Tema 6. Recorridos. Conectividad.

Recorridos
- Recorrido en Amplitud (BFS)
- Recorrido en Profundidad (DFS)

Conectividad
- Cálculo de las componentes conexas

Conexión:
buscador social
en Facebook

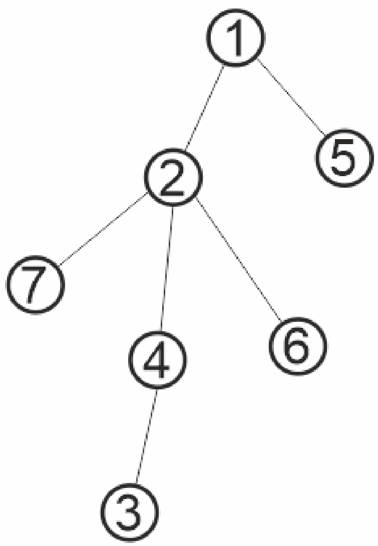
Recorridos

Un recorrido en un grafo es un método que nos permite visitar todos los nodos accesibles, partiendo de uno dado, y viajando por la adyacencia o por los sucesores o por los predecesores.
El esquema general que se muestra viaja por la adyacencia, usa un vector de marcas, visitado, para saber qué nodos ya he visitado, y un conjunto ToDo donde sitúo las tareas pendientes: nodos accedidos aún pendientes de ser explorados.

Esquema general de un recorrido: usando marcas de visitas y lista de nodos por visitar

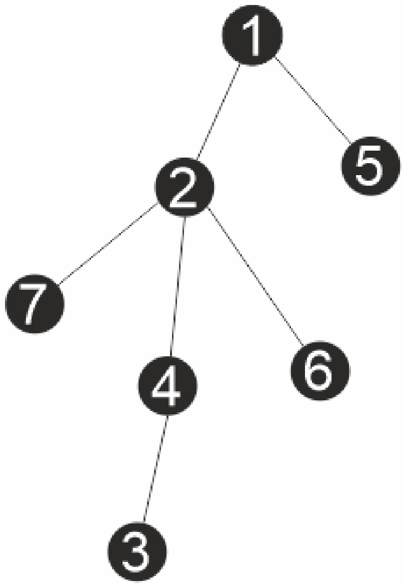
```
{inicialización}
Para todo nodo v, visitado[v] = falso;
{preparamos el inicio del recorrido desde el nodo i}
Visitado[i] = verdadero; ToDo ={i};
{bucle principal}
Mientras ToDo no vacío hacer
        Sea k en ToDo
        ToDo = ToDo – {k}
        Para todo adyacente j de k hacer
                Si visitado[j] = falso entonces
                        Visitado[j]=verdadero;
                        ToDo = ToDo + {j}
```

Recorridos



$\Gamma_1 = \{2,5\}$
$\Gamma_2 = \{1,7,4,6\}$
$\Gamma_3 = \{4\}$
$\Gamma_4 = \{2,3\}$
$\Gamma_5 = \{1\}$
$\Gamma_6 = \{2\}$
$\Gamma_7 = \{2\}$

Vamos a realizar una traza de este procedimiento sobre el siguiente grafo, partiendo del nodo 1.



$\Gamma_1 = \{2, 5\}$

$\Gamma_2 = \{1, 7, 4, 6\}$

$\Gamma_3 = \{4\}$

$\Gamma_4 = \{2, 3\}$

$\Gamma_5 = \{1\}$

$\Gamma_6 = \{2\}$

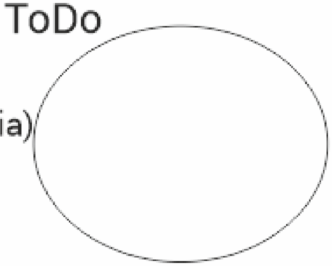
$\Gamma_7 = \{2\}$

```
Mientras ToDo no vacío hacer
        Sea k en ToDo
        ToDo = ToDo – {k}
        Para todo adyacente j de k hacer
            Si visitado[j] = falso entonces
                Visitado[j]=verdadero;
                ToDo = ToDo + {j}
```

- Todos los nodos no visitados
- Comenzamos desde el nodo 1
- ToDo ={1}
- Recorremos la adyacencia del nodo k=1
- 3 nodos ya visitados; 2 nodos en ToDo, no vacío
- Recorremos la adyacencia del nodo k=5 (no hay preferencia)
- no se visita ningún nodo nuevo, pues el nodo 1 ya visitado
- Recorremos la adyacencia del nodo k=2
- 6 nodos ya visitados; 3 nodos en ToDo, no vacío
- Recorremos la adyacencia del nodo k=7 (no hay preferencia)
- no se visita ningún nodo nuevo, pues el nodo 2 ya visitado
- Recorremos la adyacencia del nodo k=6 (no hay preferencia)
- no se visita ningún nodo nuevo, pues el nodo 2 ya visitado
- Recorremos la adyacencia del nodo k=4
- se visita el nodo 3, y se incorpora a ToDo
- Recorremos la adyacencia del nodo k=3
- no se visita ningún nodo nuevo, pues el nodo 4 ya visitado
- ToDo queda vacío: FIN
- Se han visitado los nodos accesibles en este orden:

1->2->5->7->4->6->3

ToDo



Recorridos

Ahora veremos dos tipos recorridos: en el primero, la gestión del conjunto ToDo se hará como si fuera una cola (el primero que entra es el primero que sale), dando lugar al recorrido en amplitud; en el segundo caso, la gestión del ToDo será mediante una pila (el primero que entra es el ultimo en salir), dando lugar al recorrido en profundidad.

Recorrido en Amplitud o búsqueda del primero en Anchura (BFS).

Dado un Grafo G=(V, A) y un nodo distinguido s, el recorrido en amplitud explora de manera sistemática los arcos de G para *descubrir* todo nodo que es alcanzable desde s mediante un secuencia de nodos. Un recorrido en amplitud produce un árbol (o bosque si el grafo no es conexo) cuya raíz es el nodo s y contiene el arco o arista (i, j) si el nodo j es “descubierto” estudiando el nodo i. En este caso se dice que i es el predecesor de j en el árbol (pred[j]=i).

Recorridos

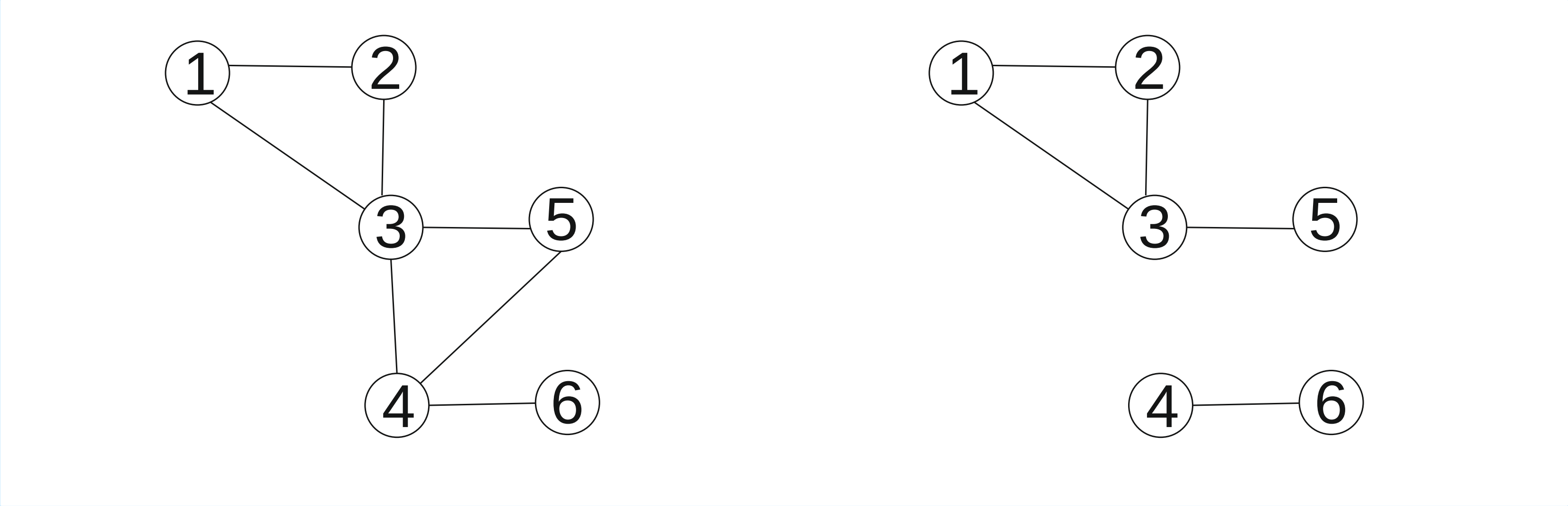
Recorrido en Profundidad o búsqueda del primero en Profundidad (DFS).

El DFS busca en "profundidad" en el grafo siempre que sea posible. La regla es explorar la lista de sucesores/predecesores/adyacentes del nodo *v* más recientemente descubierto que todavía tiene por explorar. Cuando todos los sucesores de uno han sido descubiertos, el método retrocede para examinar otros nodos que permitieron alcanzar a *v* (recursión). En un recorrido en profundidad cada nodo visitado es susceptible de ser caracterizado en dos instantes: La primera vez que se descubre y cuando se finaliza la examinación de todos sus sucesores. Llamaremos *prenumeración* al primer instante y *postnumeración* al segundo.

Conectividad

Conectividad: decimos que dos nodos i y j, están conectados si el grafo contiene al menos una cadena con extremos los nodos i y j.
Decimos que un grafo es conexo si cualquier par de nodos está conectados, y en otro caso diremos que es no conexo.
Llamaremos componentes conexas de un grafo no conexo a los subgrafos conexos de él.

En el ejemplo de la diapositiva, a la izquierda mostramos un grafo conexo, y a la derecha un grafo no conexo, con dos componentes conexas, {1, 2, 3, 5} y {4, 6}



Conectividad

Cálculo de las componentes conexas:

Para la construcción de las componentes conexas podemos usar cualquier recorrido: sólo necesitamos que se visiten los nodos viajando por la adyacencia. En el caso del dfs, tenemos:

```
Procedimiento ComponentesConexas(G) {
componentesconexas = 0;
Para i ← 1 hasta n Hacer
      Marca[i] ← False;
i ← 1;
Mientras (i <= n) {
      Si (Marca[i] = false) entonces {
            componentesconexas ← componentesconexas + 1;
            Escribir "Componente Conexa número componentesconexas {";
            dfs(G,i,Marca);
  }
  i← i + 1;
}
```