

Diseño y Análisis de Algoritmos

TEMA 4. ALGORITMOS EN GRAFOS



Universidad
Rey Juan Carlos

Tema 4. Exploración de grafos

Introducción

- Definiciones
- Recorrido en profundidad
- Recorrido en anchura
- Algoritmos sobre grafos
 - Recorridos de grafos infinitos
 - Ordenación topológica
 - Componentes fuertemente conexas
 - Puntos de articulación

Tema 4. Exploración de grafos

Definiciones

- Se suele **definir** un grafo $G=(V,E)$ como un conjunto de vértices V y de aristas $E \subseteq V \times V$.
- Usualmente, la **complejidad** de los algoritmos sobre grafos suele medirse en función del
 - número de vértices: $|V|$
 - número de aristas: $|E|$
- *Convenios de notación:* Cuando nos refiramos a estas cantidades en notación asintótica, sustituiremos $|V|$ y $|E|$ por V y E .

Ejemplo: $O(VE)$ en lugar de $O(|V||E|)$

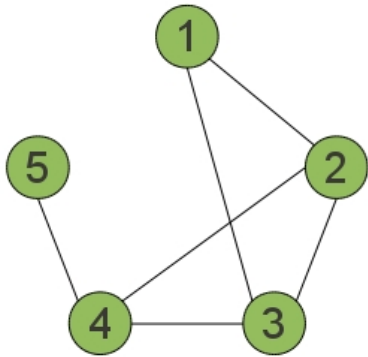
Tema 4. Exploración de grafos

Definiciones

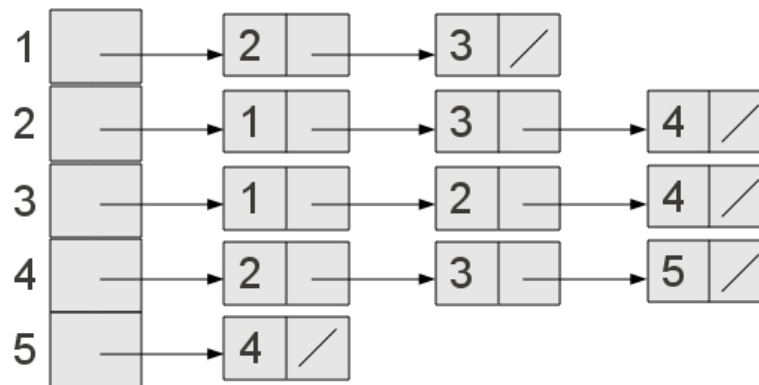
- Existen dos **representaciones** típicas de grafos:
 - Matriz de adyacencia
 - Lista de adyacencia

Grafo no dirigido:

Grafo



Lista de adyacencias



Matriz de adyacencias

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

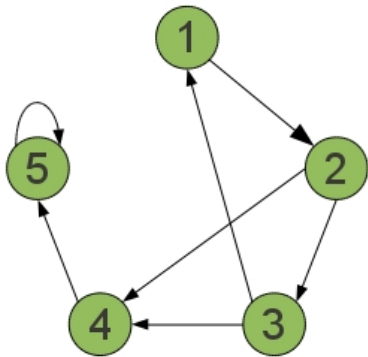
Tema 4. Exploración de grafos

Definiciones

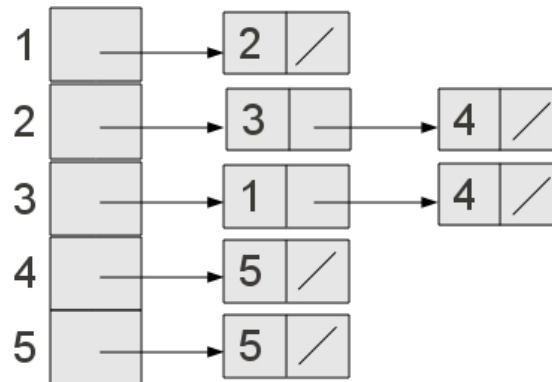
- Existen dos **representaciones** típicas de grafos:
 - Matriz de adyacencia
 - Lista de adyacencia

Grafo dirigido:

Grafo



Lista de adyacencias



Matriz de adyacencias

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

Tema 4. Exploración de grafos

Introducción

Lista de adyacencia:

- Representación **compacta** para grafos dispersos ($|E| \ll |V|^2$)
- **No aseguran** un acceso rápido a la hora de comprobar si hay una arista entre dos vértices dados

Matriz de adyacencia:

- **Aseguran** un acceso rápido a la hora de comprobar si hay una arista entre dos vértices dados
- Se requiere una **memoria** de $\Theta(V^2)$, y no depende de la densidad del grafo
- Para grafos **densos** (aquéllos en los que $|E| \cong |V|^2$) ya no se desperdicia tanta memoria

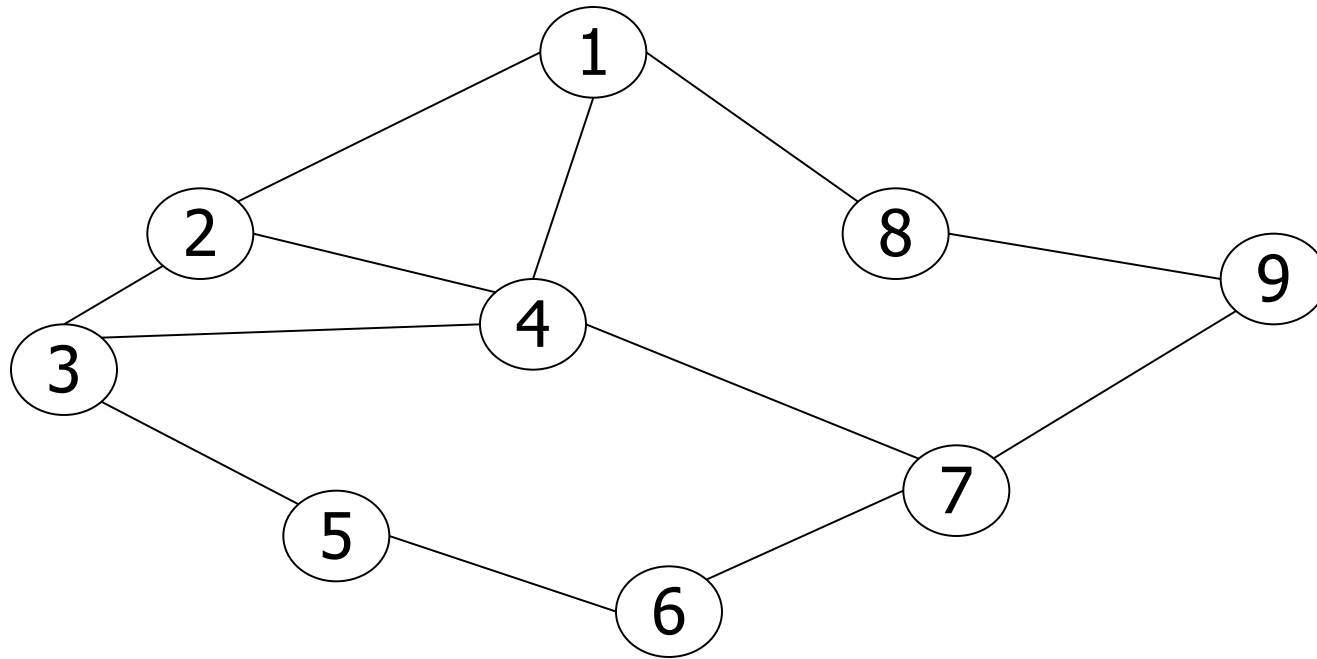
Tema 4. Exploración de grafos

Recorrido en profundidad

- El recorrido en profundidad (*depth-first search*) tiene como estrategia **profundizar** en el grafo siempre que sea posible
 - Dado un vértice, antes de **visitar** su *hermano* se visita a su *hijo* (equivalente a un recorrido preorden)
 - Suele implementarse de manera **recursiva**
 - Se tiene que incluir un **conjunto** de vértices visitados para evitar **ciclos** en la búsqueda

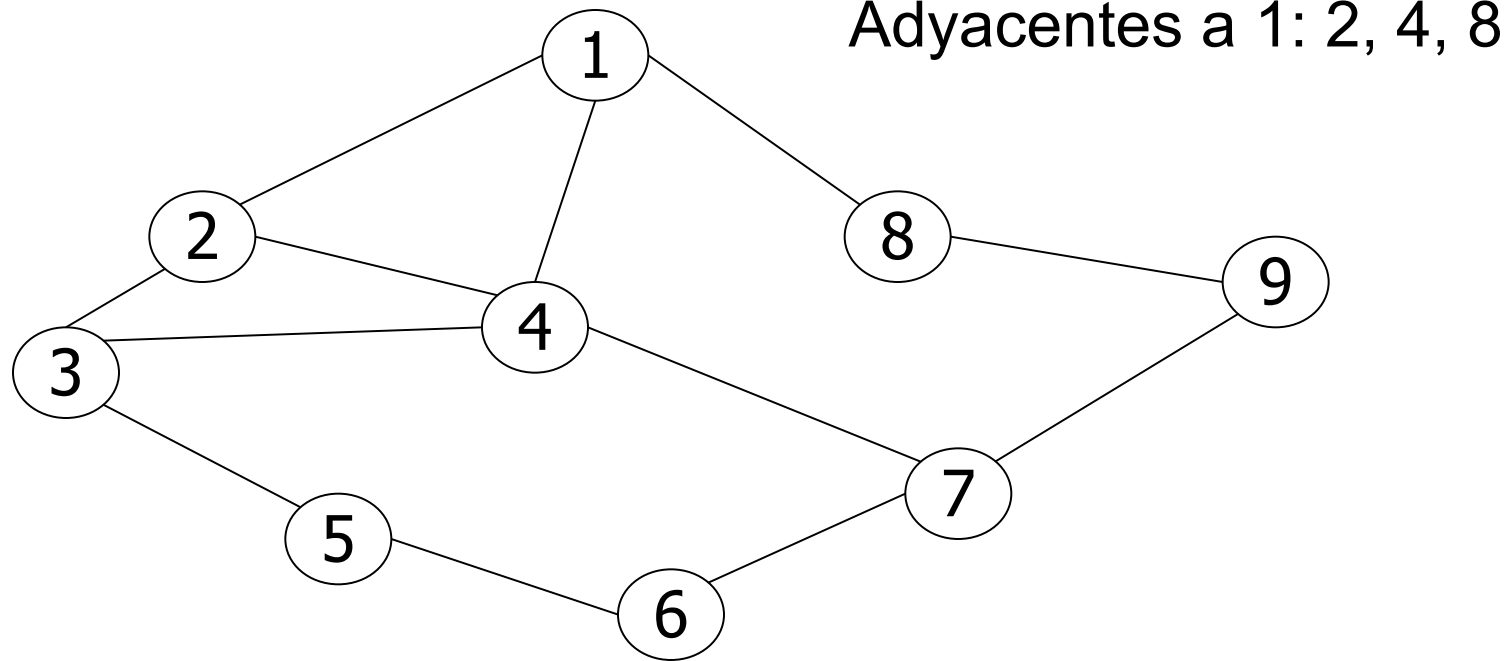
Tema 4. Exploración de grafos

Recorrido en profundidad



Tema 4. Exploración de grafos

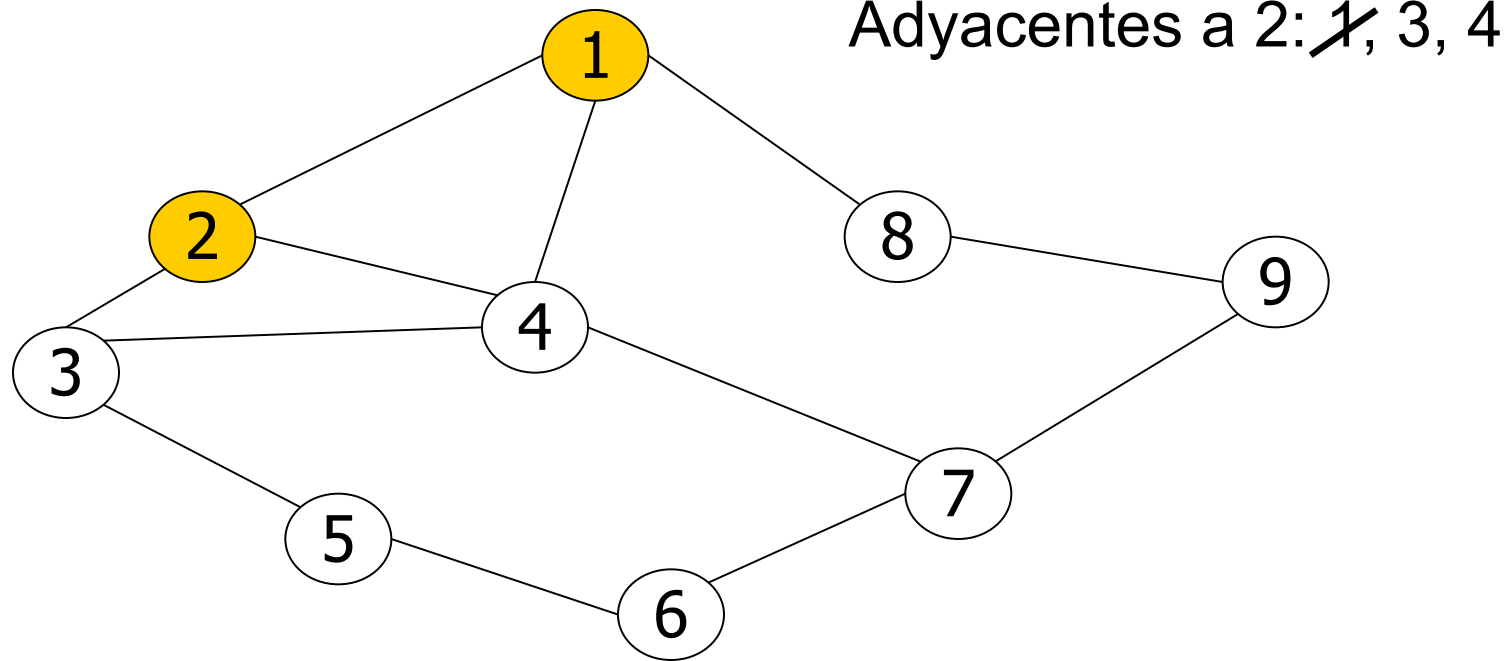
Recorrido en profundidad



Recorrido: 1, 2

Tema 4. Exploración de grafos

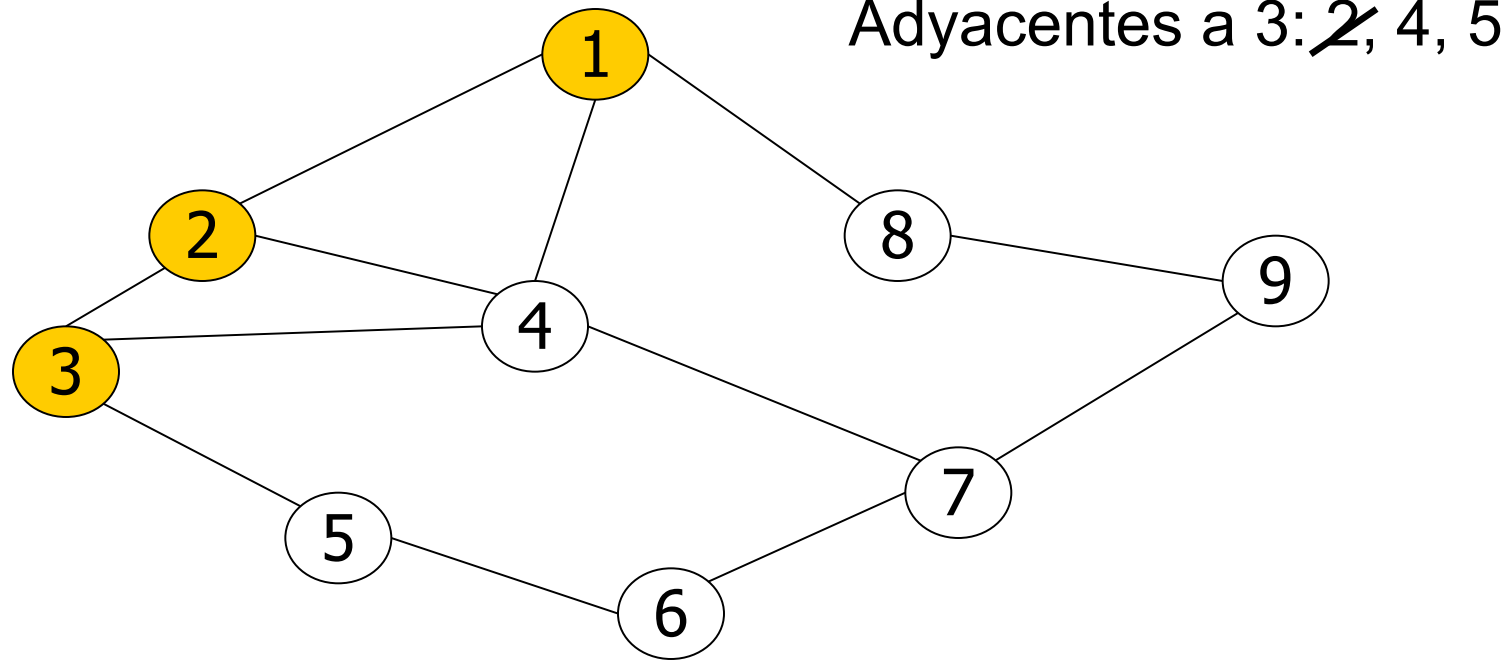
Recorrido en profundidad



Recorrido: 1, 2, 3

Tema 4. Exploración de grafos

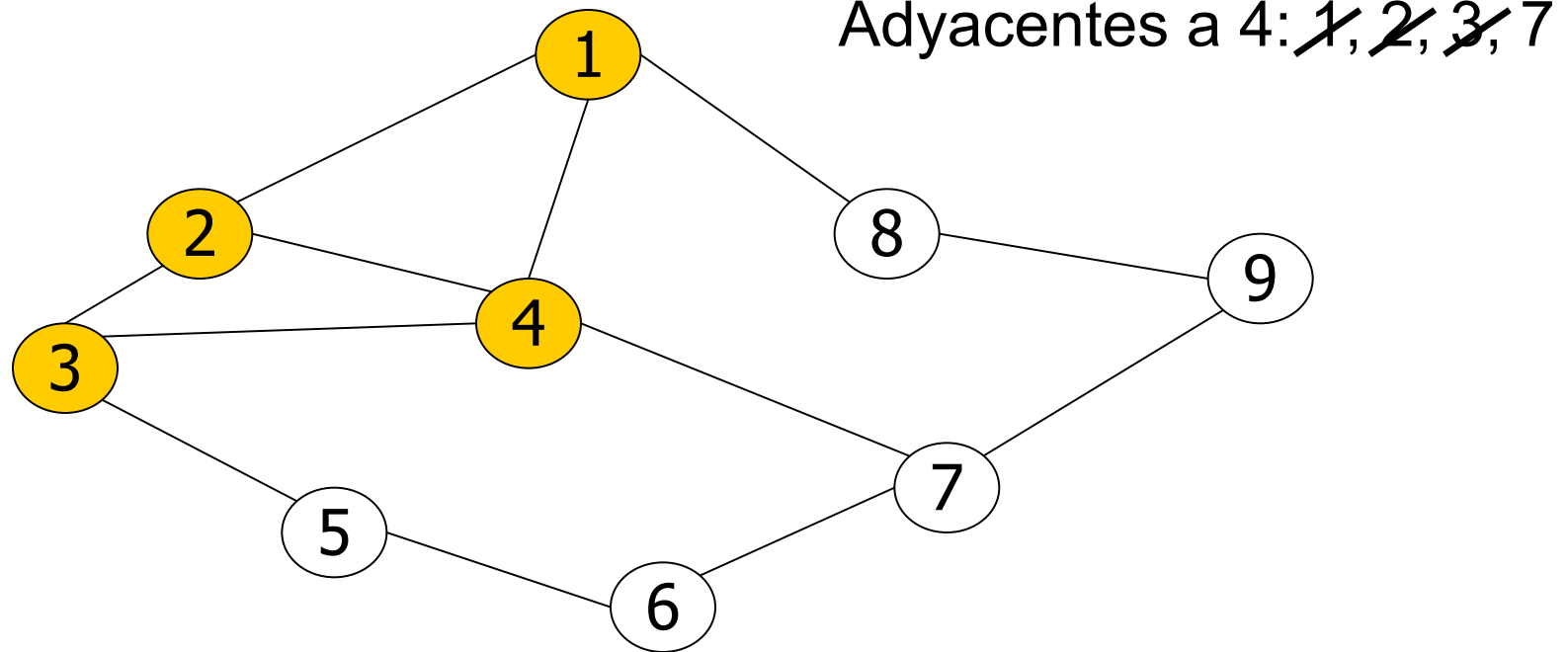
Recorrido en profundidad



Recorrido: 1, 2, 3, 4

Tema 4. Exploración de grafos

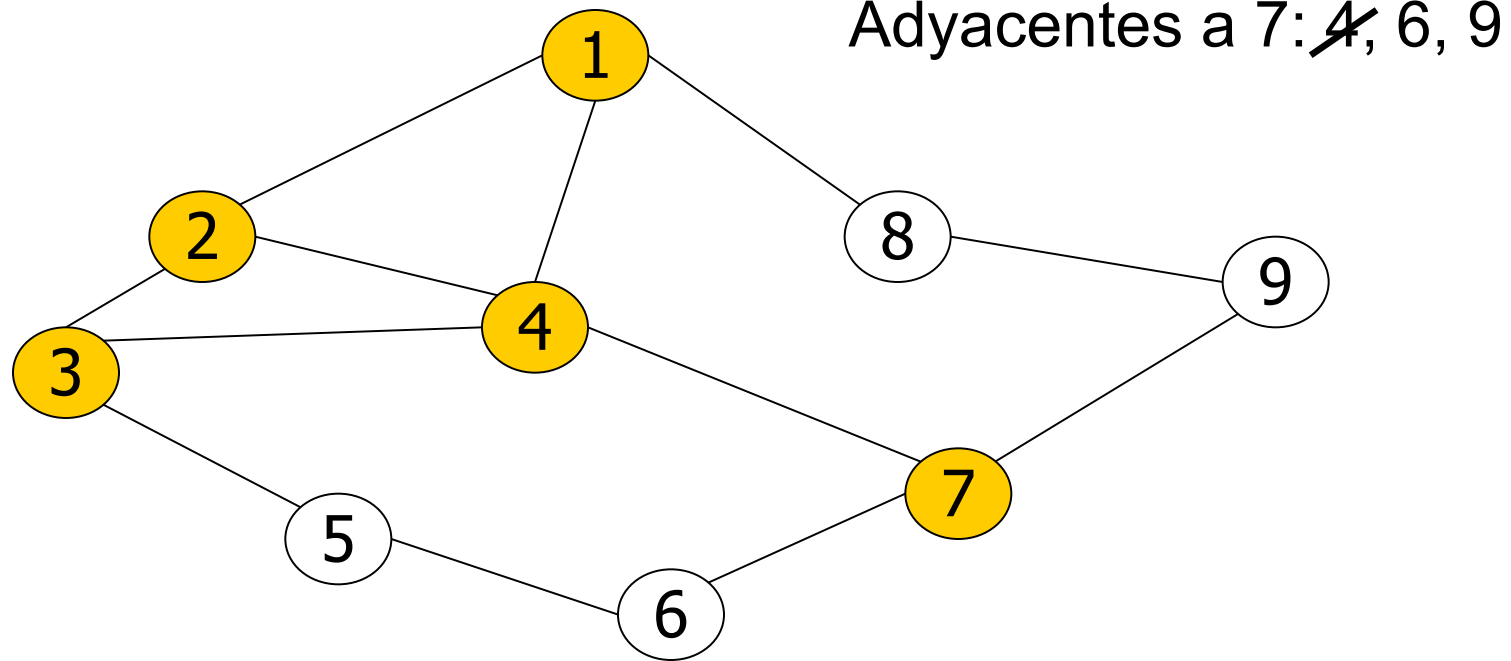
Recorrido en profundidad



Recorrido: 1, 2, 3, 4, 7

Tema 4. Exploración de grafos

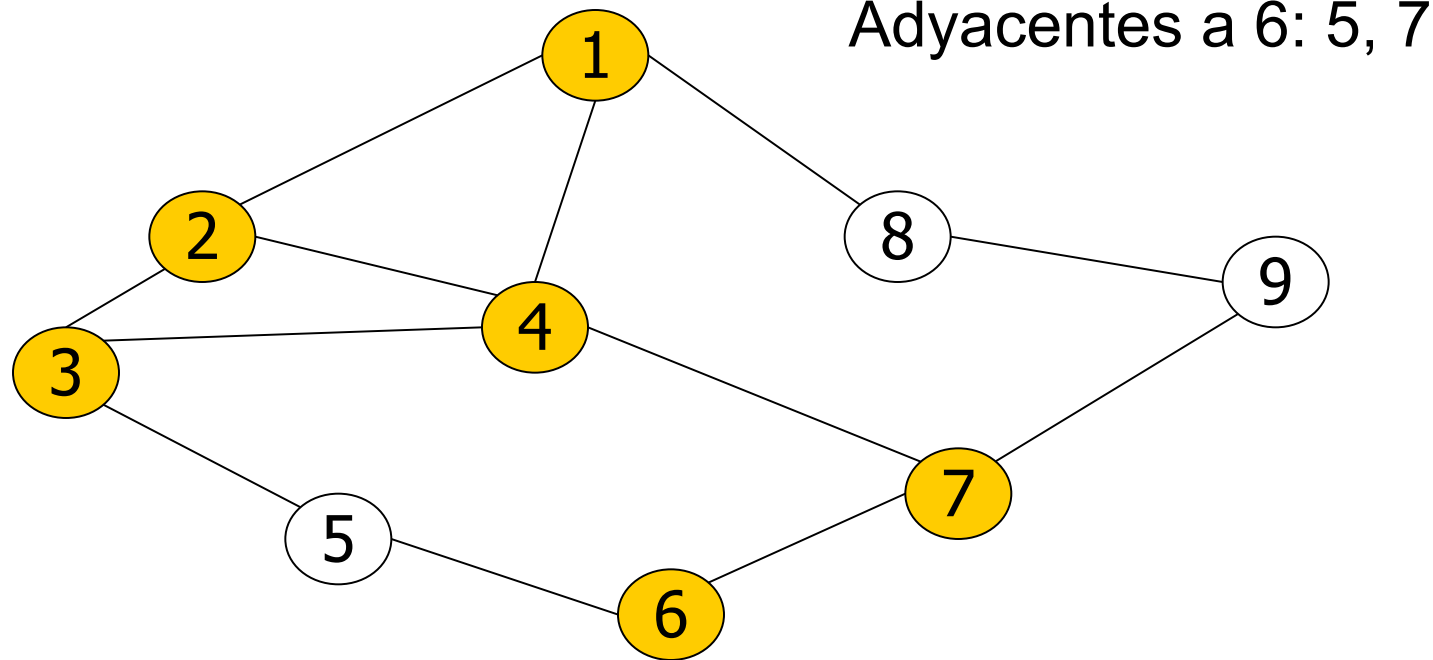
Recorrido en profundidad



Recorrido: 1, 2, 3, 4, 7, 6

Tema 4. Exploración de grafos

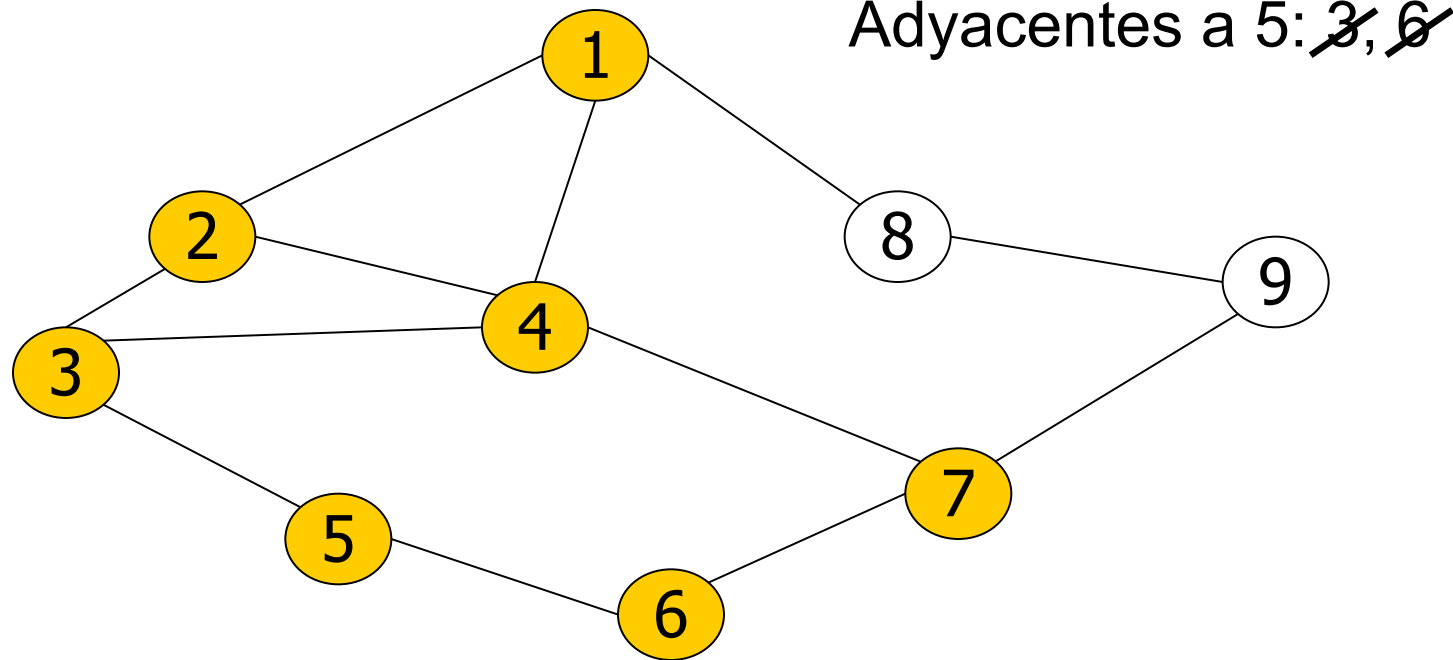
Recorrido en profundidad



Recorrido: 1, 2, 3, 4, 7, 6, 5

Tema 4. Exploración de grafos

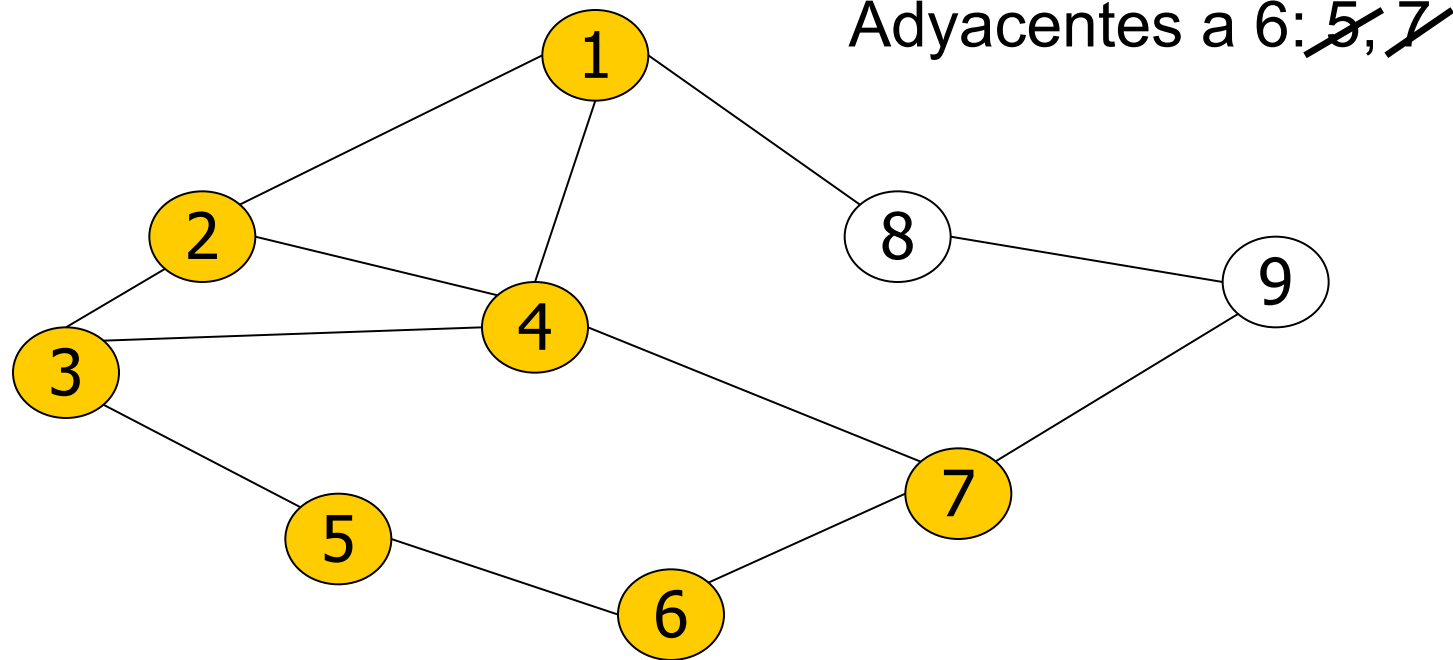
Recorrido en profundidad



Recorrido: 1, 2, 3, 4, 7, 6, 5

Tema 4. Exploración de grafos

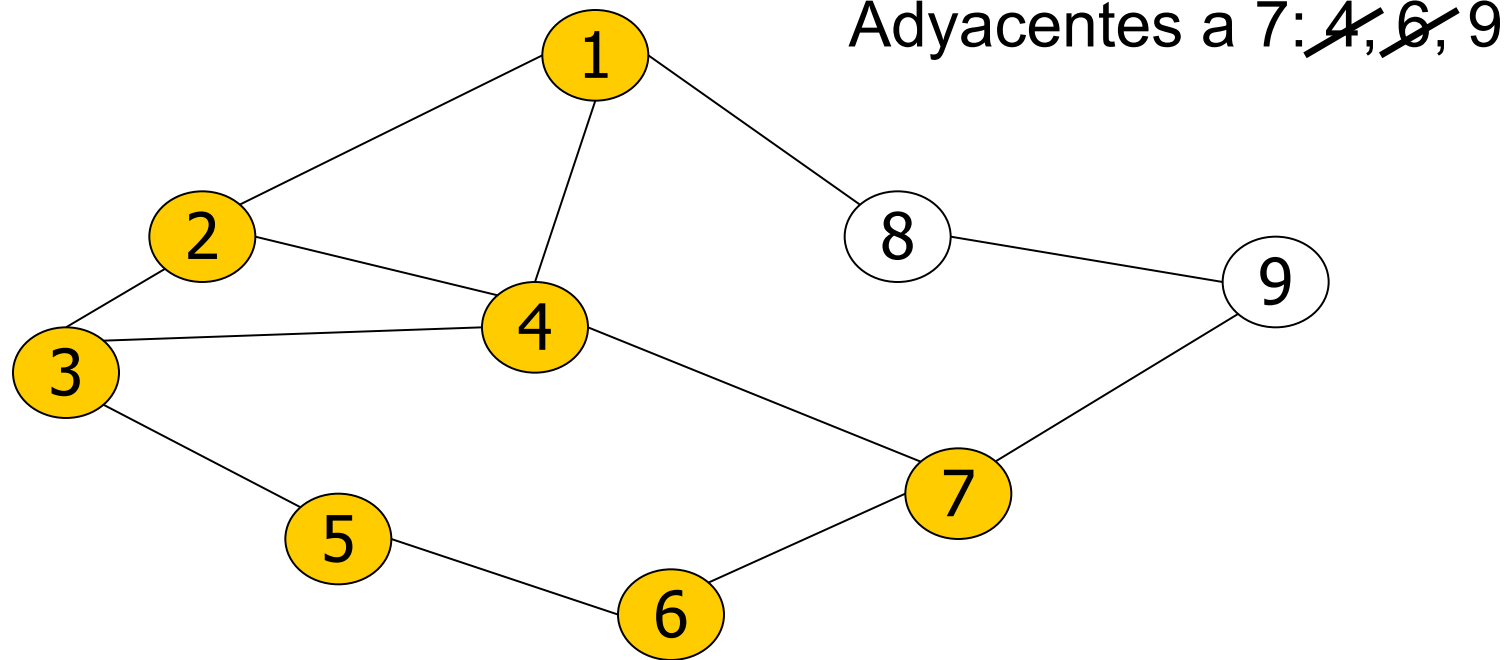
Recorrido en profundidad



Recorrido: 1, 2, 3, 4, 7, 6, 5

Tema 4. Exploración de grafos

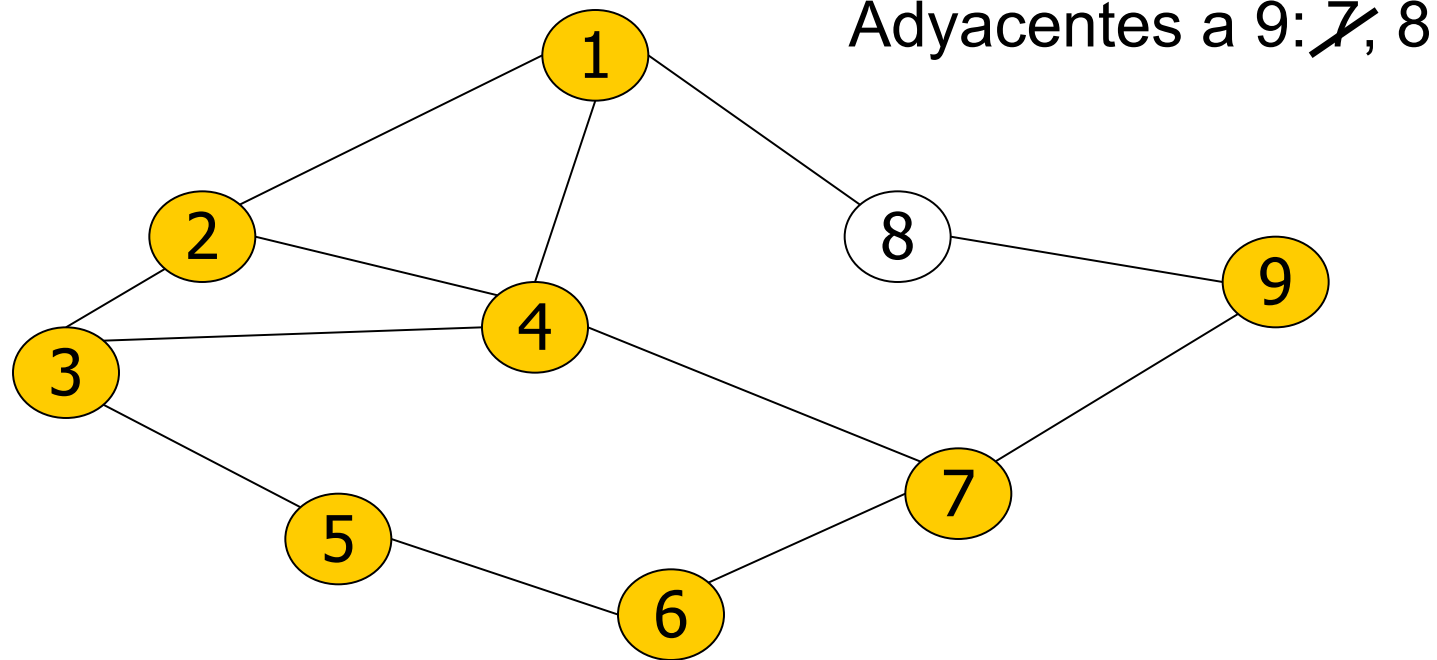
Recorrido en profundidad



Recorrido: 1, 2, 3, 4, 7, 6, 5, 9

Tema 4. Exploración de grafos

Recorrido en profundidad



Recorrido: 1, 2, 3, 4, 7, 6, 5, 9, 8

Tema 4. Exploración de grafos

Recorrido en profundidad

procedimiento recorrido_en_profundidad (G: grafo)

para cada $v \in N$ **hacer**

 marca[v] \leftarrow no visitado

fpara

para cada $v \in N$ **hacer**

si marca[v] \neq visitado **entonces**

 rp(G, marca, v)

fsi

fpara

fprocedimiento

procedimiento rp (G: grafo; marca:vector[1..n]; v: nodo)

{El nodo v no ha sido visitado anteriormente}

 marca[v] \leftarrow visitado

para cada nodo adyacente a v **hacer**

si marca[w] \neq visitado **entonces**

 rp(G, marca, w)

fsi

fpara

fprocedimiento

Tema 4. Exploración de grafos

Recorrido en profundidad

Análisis de la complejidad

- Cada **vértice** se visita **una única** vez $\Rightarrow n$ llamadas al procedimiento $rp \Rightarrow \Theta(V)$
- El algoritmo examina **todas** las **aristas** $\Rightarrow \Theta(E)$
- La **complejidad global** del algoritmo es
$$\Theta(\max(V, E))$$

Tema 4. Exploración de grafos

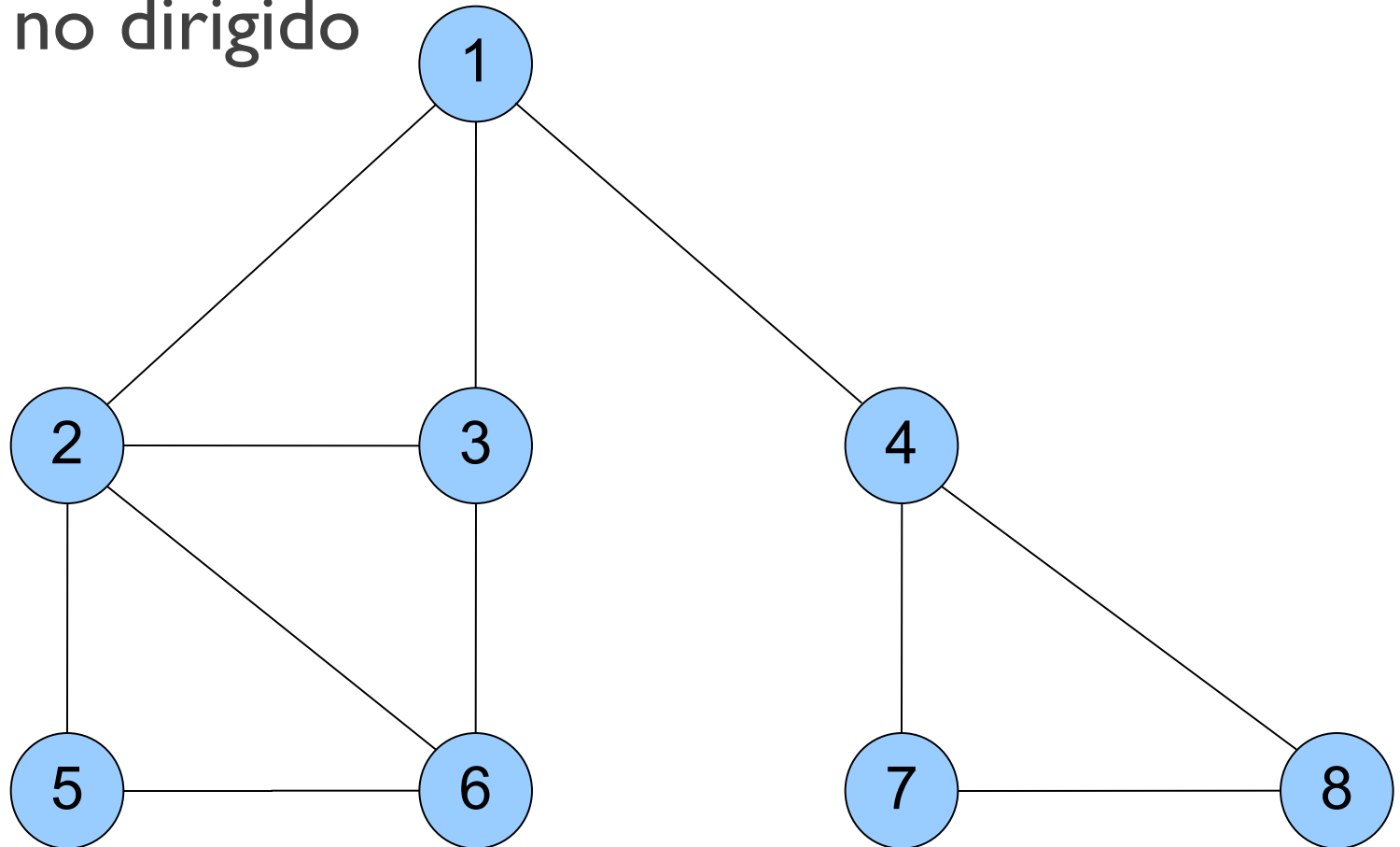
Recorrido en profundidad

- El recorrido en profundidad de un grafo conexo G crea un **árbol de recubrimiento** T
 - Las aristas de T son un **subconjunto** de las aristas de G
 - La **raíz** de T es el punto de partida de la exploración de G
- Si el grafo no es **conexo**, se obtiene un árbol por cada **componente conexa** (bosque)
- La exploración en profundidad de un grafo visita los nodos del grafo en "**preorden**"

Tema 4. Exploración de grafos

Recorrido en profundidad

- Ejemplo: recorrido en profundidad del siguiente grafo no dirigido

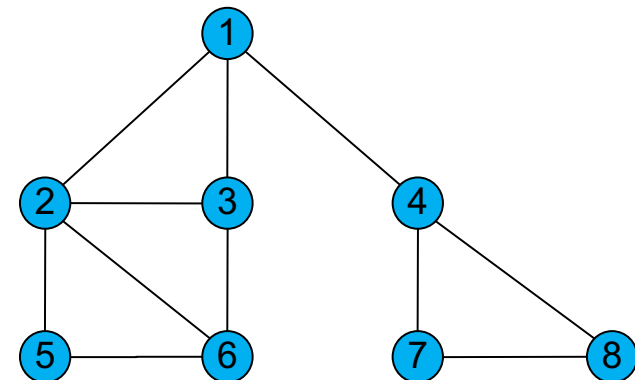


Tema 4. Exploración de grafos

Recorrido en profundidad

• Secuencia de llamadas a rp

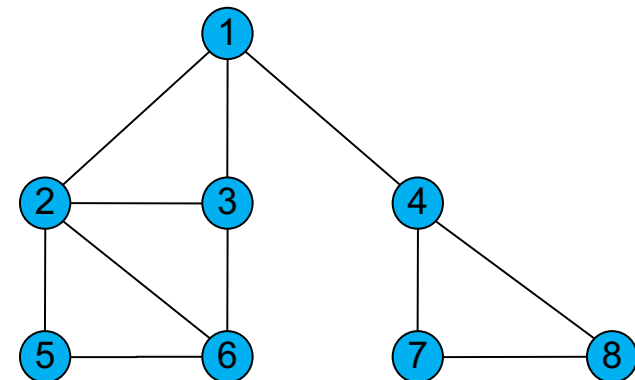
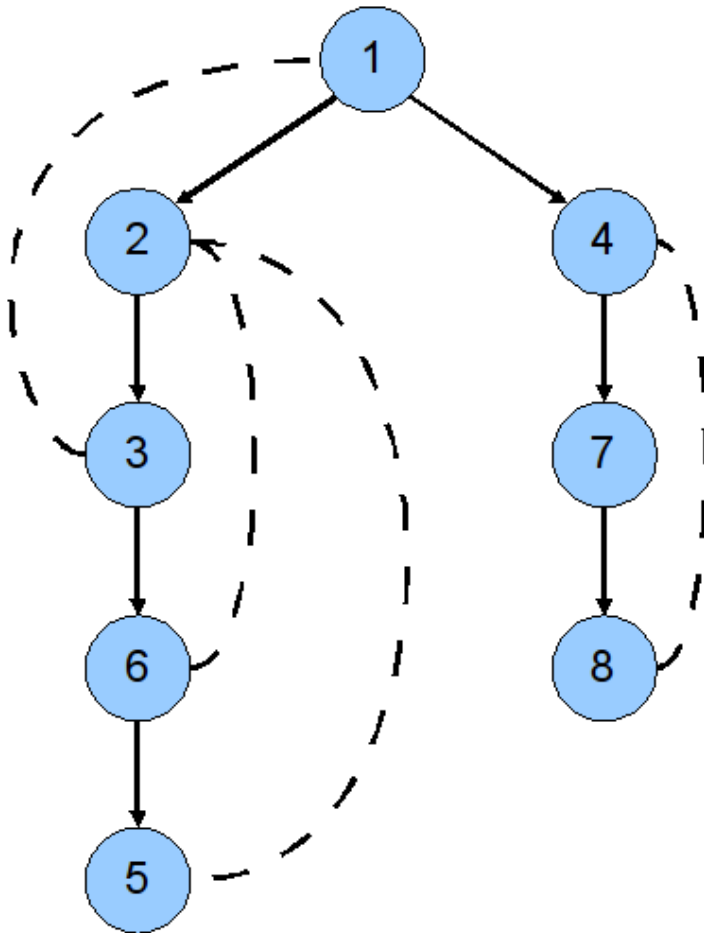
1. $rp(1)$ {llamada inicial}
2. $rp(2)$ {llam. rec.}
3. $rp(3)$ {llam. rec.}
4. $rp(6)$ {llam. rec}
5. $rp(5)$ {llam. rec. no se puede continuar}
6. $rp(4)$ {llam. adyacente al nodo 1}
7. $rp(7)$ {llam. rec}
8. $rp(8)$ {llam. rec. no se puede continuar}



Tema 4. Exploración de grafos

Recorrido en profundidad

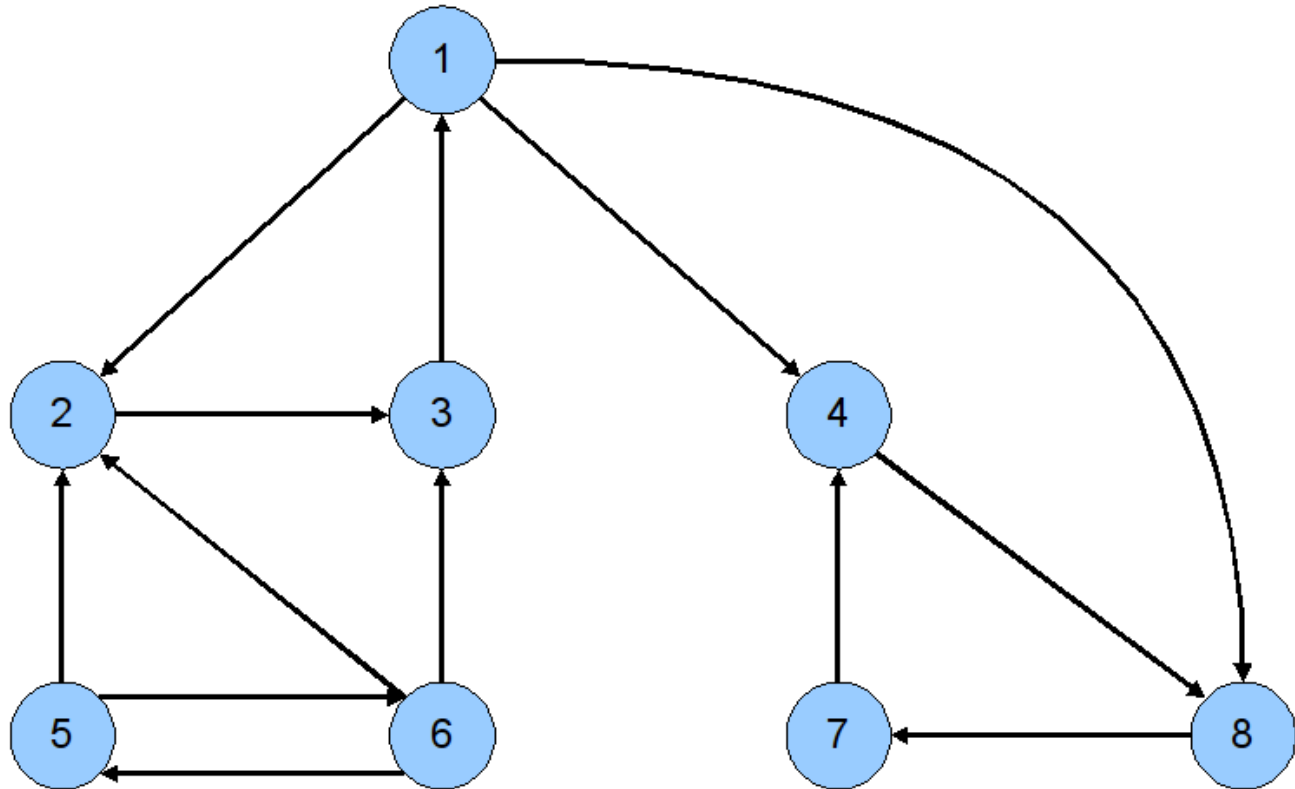
- Árbol de recorrido en profundidad (T)



Tema 4. Exploración de grafos

Recorrido en profundidad en grafos dirigidos

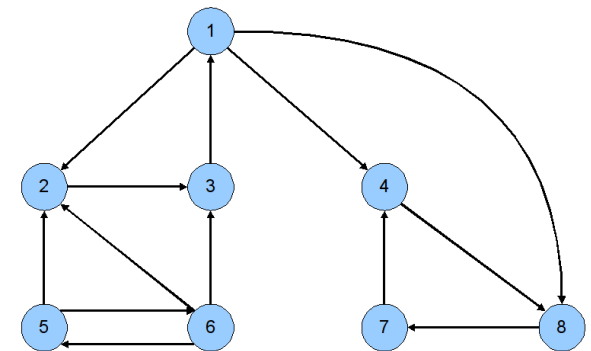
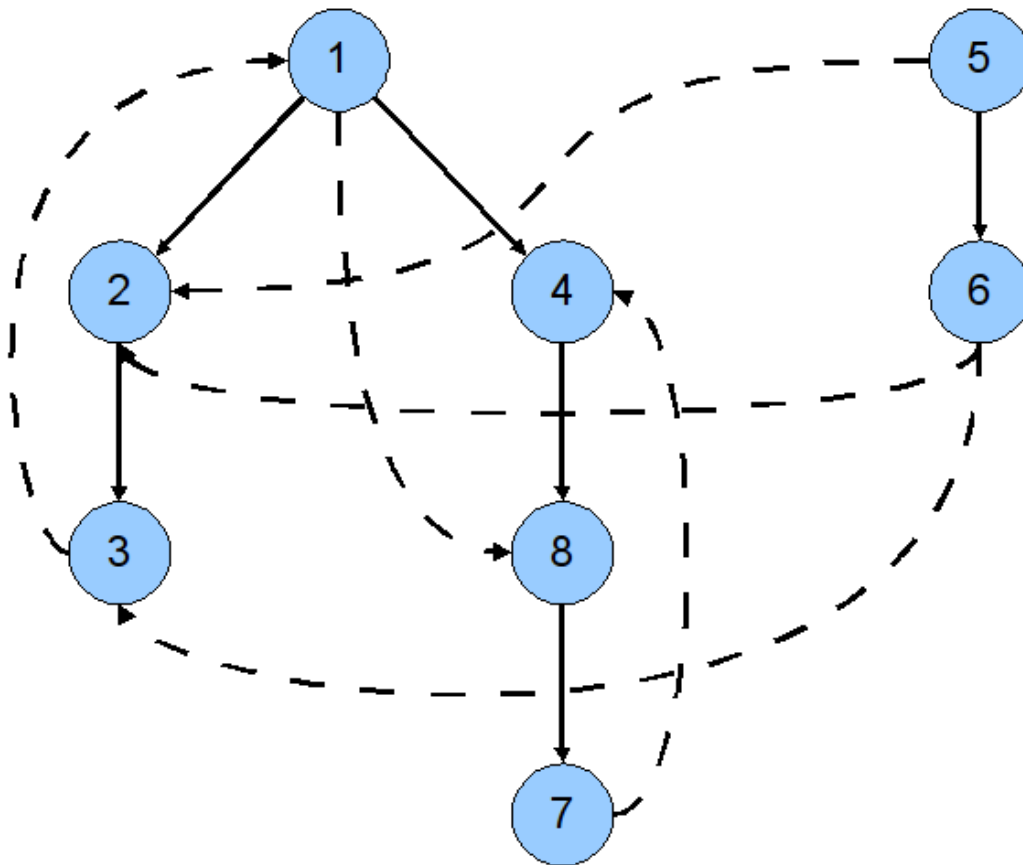
El procedimiento es esencialmente el mismo, salvo que se distingue entre adyacentes e incidentes



Tema 4. Exploración de grafos

Recorrido en profundidad en grafos dirigidos

“Bosque” de recorrido en profundidad



Tema 4. Exploración de grafos

Recorrido en anchura

- El recorrido en anchura (*breath-first search*) se llama así porque recorre la frontera en anchura
 - Visita todos los vértices a una **distancia k** antes de descubrir el primer vértice a la **distancia $k+1$**
 - Suele implementarse de manera **iterativa**
 - Se tiene que incluir una **cola** con los vértices visitados para evitar **ciclos** y establecer el orden en la búsqueda

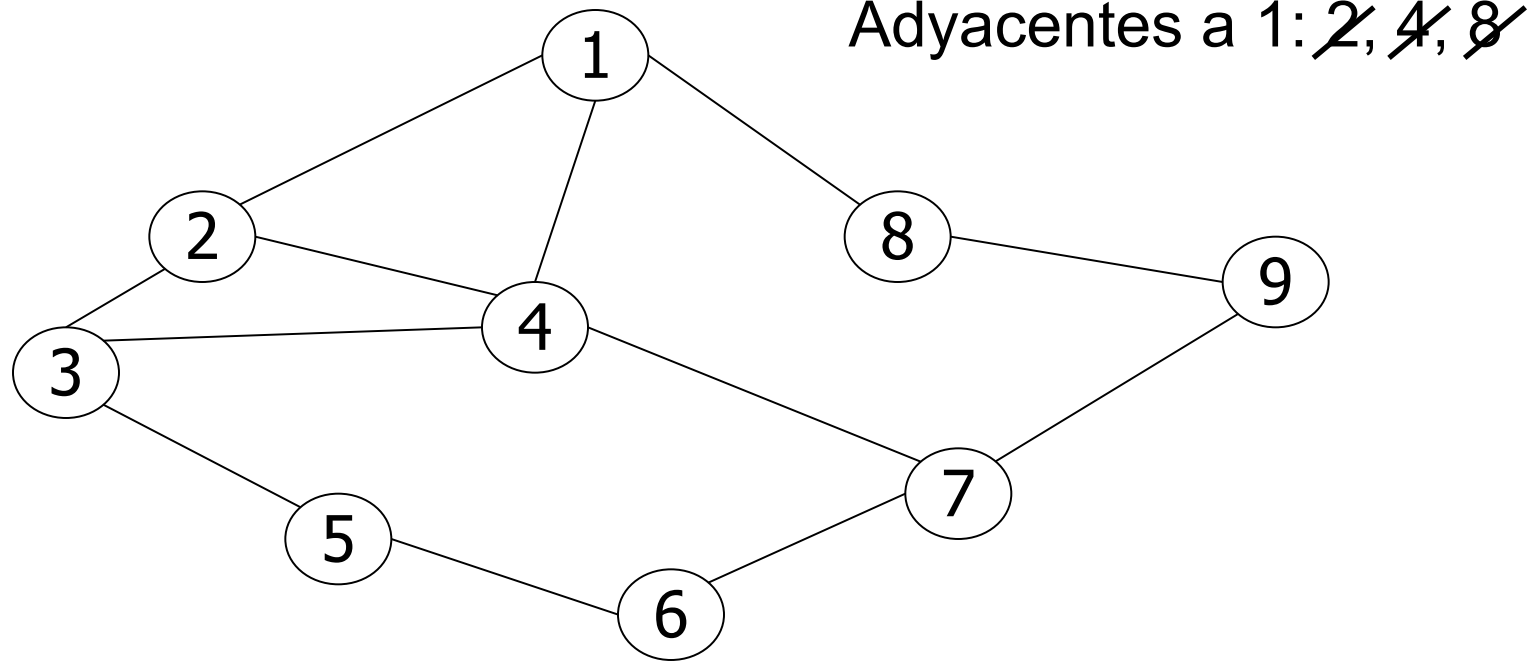
Tema 4. Exploración de grafos

Recorrido en anchura

- Dado un grafo $G=(V, E)$ y un vértice inicial s :
 - Calcula la **distancia** (menor número de vértices) desde s hasta los vértices alcanzables
 - Produce un **árbol** de recorrido en anchura donde la raíz es s
 - Para cualquier vértice v alcanzable desde s , la ruta en el árbol de recorrido en anchura desde s hasta v es el **camino más corto** entre esos dos vértices
 - Es un algoritmo adecuado para grafos dirigidos y no dirigidos

Tema 4. Exploración de grafos

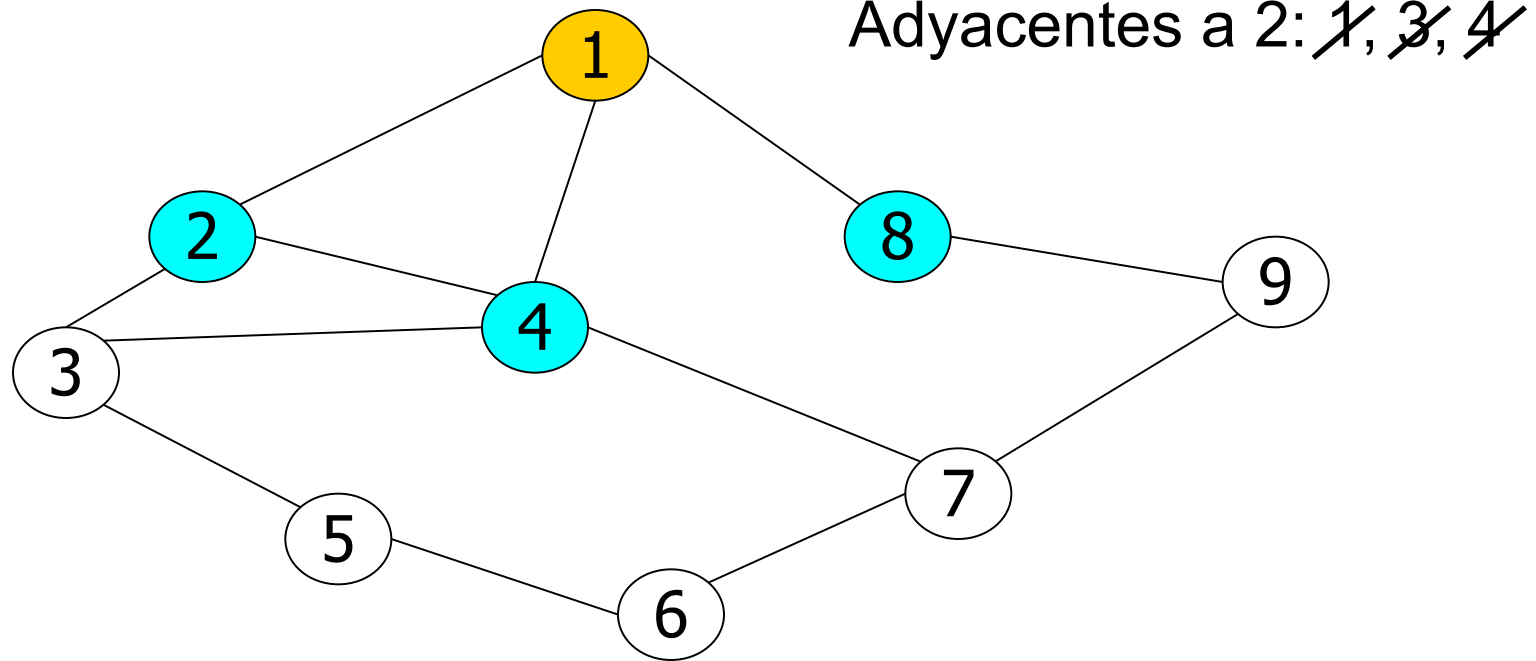
Recorrido en anchura



Recorrido: 1, 2, 4, 8

Tema 4. Exploración de grafos

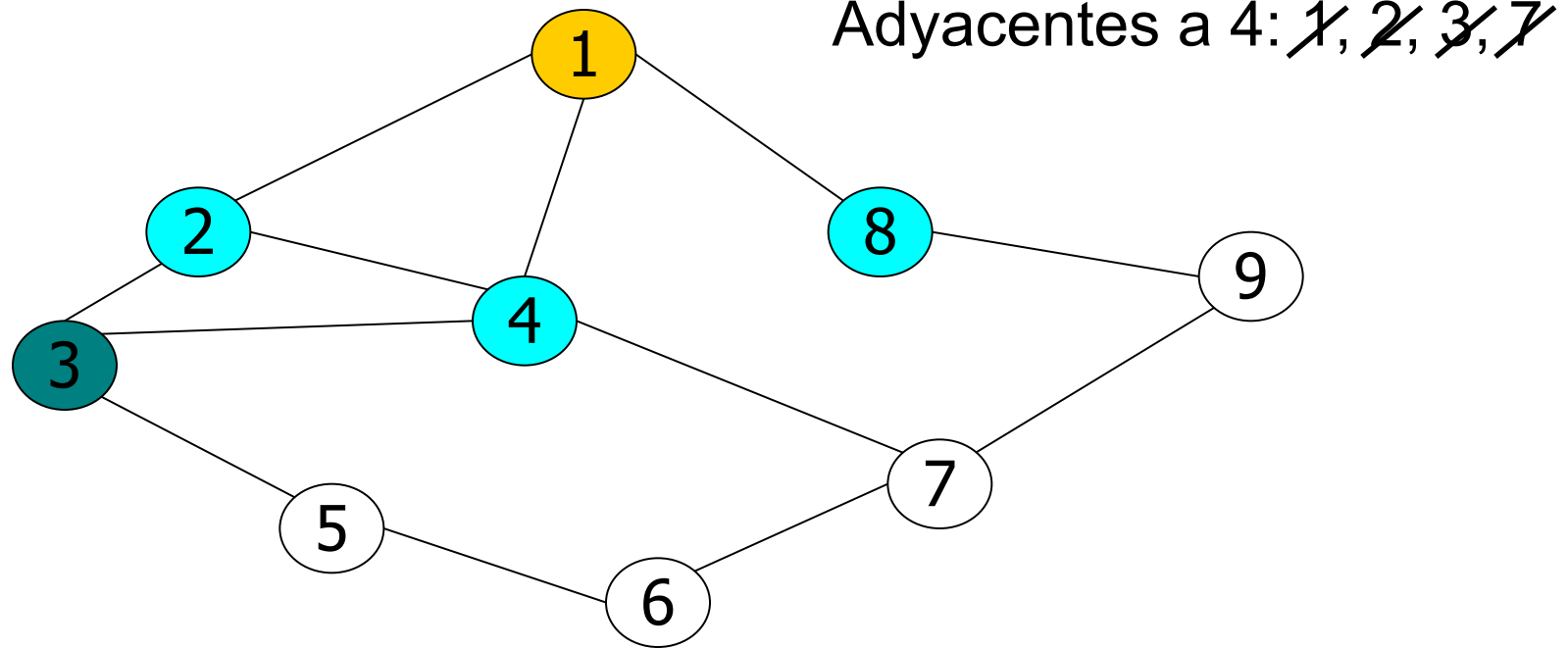
Recorrido en anchura



Recorrido: 1, 2, 4, 8, 3

Tema 4. Exploración de grafos

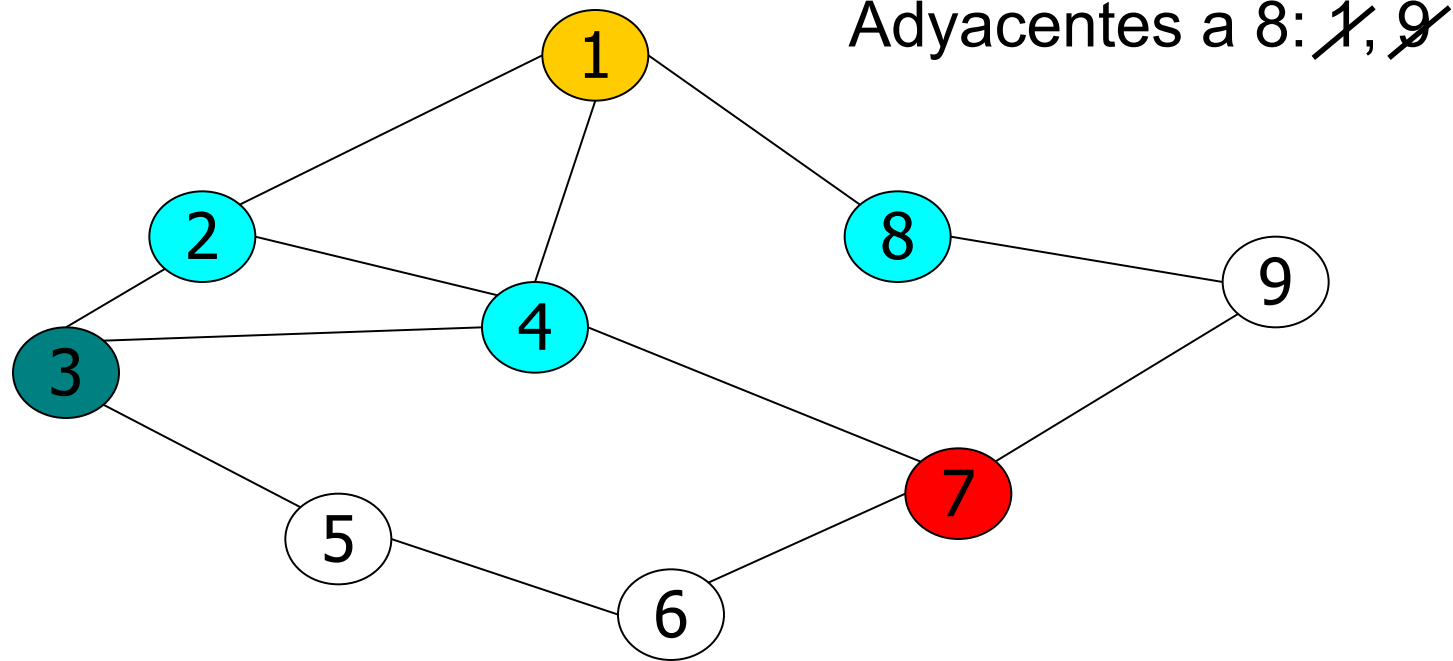
Recorrido en anchura



Recorrido: 1, 2, 4, 8, 3, 7

Tema 4. Exploración de grafos

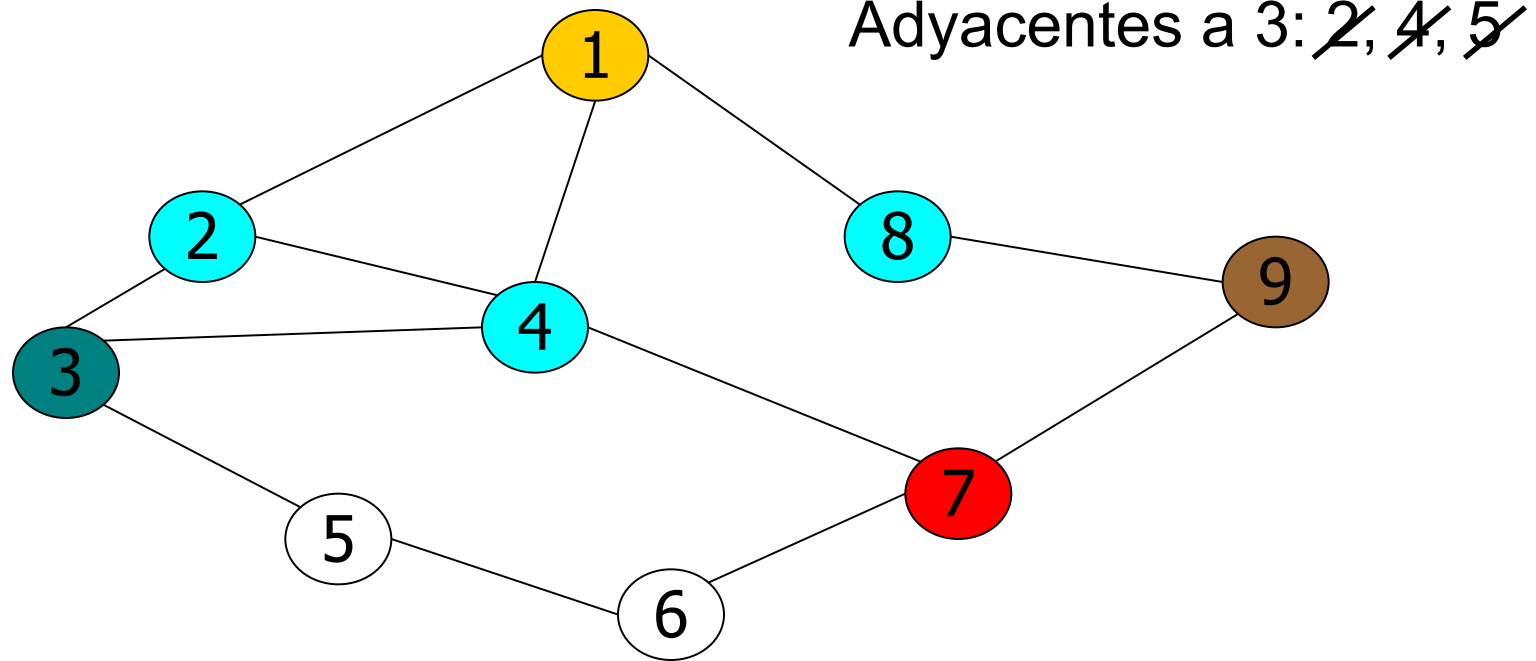
Recorrido en anchura



Recorrido: 1, 2, 4, 8, 3, 7, 9

Tema 4. Exploración de grafos

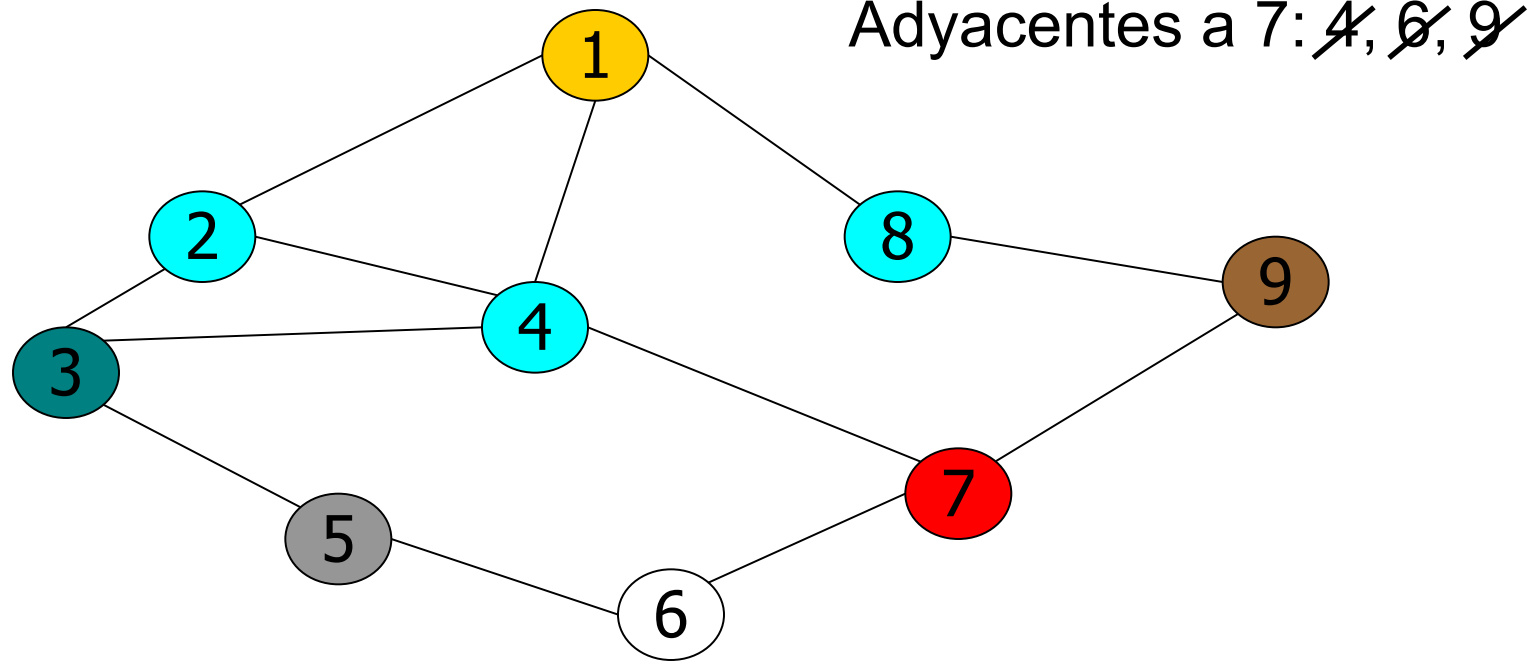
Recorrido en anchura



Recorrido: 1, 2, 4, 8, 3, 7, 9, 5

Tema 4. Exploración de grafos

Recorrido en anchura



Recorrido: 1, 2, 4, 8, 3, 7, 9, 5, 6

Tema 4. Exploración de grafos

Recorrido en anchura

```
procedimiento recorrido(G)
  para cada  $v \in N$  hacer
    marca[v]  $\leftarrow$  no visitado
  fpara
  para cada  $v \in N$  hacer
    si marca[v]  $\neq$  visitado entonces
      ra(v)
  fsi
  fpara
fprocedimiento
procedimiento ra(v)
  Q  $\leftarrow$  colavacia
  marca[v]  $\leftarrow$  visitado
  poner v en Q
  mientras Q no esté vacía hacer
    quitar aux de Q
    para todo w adyacente a aux hacer
      si marca[w]  $\neq$  visitado entonces
        marca[w]  $\leftarrow$  visitado
        poner w en Q
      fsi
    fpara
  fmientras
fprocedimiento
```

Tema 4. Exploración de grafos

Recorrido en anchura

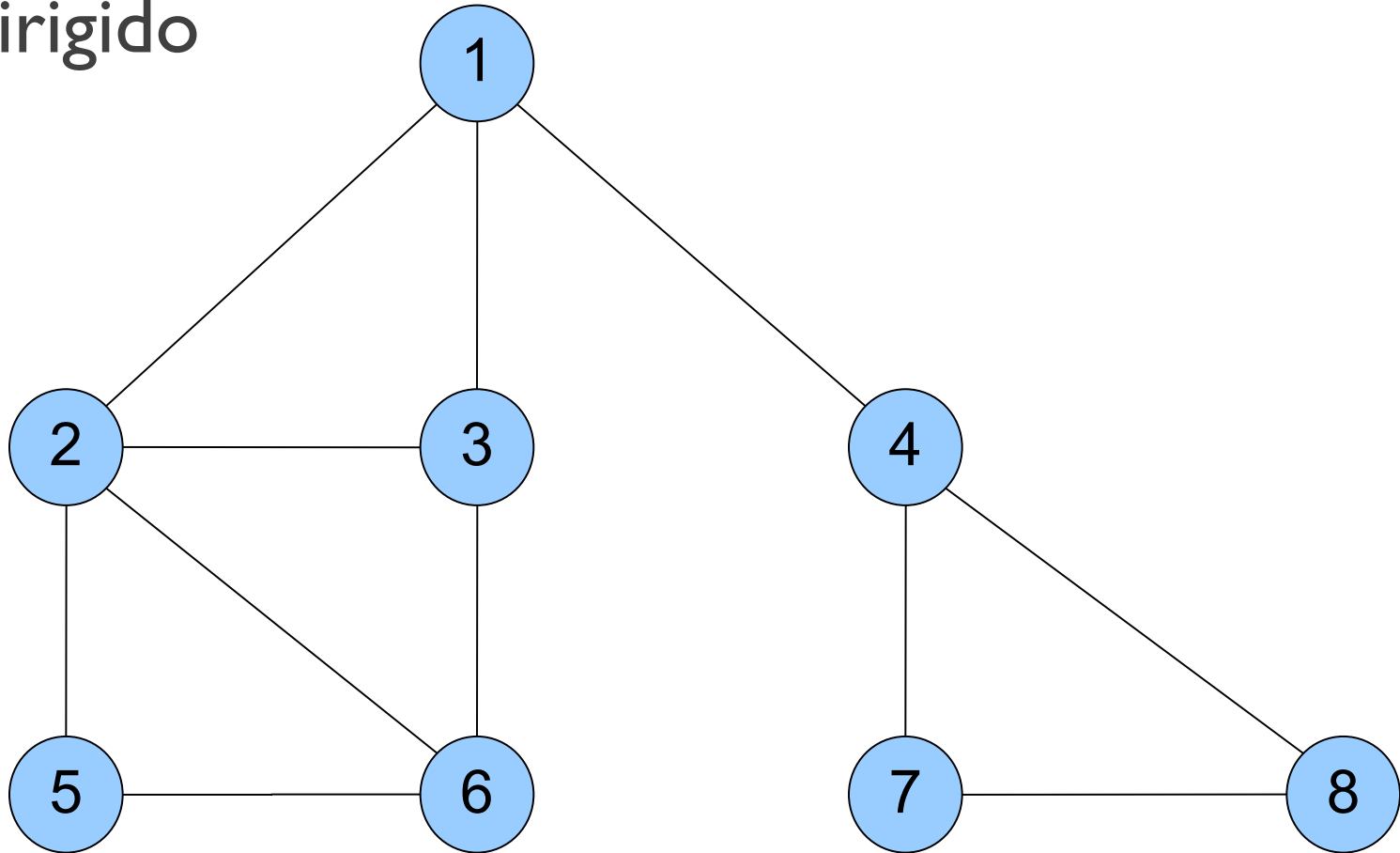
Análisis de complejidad

- El **tiempo requerido** para realizar un recorrido en **anchura** de un grafo es el mismo que para un recorrido en **profundidad** $\Rightarrow \Theta(\max(V, E))$
- También genera **árboles de recubrimiento**. Si el grafo es conexo \Rightarrow un único árbol.
- Se emplea en exploraciones parciales de grafos, para hallar el camino más corto entre dos puntos de un grafo, etc.

Tema 4. Exploración de grafos

Recorrido en anchura

- Ejemplo: recorrido en anchura del siguiente grafo no dirigido

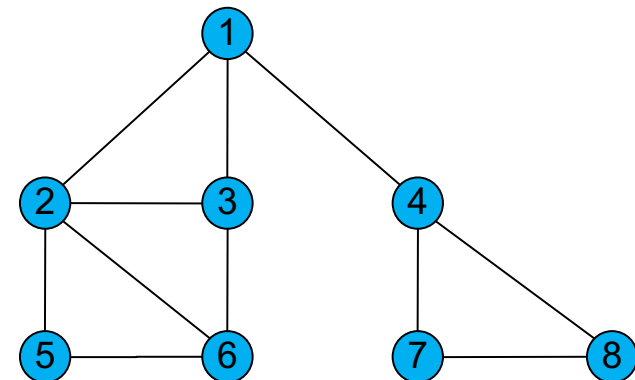


Tema 4. Exploración de grafos

Recorrido en anchura

- Ejemplo: recorrido en anchura

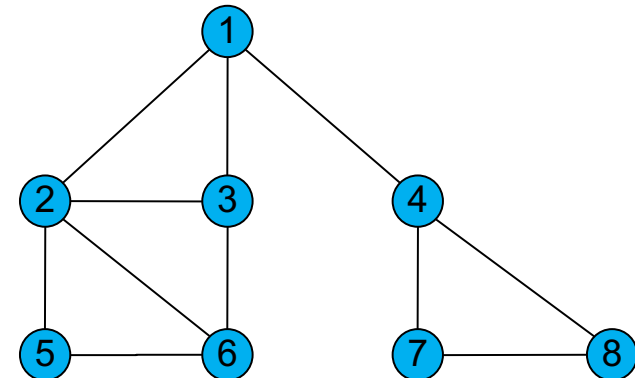
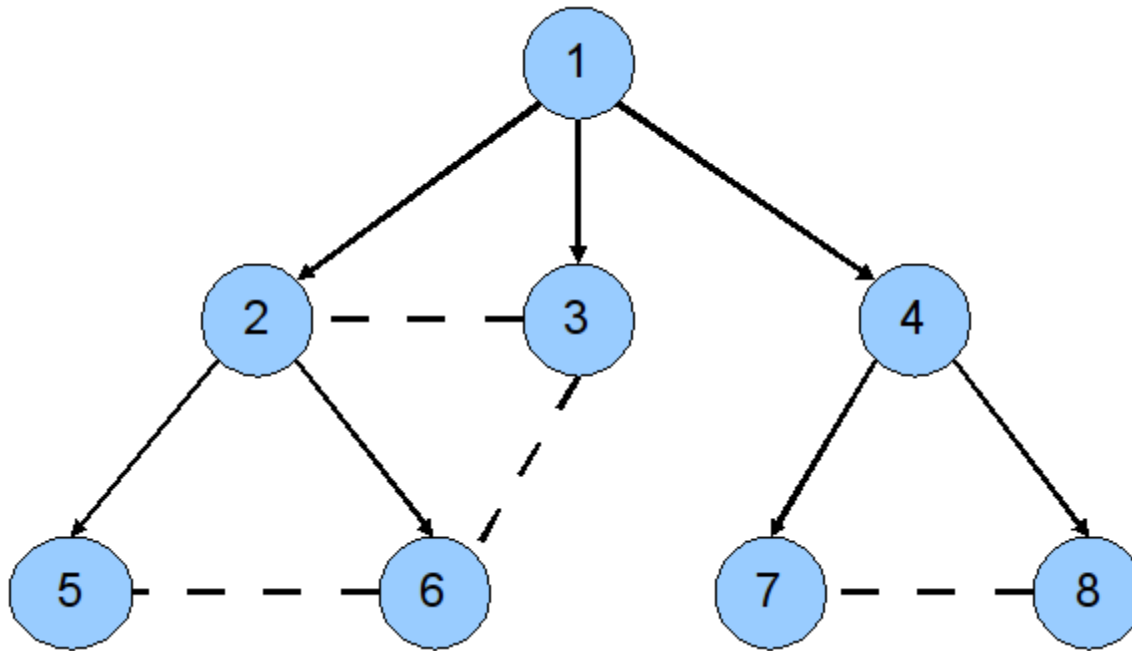
Paso	Nodo Visitado	Q
1	1	2, 3, 4
2	2	3, 4, 5, 6
3	3	4, 5, 6
4	4	5, 6, 7, 8
5	5	6, 7, 8
6	6	7, 8
7	7	8
8	8	-



Tema 4. Exploración de grafos

Recorrido en anchura

Árbol de recorrido en anchura:



Tema 4. Exploración de grafos

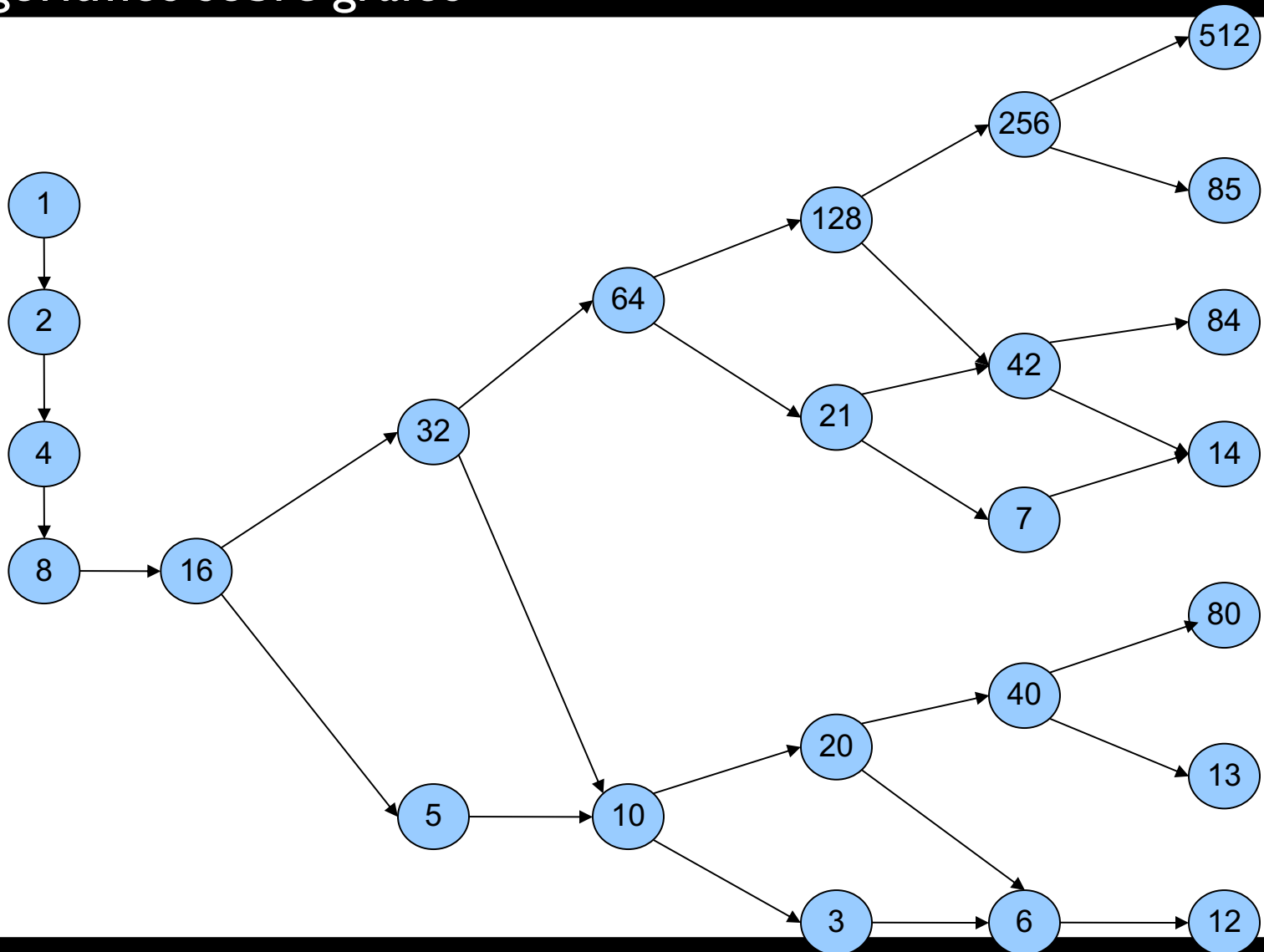
Algoritmos sobre grafos

Ejemplo de recorridos sobre grafos infinitos

- Se nos da:
 - El valor inicial 1
 - Las operaciones $\times 2$ y $\div 3$ (\div es división entera)
- El objetivo es construir otros valores. Por ejemplo: $10 = 1 \times 2 \times 2 \times 2 \times 2 \div 3 \times 2$
- Se desea construir un valor concreto n :
 - Búsqueda en un grafo infinito
 - Un recorrido en profundidad podría no funcionar
 - Un recorrido en anchura es la mejor solución

Tema 4. Exploración de grafos

Algoritmos sobre grafos



.....

Tema 4. Exploración de grafos

Algoritmos sobre grafos

- Dado un grafo dirigido y acíclico, se denomina **ordenación topológica** a una disposición lineal de los nodos tal que, dado un arco (u,v) , el nodo u esté antes que v en la ordenación
- Un vértice se visita sí y sólo sí se han visitados todos sus predecesores
- Aplicaciones prácticas
 - Fases de un proyecto (PERT)
 - Evaluación de atributos en la fase semántica de un compilador

Tema 4. Exploración de grafos

Algoritmos sobre grafos

Recorrido en profundidad. Distancia y finalización

- Los vértices se colorean como sigue:
 - Inicialmente son todos **blancos**
 - Cuando se visitan, se colorean de **gris** (denotado por $d[v]$)
 - Cuando toda la adyacencia se ha visitado se colorea de **negro** (denotado por $f[v]$)
- Propiedades
 - Un vértice es **blanco** hasta $d[v]$, **gris** desde $d[v]$ hasta $f[v]$ y **negro** en adelante
 - $d[v]$ y $f[v]$ (con $d[v] < f[v]$ para todo v) **toman valores** entre 1 y $2^*|V|$

Tema 4. Exploración de grafos

Algoritmos sobre grafos

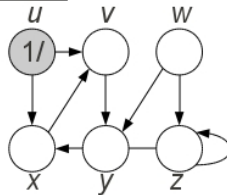
Las letras U, V, W, X, Y y Z son los identificadores de los nodos

Ejemplo de recorrido

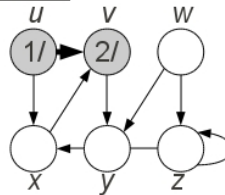
Errata: la arista entre Y y Z tiene que tener una flecha desde Z hacia Y

A partir del paso 7 hay otra errata, en el nodo X debería poner 4/5 como en el paso anterior en vez de 4/6

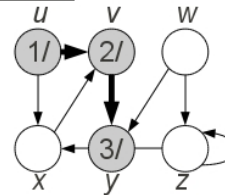
Paso 1



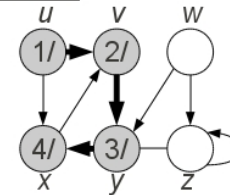
Paso 2



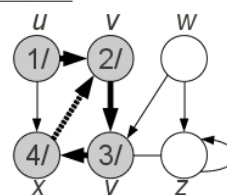
Paso 3



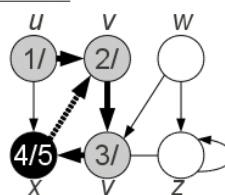
Paso 4



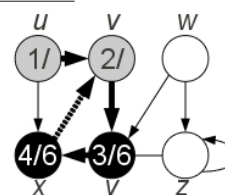
Paso 5



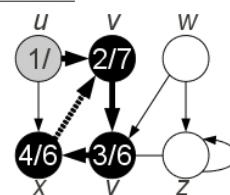
Paso 6



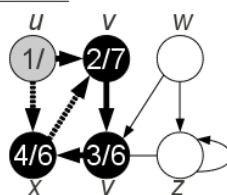
Paso 7



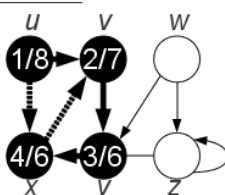
Paso 8



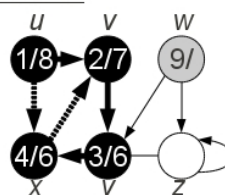
Paso 9



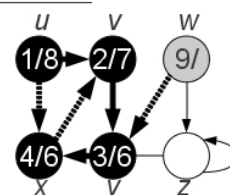
Paso 10



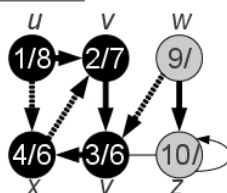
Paso 11



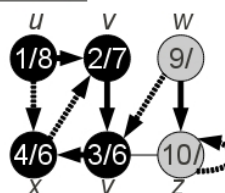
Paso 12



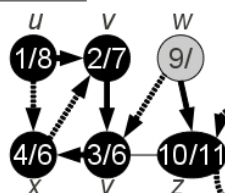
Paso 13



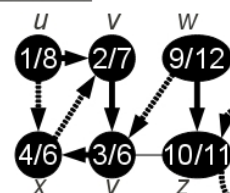
Paso 14



Paso 15



Paso 16



Tema 4. Exploración de grafos

Algoritmos sobre grafos

- Algoritmo

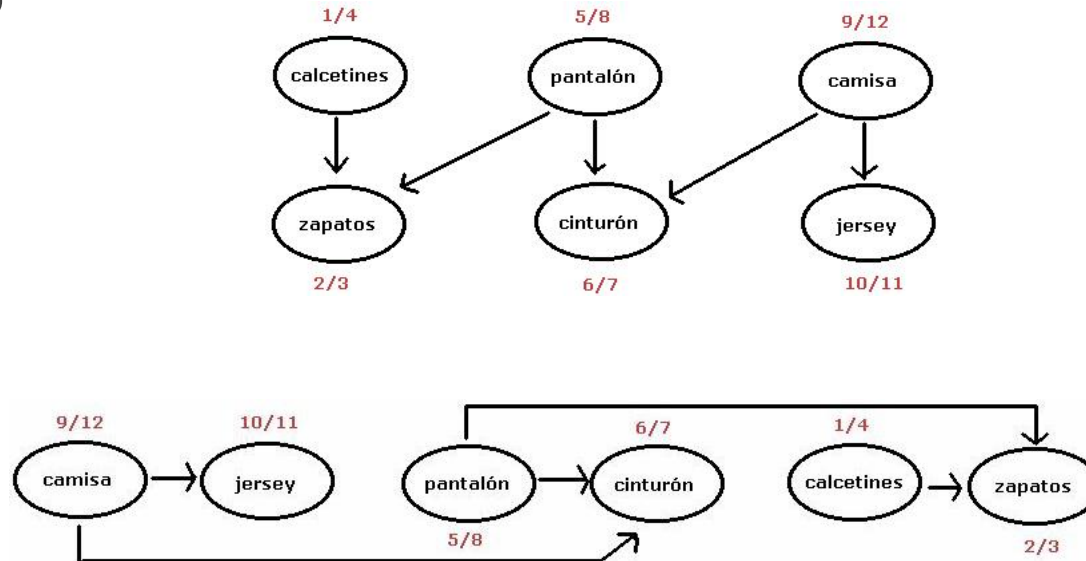
```
ORDENACIÓN TOPOLÓGICA(grafo G)
for each vértice  $u \in V[G]$  do
    estado[u] = NO_VISITADO
    padre[u] = NULL
    tiempo = 0
    for each vértice  $u \in V[G]$  do
        if estado[u] = NO_VISITADO then
            TOPOLÓGICO-Visitar(u)
```

```
TOPOLÓGICO-Visitar(nodo u)
estado[u]=VISITADO
tiempo = tiempo+1
d[u] = tiempo
for each  $v \in \text{Adyacencia}[u]$  do
    if estado[v]=NO_VISITADO then
        padre[v]=u
        TOPOLÓGICO-Visitar(v)
estado[u] = TERMINADO
tiempo = tiempo+1
f[u] = tiempo
insertar (lista, u)
```

Tema 4. Exploración de grafos

Algoritmos sobre grafos

- Ejemplo



- Ponerse la **camisa** antes que el **cinturón** y **jersey**
- Ponerse el **pantalón** antes que los **zapatos** y **cinturón**
- Ponerse los **calcetines** antes que los **zapatos**

Tema 4. Exploración de grafos

Algoritmos sobre grafos

- Un grafo se dice que es **fuertemente conexo** si existe un camino entre cada par de vértices
- Para **descomponer** un grafo G en sus componentes fuertemente conexas
 - Aplicar ordenación topológica sobre G
 - Calcular el grafo traspuesto G^T
 - Aplicar búsqueda en profundidad sobre G^T iniciando la búsqueda en los nodos de mayor a menor **tiempo de finalización** obtenidos en la primera ejecución de búsqueda en profundidad
 - El resultado será un bosque de árboles. Cada árbol es un componente fuertemente conexo.

Tema 4. Exploración de grafos

Algoritmos sobre grafos

- **Puntos de articulación:** un vértice v de un grafo conexo es un punto de articulación si el subgrafo que se obtiene al eliminarlo (junto con sus aristas) es no conexo
- **Grafo biconexo** (o no articulado): aquél grafo que no tiene puntos de articulación
- **Grafo bicoherente** (o 2-arista conexo): aquél grafo cuyos puntos de articulación están unidos a cada componente del subgrafo restante por, al menos, dos aristas

Tema 4. Exploración de grafos

Algoritmos sobre grafos

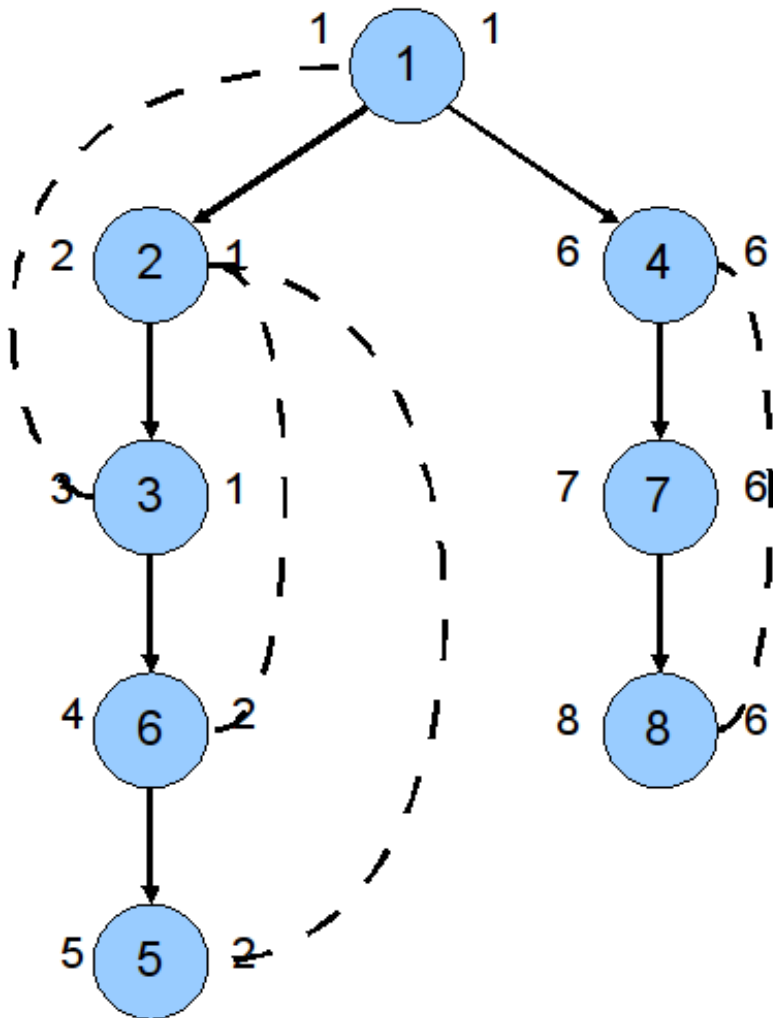
Si el grafo representa una red de comunicaciones

- Si es **biconexo** nos asegura el correcto funcionamiento de la red, aunque falle uno de los equipos
- Si es **bicoherente** nos asegura el correcto funcionamiento de la red, aunque falle una de las líneas de transmisión

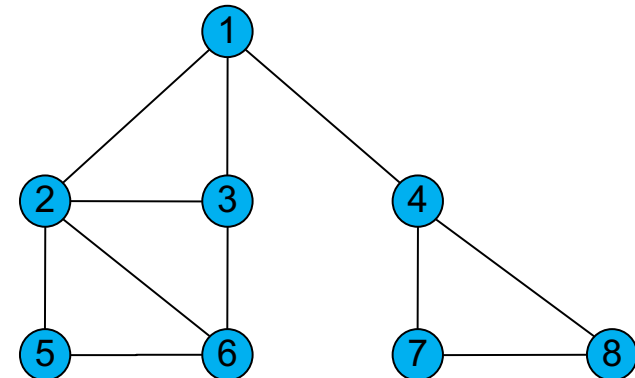
Tema 4. Exploración de grafos

Algoritmos sobre grafos

Cálculo de los puntos de articulación



- A la izda de cada vértice se representa la it. de visita
- A la dcha de cada vértice se representa el nodo más alto alcanzable



Tema 4. Exploración de grafos

Algoritmos sobre grafos

Cálculo de los puntos de articulación

- Sea u cualquier vértice del árbol (excepto la raíz) y v un hijo de u
 - Si $\text{low}[v] < \text{disc}[u]$ implica que se puede llegar desde x a regiones más altas sin pasar por (v,x) . Entonces u **no es un punto de articulación**
 - Si $\text{low}[x] \geq \text{disc}[v]$ no se puede llegar desde x a regiones más altas del grafo sin pasar por (v,x) . Entonces u **es un punto de articulación**
- Si u es la **raíz del árbol** y tiene más de un hijo, **es un punto de articulación**

Tema 4. Exploración de grafos

Algoritmos sobre grafos

Cálculo de los puntos de articulación

- Realizar un recorrido en profundidad (modificado) de G
 - Obtener el árbol de recubrimiento
 - Calcular la **iteración de descubrimiento** de cada nodo del árbol
- Calcular $\text{low}[u]$ como
 - **Bajando** (todo lo que se necesite) por las aristas usadas en el árbol de recubrimiento
 - **Subiendo** (por sólo una) arista que no pertenezca al árbol
- Los puntos de articulación de G serán
 - La **raíz** sí y sólo sí tiene más de un hijo
 - Cualquier nodo u que tenga un hijo v tal que $\text{low}[v] \geq \text{disc}[u]$

Tema 4. Exploración de grafos

Algoritmos sobre grafos

Otras aplicaciones interesantes

- Comprobar que un grafo es bipartito
- Detectar ciclos en grafos
- Camino más largo en un DAG
- Detectar si dos nodos están conectados o no
- Caminos y ciclos eulerianos
- Cierre transitivo
- Caminos entre un origen y un destino con k aristas