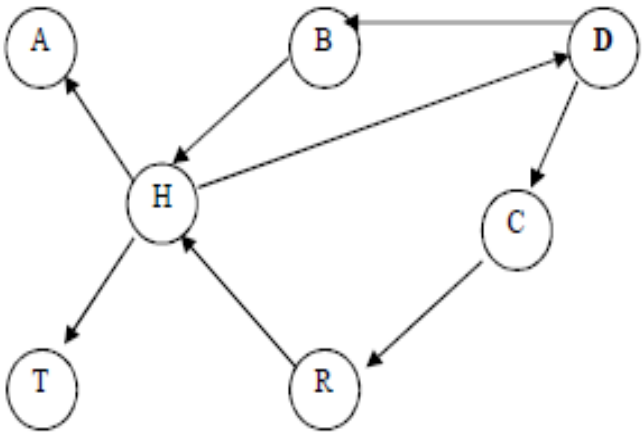
UNIDAD 1

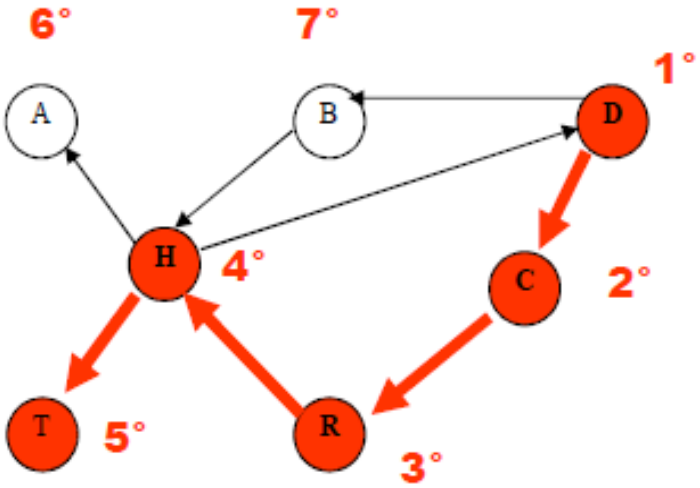
Grafos

Introducción a la teoría de grafos

Aplicaciones

Recorrido DFS





Recorrido DFS a partir del vértice D:
{D,C,R,H,T,A,B}

Recorrido DFS

```
DFS (G, u)
{
Marca[u]= VISITADO;
for  (cada vértice v ε Adyacente[u])
      if Marca[v]= NO_VISITADO
            {
            Padre[v]= u;
            DFS (G,v);
            }
}
```

Obtener los adyacentes

Control de no visitado

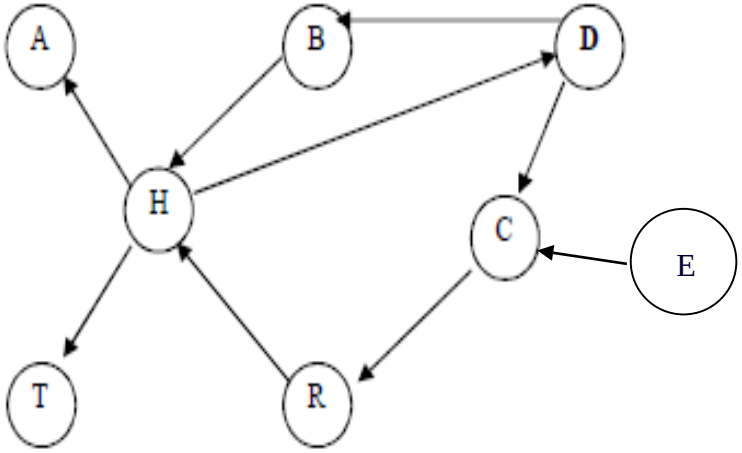
Guardo el padre de v

Recorrido DFS

Todas las invocaciones a DFS de un grafo G con **a** arcos y **n** <= **a** vértices lleva un t de ejecución O(a).

- Ningún vértice se llama mas de una vez.
- Tan pronto como se invoca a DFS se marca el vértice => nunca se llama a un vértice marcado

Recorrido DFS



Si tenemos este nuevo grafo y comenzamos el DFS a partir de D nunca alcanzaríamos a E, lo mismo pasa si partimos a partir de R por ejemplo.

Sin embargo, si recorremos a partir de E:

DFS (G, E)= {E,C,R,H,A,T,D,B}, se alcanzan **todos** los arcos

Recorrido DFS

Como garantizamos alcanzar **todos** los nodos?

DFS Forest que aplicado al grafo G= (V,A)
retorna un bosque de exploración depth-first compuesto de **uno o más** árboles de exploración depth-first

Recorrido DFS Forest

```
DFS_forest (G)
{
for (cada vértice u ε V)
      {
      Marca[u]= NO_VISITADO;
      Padre[u]= NULO;
      }
for (cada vértice u ε V)
      if Marca[u]= NO_VISITADO
            DFS (G, u);
 }
```

Revisa **todos** los vertices de G

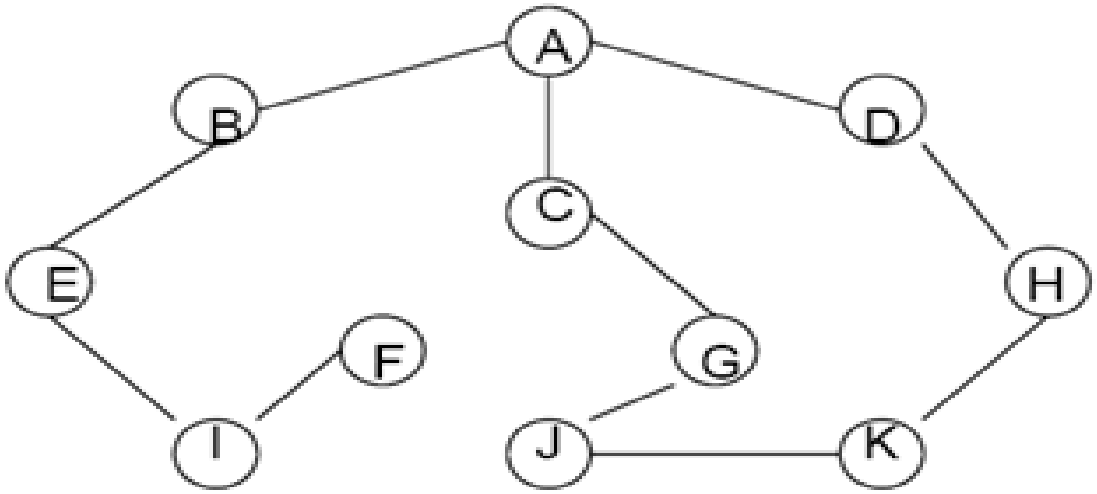
Invoca a DFS si no ha sido ya explorado anteriormente

Recorrido DFS Forest

Tiempo de ejecución ? dependiendo de?

El tiempo de ejecución del algoritmo DFS Forest es O(m), m es el máximo entre el número de vértices y el número de arcos.

Recorrido BFS (breath-first-search)



- Recorrido BFS (G, A) =

A-B-C-D-E-G-H-I-J-K-F

Recorrido BFS

Adecuado para resolver problemas de optimización, donde hay que elegir la mejor solución de entre varias posibles. Procedimiento:

- Comienzo en un vértice v (vértice activo)
- Se etiquetan como visitados todos los adyacentes al vértice activo que no hayan sido etiquetados
- Se continúa etiquetando todos los adyacentes de los hijos de v (que no hayan sido visitados aun)

Recorrido BFS

Características:

Breadth-first search basa el orden de visita de los nodos del grafo en una estructura de datos Cola, incorporando:

- en cada paso los adyacentes al nodo actual
- esto implica que se visitarán todos los hijos de un nodo antes de proceder con sus demás descendientes
- las operaciones sobre la estructura Cola se suponen implementadas en tiempo O(1)

Recorrido BFS

BFS (G,s)

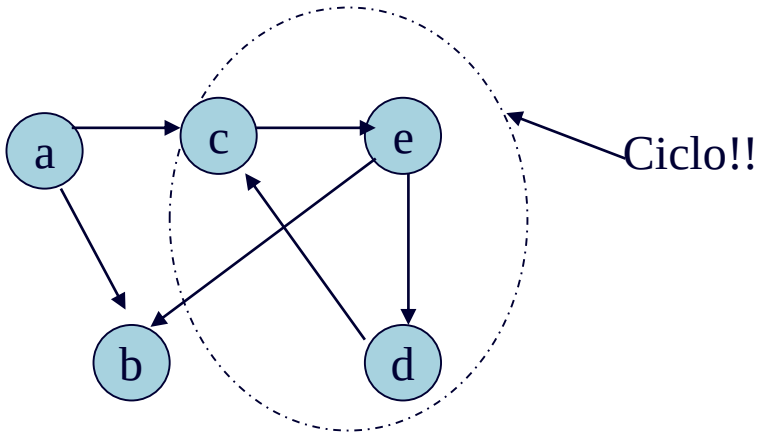
```
{for (u ε V(G)) // inicialmente todos los vértices no_visitados //
   estado[u]=no_visitado;
  estado [s]=visitado;
  encolar (Q,s);
  while (no_vacia (Q))
     {u=extraer (Q);  //extraer u de la cola Q y explora sus adyacentes //
       for (v ε adyacentes (u))
          {if (estado [v]= no_visitado)
                estado[v]= visitado;
                encolar (Q, v); }
       }
   }
```

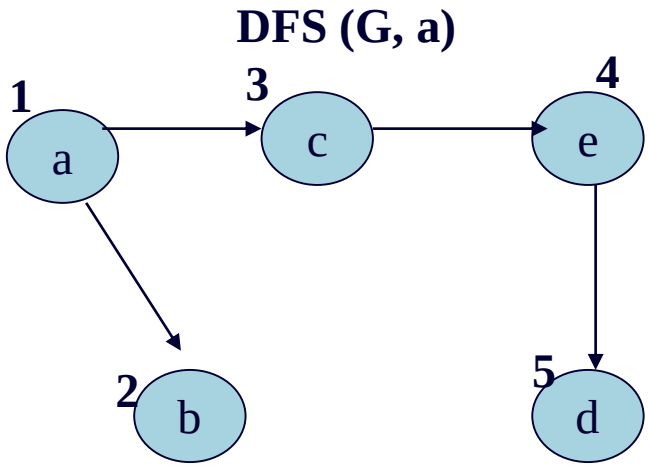
Propiedades del recorrido DFS

- Bosque de exploración: refleja las estructuras de las llamadas recursivas
- Clasificación de los arcos de un **grafo dirigido** G=(V,A).
  - **Arcos del árbol**: arcos (u,v) que conducen a vértices no visitados (marca[v]=NO_VISITADO)
  - **Arcos de retroceso** (backward): arcos (u,v) $\notin$ bosque donde v es antecesor de u en el árbol (marca[v]= DESCUBIERTO)
  - **Arcos de avance** (forward): arcos (u,v) $\notin$ bosque donde v es descendiente de u en el árbol (marca[v]=VISITADO y d[u]<d[v])
  - **Arcos cruzados**: los demás arcos (u,v) donde v no es ni antecesor ni sucesor en el árbol (marca[v]=VISISTADO y d[u]>d[v]

Aplicaciones del recorrido DFS. Grafo dirigido

Test de **aciclicidad**: verificar si un **grafo dirigido contiene ciclos**





Clasificamos todos los arcos según orden de visita de los vértices de DFS

Aplicaciones del recorrido DFS. Grafo dirigido

Test de aciclicidad

DFS (G,a)= a,b,c,e,d

arco	#visita	Tipo arco
(a,c)	1 - 3	Avance
(a,b)	1 - 2	Avance
(c,e)	3 - 4	Avance
(e,b)	4 - 2	Cruce
(e,d)	4 - 5	Avance
(d,c)	5 - 3	Retroceso

Genera ciclo ???

Genera ciclo

Aplicaciones del recorrido DFS. Grafo dirigido

Test de aciclicidad

Si pensamos en el ciclo => Todos los arcos (v,w) tales que: visitado [v] > visitado [w] generan ciclos???:

NO

Arco de cruce: misma propiedad del arco de retroceso, pero es un arco que cruza, no pertenece a la misma rama del árbol.

Luego, un grafo tiene ciclo si hay **arco de retroceso**, NO de cruce

Aplicaciones del recorrido DFS. Grafo dirigido

- Arco (e,b) del ejemplo es un "**arco cruzado**".

  Tiene la misma propiedad que el de retroceso pero no pertenece a la misma pila de ejecución

- Arco (d,c) del ejemplo es arco de retroceso:

  Origina **ciclo**

Aplicaciones del recorrido DFS. Grafo dirigido

Verificar que los arcos no pertenezcan a la misma pila de ejecución, cómo???

Una estrategia: además de asignar el #orden, se marca al vértice **a la salida** de llamado del DFS.

Aplicaciones del recorrido DFS. Grafo dirigido

```
int dfs (grafo<vertice,costo> g, vertice e)
{lista[vertice] l1;
  vertice e1;
  visitado[e]=nro_orden;
  nro_orden++;
  l1=g.sucesores(e);
  int ciclo=0;
  while (!l1.vacia_lista() && !ciclo)
      {e1=l1.recup_lista();
       if (!visitado[e1])
          ciclo=dfs(g,e1);
        else
          ciclo=(visitado[e1]<visitado[e] && !marcado[e1]);
       l1.elim_lista();
      }
  marcado[e]=1;}
```

Complejidad temporal
Test de Aciclicidad:
Complejidad temporal
de DFS

Aplicaciones del recorrido DFS. Grafo dirigido

Sort topológico: dado un **grafo dirigido acíclico** G, un sort u orden topológico es un ordenamiento lineal de sus vértices:

Se usa para planificar una serie de acciones que tienen precedencias:

- vértice representa una acción
- arco (u,v) significa que la acción u debe ejecutarse necesariamente antes que la acción v

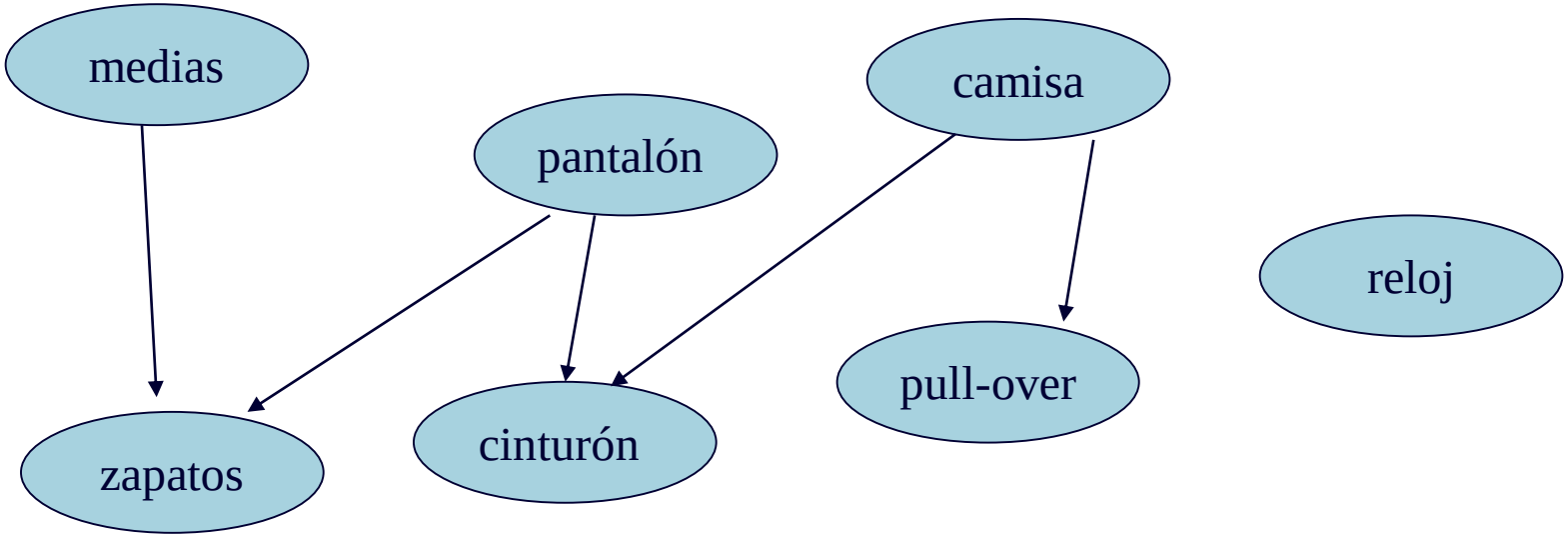
Aplicaciones del recorrido DFS. Grafo dirigido

Sort topológico

- (u,v) $\in$ G => u aparece antes que v en el ordenamiento
- si el grafo tiene ciclos => no existe el ordenamiento
- orden topológico: usado para planificar una serie de acciones que tienen precedencia

Aplicaciones del recorrido DFS. Grafo dirigido

Cada nodo representa una **acción**



la numeración de los nodos en un recorrido DFS da una idea del algoritmo para resolver el problema

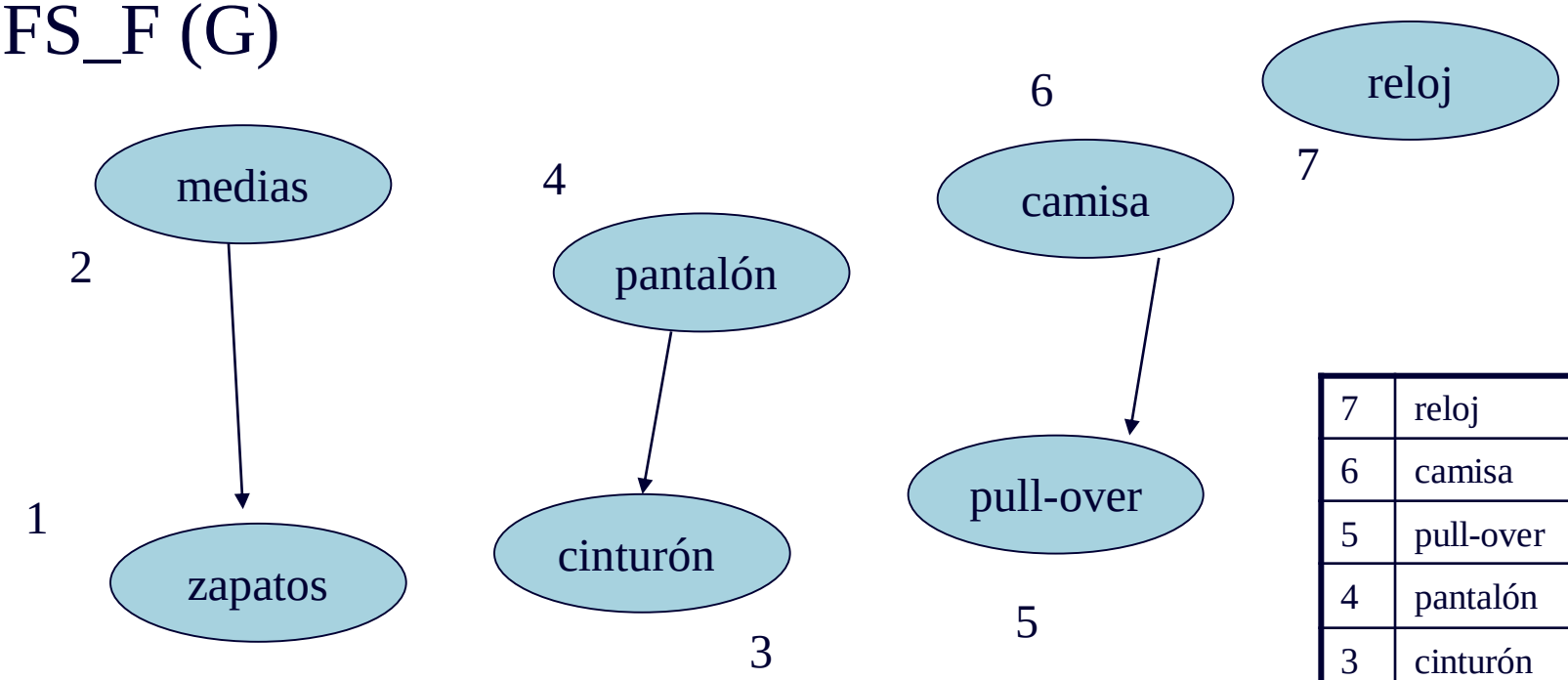
Aplicaciones del recorrido DFS. Grafo dirigido

Algoritmo:

- DFS Forest: marcando cada vértice con el número en post-orden (apilar)
- Explorados todos los vértices (a partir de un vértice dado) se numera antes de retroceder en la búsqueda

Aplicaciones del recorrido DFS. Grafo dirigido

DFS_F (G)



7	reloj
6	camisa
5	pull-over
4	pantalón
3	cinturón
2	medias
1	zapatos

Orden topológico: reloj, camisa, pull-over, pantalón, cinturón, medias, zapatos

Aplicaciones del recorrido DFS. Grafo dirigido

- Se listan los vértices según sus números en post-orden de mayor a menor
  - Se obtiene una secuencia válida para las acciones
- Complejidad temporal: O(max(v,a))