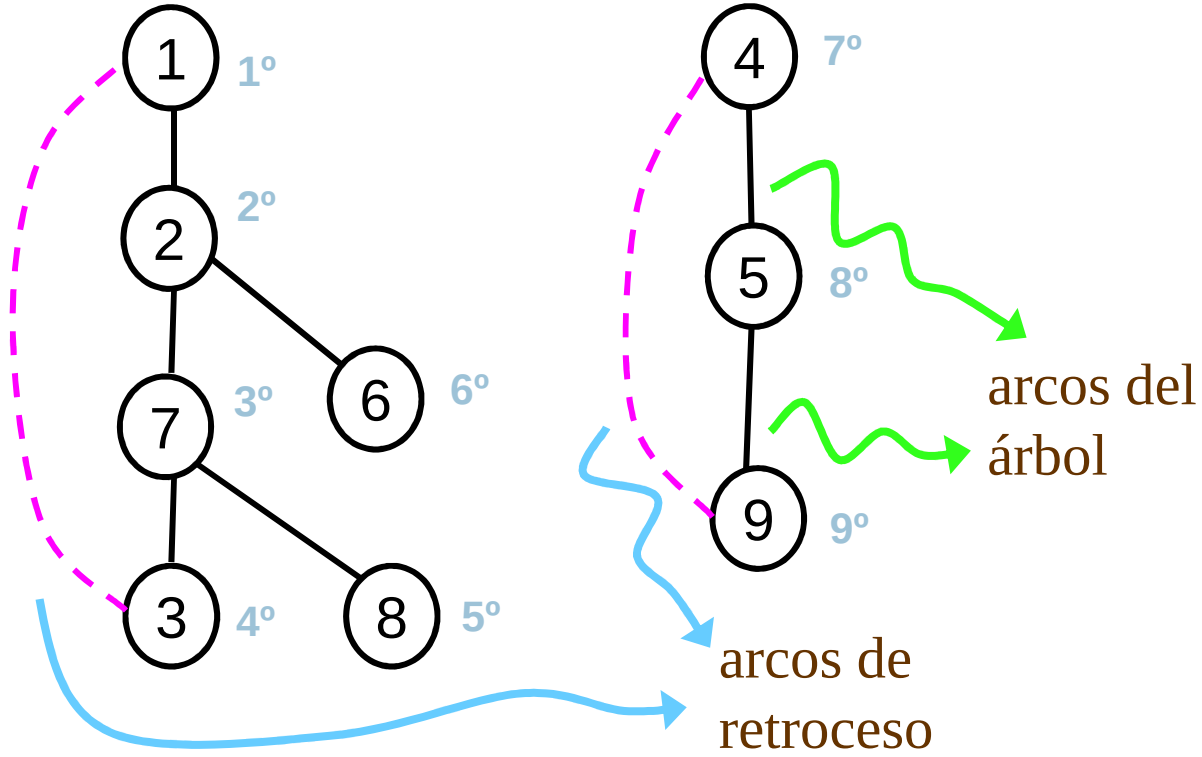
Bosque de expansión del DFS

- *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****.*

- *Ejemplo. Grafo no dirigido.*



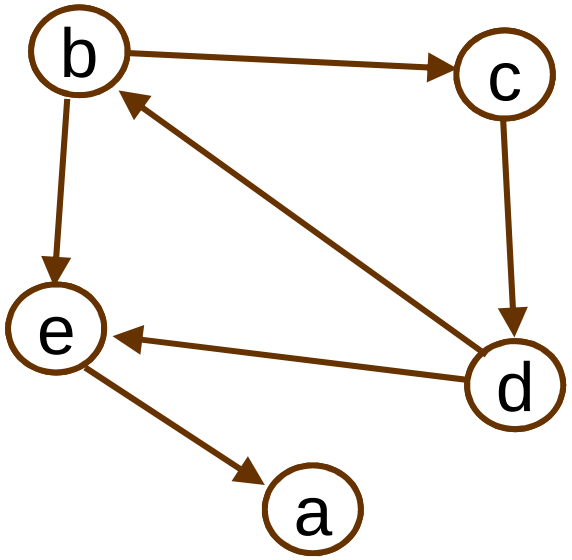
Bosque de expansión del DFS

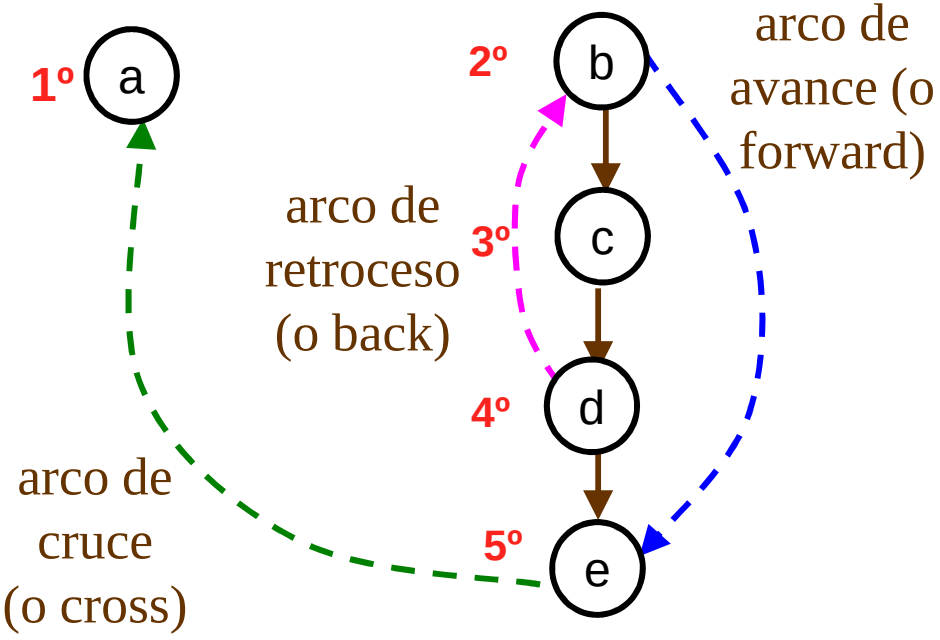


- **Arcos de retroceso:** si marca[v] == noVisitado ...
- → se detectan cuando la condición es falsa.

Bosque de expansión del DFS

- ***Ejemplo:*** *grafo dirigido.*





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, arcos que conducen a vértices no visitados durante la búsqueda.*

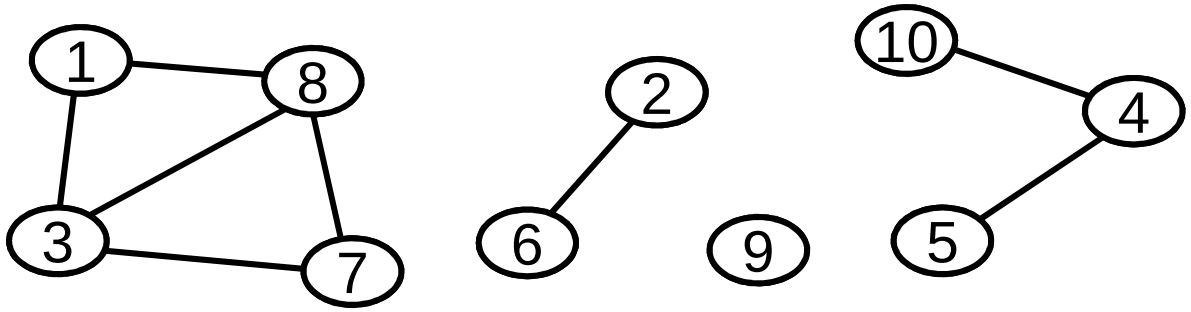
*• Arcos **forward**: son los arcos u→v que no están en el bosque, donde v es descendiente, pero no es hijo en el árbol.*

*•Arcos **backward**: son los arcos u→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→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:**
  - 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 2: Prueba de aciclicidad**
  - **Grafo no dirigido.** Hacer un dfs (o bfs). Existe algún ciclo si y sólo si aparece algún arco que no es del árbol de expansión.
  - **Grafo dirigido.** Hacer un dfs (o bfs). 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: 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,A) 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

Algoritmo para encontrar las Componentes Fuertemente Conexas

Pasos:

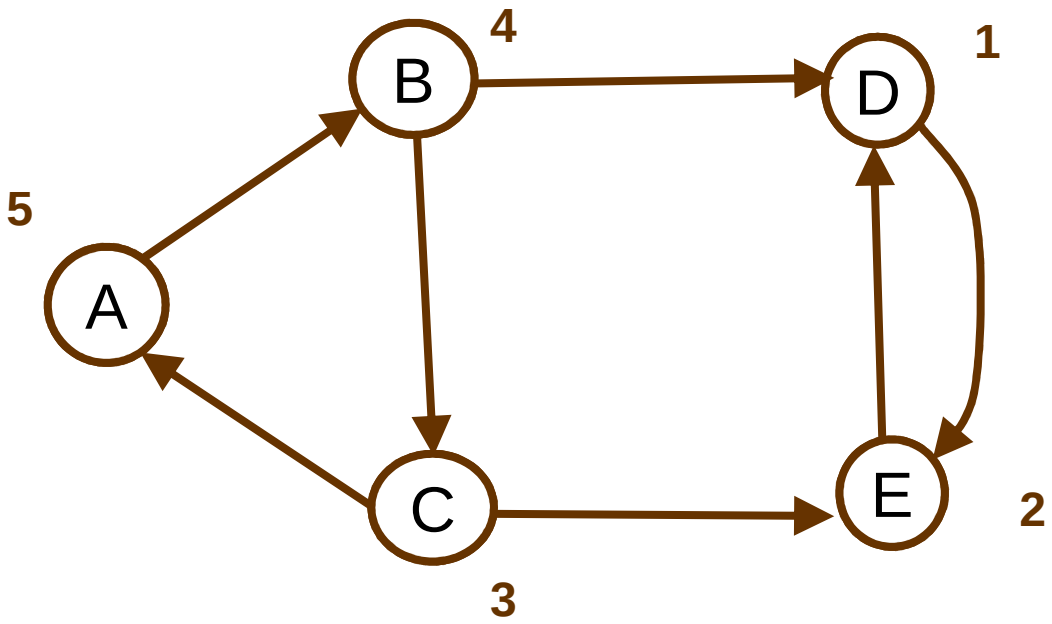
1. Aplicar DFS(G) rotulando los vértices de G en post-orden (apilar).
2. Construir el grafo reverso de G, es decir G^R (invertir los arcos).
3. Aplicar DFS (G^R) comenzando por los vértices de mayor rótulo (tope de la pila).
4. Cada árbol de expansión resultante del paso 3 es una componente fuertemente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

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

Aplicaciones del DFS

Algoritmo para encontrar las Componentes Fuertemente Conexas (cont.)

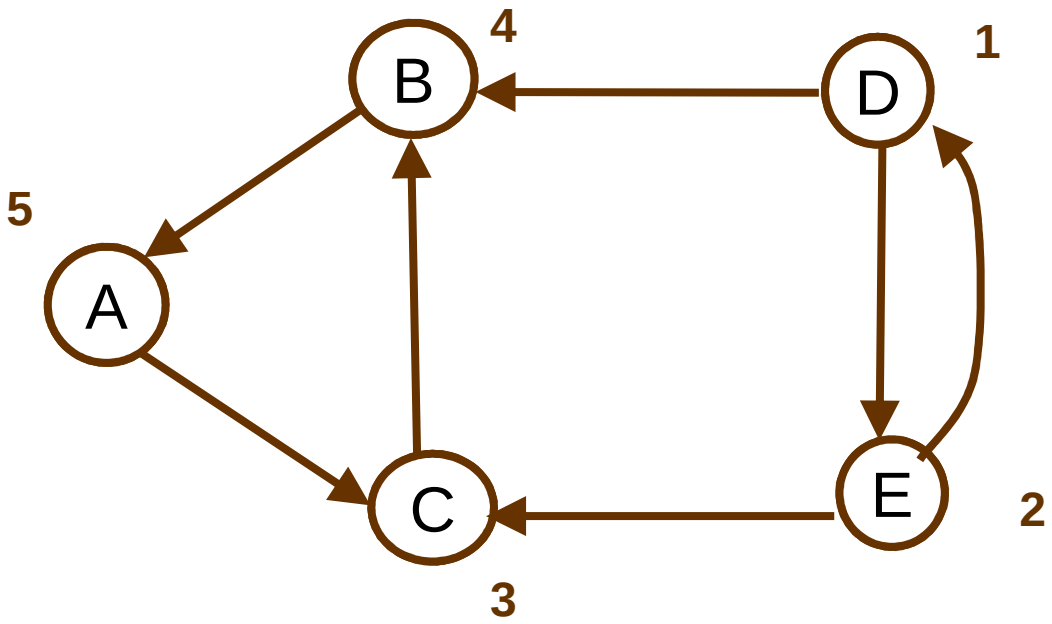
Paso 1: Aplicar DFS a partir de A, y numerar los vértices en post-orden



Aplicaciones del DFS

Algoritmo para encontrar las Componentes Fuertemente Conexas (cont.)

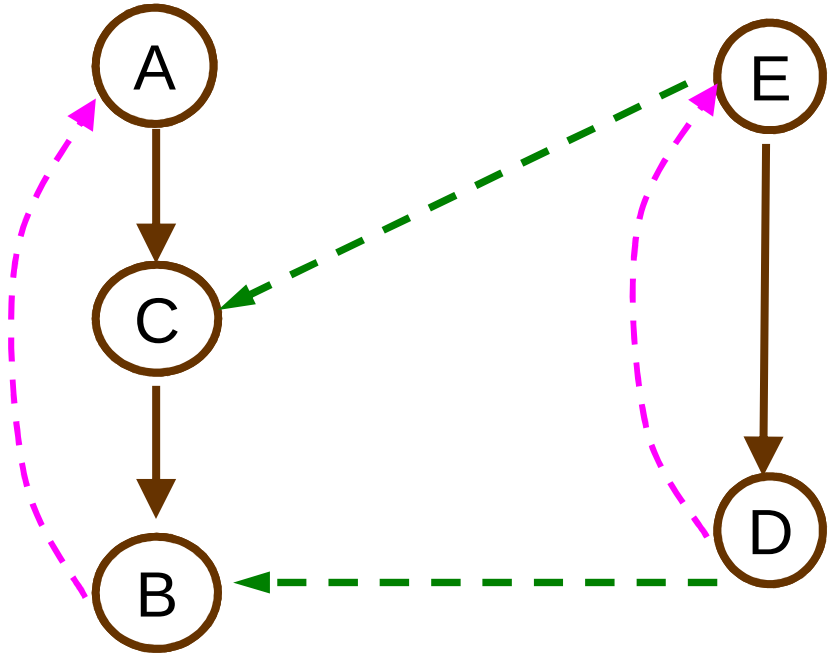
Paso 2: Construir el grafo reverso de G, es decir G^R (invertir los arcos).



Aplicaciones del DFS

Algoritmo para encontrar las Componentes Fuertemente Conexas (cont.)

Paso 3: Aplicar DFS (G^R) comenzando por los vértices de mayor numeración



Componentes fuertemente conexas: {A, B, C}, {E, D}