Diseño de algoritmos

Recorridos de grafos

Jesús Bermúdez de Andrés

Universidad del País Vasco/Euskal Herriko Unibertsitatea **(UPV/EHU)**

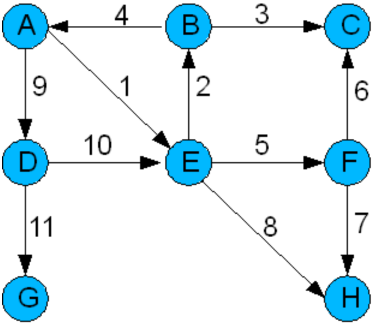
Curso 2008-09

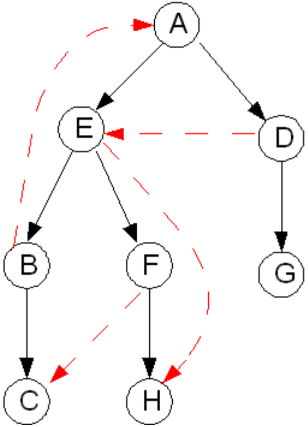
1. Recorridos de grafos
   - Recorrido en profundidad
   - Recorrido en anchura
   - Ordenación topológica

Dos representaciones básicas de grafos

- Mediante **matriz de adyacencia**: disponemos de una numeración $1, \ldots, n$ de los nodos del grafo y consiste en una matriz cuadrada $A = (a_{ij})$, indexada por los nodos, tal que $a_{ij} = 1$ si la arista (i, j) está en el grafo y $a_{ij} = 0$ si no.
  - Si la matriz tiene pesos asociados a las aristas entonces la matriz de adyacencia $P = (p_{ij})$ tiene en p_{ij} el peso asociado a la arista (i, j) y un valor especial (por ejemplo ∞) si no existe la arista (i, j).
- Mediante **listas de adyacencias**, consiste en un vector de listas, indexado por los nodos numerados $1, \ldots, n$, tal que para cada nodo i tenemos acceso a una lista con todos sus nodos adyacentes; es decir, el nodo j está en la lista de i si y sólo si la arista (i, j) está en el grafo.
  - Cuando hay pesos asociados a las aristas, el valor de ese peso está almacenado junto al nodo correspondiente.

Recorrido en profundidad





La numeración de las aristas indica el orden en el que se van visitando los nodos, comenzando desde el nodo A

Esquema de recorrido en profundidad

```
proc RecorridoEnProfundidad (G = (N, A))
   for cada v ∈ N loop marca(v)← falso end loop
   for cada v ∈ N loop
      if ¬ marca(v) then MarcaProf(v)

proc MarcaProf (v)
   marca(v)← verdadero
   for cada w ∈ N adyacente de v loop
      if ¬ marca(w) then MarcaProf(w)
```

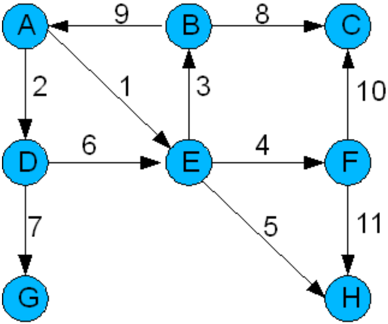
Análisis: Sea n el número de nodos del grafo y a el número de aristas

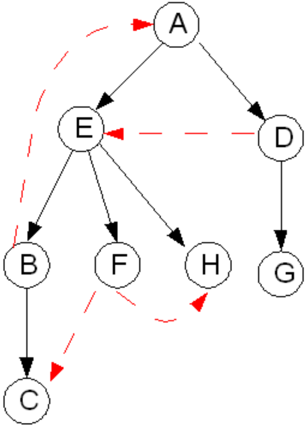
- $\Theta(\mathrm{n}+\mathrm{a})$ si el grafo se representa con listas de adyacencias
- $\Theta(n^2)$ si el grafo se representa con matriz de adyacencia
- espacio extra de $\Theta(\mathrm{n})$, en cualquier caso

Esquema general para MarcaProf

```
proc Marca_Prof_Gen(v)
   [Procesar v en primera visita]
   [Como en preorden]
   marca(v) ← cierto
   for cada w ∈ N adyacente de v loop
      [Procesar arista (v, w)]
      if ¬ marca(w) then
         [Procesar arista (v, w) del árbol]
         Marca_Prof_Gen(w)
         [Procesar v al regreso de procesar w]
         [Como en inorden]
      else
         [Procesar arista (v, w). NO es del árbol]
   end for
   [Procesar v al abandonarlo]
   [Como en postorden]
```

Recorrido en anchura





La numeración de las aristas indica el orden en el que se van visitando los nodos, comenzando desde el nodo A

Esquema de recorrido en anchura

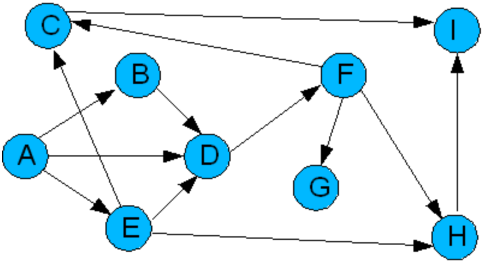
proc RECORRIDOENANCHURA $(G = (N, A))$
  for cada $v \in N$ **loop** marca(v)$\leftarrow$ falso **end loop**
  for cada $v \in N$ **loop**
    if $\neg$ marca(v) **then** MARCAANCHO(v)

proc MARCAANCHO (v)
  $C \leftarrow$ **new** Cola
  marca(v) $\leftarrow$ verdadero
  C.insert(v)
  while $\neg$ C.is_empty() **loop**
    $u \leftarrow C$.remove_first()
    for cada $w \in N$ adyacente de u **loop**
      if $\neg$ marca(w) **then**
        marca(w) $\leftarrow$ verdadero
        C.insert(w)
  end while

Análisis: Igual que con el esquema anterior

Ordenación topológica (1/2)

Una **ordenación topológica** de un grafo dirigido *acíclico* $G = (N, A)$ es una lista de los nodos del grafo tal que si la arista (u, v) aparece en A entonces el nodo u aparece antes que v en la lista resultado.



Distintas ordenaciones topológicas del grafo:

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

Ordenación topológica (2/2)

```
func ORDENACIÓN_TOPOLÓGICA (G = (N, A)) return Lista_de_nodos
   L ← new Lista
   for cada v ∈ N loop marca(v) ← falso end loop
   for cada v ∈ N loop
      if ¬ marca(v) then ORDENTOPO(v, L)
   end for
   return L

proc ORDENTOPO (v, L)
   marca(v) ← verdadero
   for cada w ∈ N adyacente de v loop
      if ¬ marca(w) then ORDENTOPO(w, L)
   end for
   L.insert_first(v)
```