Algoritmos y Estructuras de Datos II

Recorriendo grafos

Clase de hoy

1. Repaso

2. Recorrida de grafos
   - Generalidades
   - Árboles binarios

Repaso

- cómo vs. qué
- 3 partes
  1. análisis de algoritmos
  2. tipos de datos
  3. técnicas de resolución de problemas
     - divide y vencerás
     - algoritmos voraces
     - backtracking
     - programación dinámica: problema de la moneda, problema de la mochila
     - recorrida de grafos

Clase de hoy

Recorrida de grafos

Recorrer un grafo, significa **procesar** los vértices del mismo, de forma organizada de modo de asegurarse:

- que todos los vértices sean **procesados**,
- que ninguno de ellos sea **procesado** más de una vez.

Se habla de **procesar** los vértices, pero también utilizaremos la palabra **visitar** los vértices. En este contexto, son sinónimos. Puede haber más de una forma natural de recorrer un cierto grafo.

Clase de hoy

Recorrida de árboles binarios

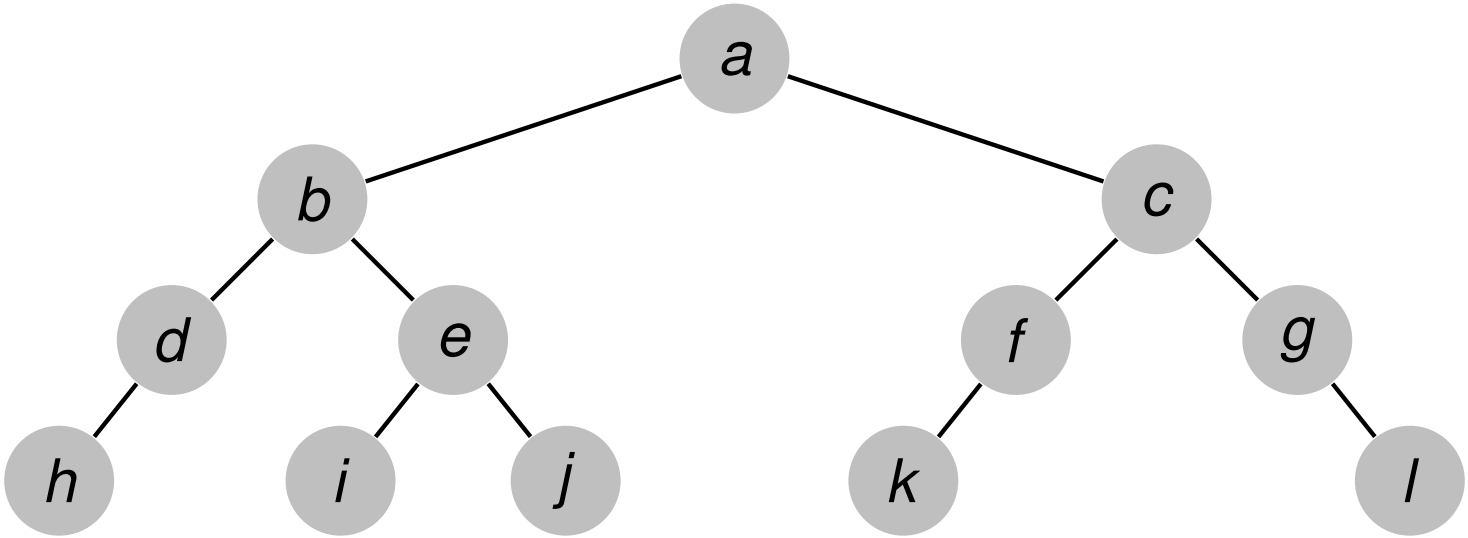
Este año veremos solo el caso de recorrida de árboles binarios. Hay unas 3 primeras maneras de **recorrerlo**:

pre-order Se **visita** primero el elemento que se encuentra en la raíz, luego se **recorre** el subárbol izquierdo y finalmente se **recorre** el subárbol derecho.

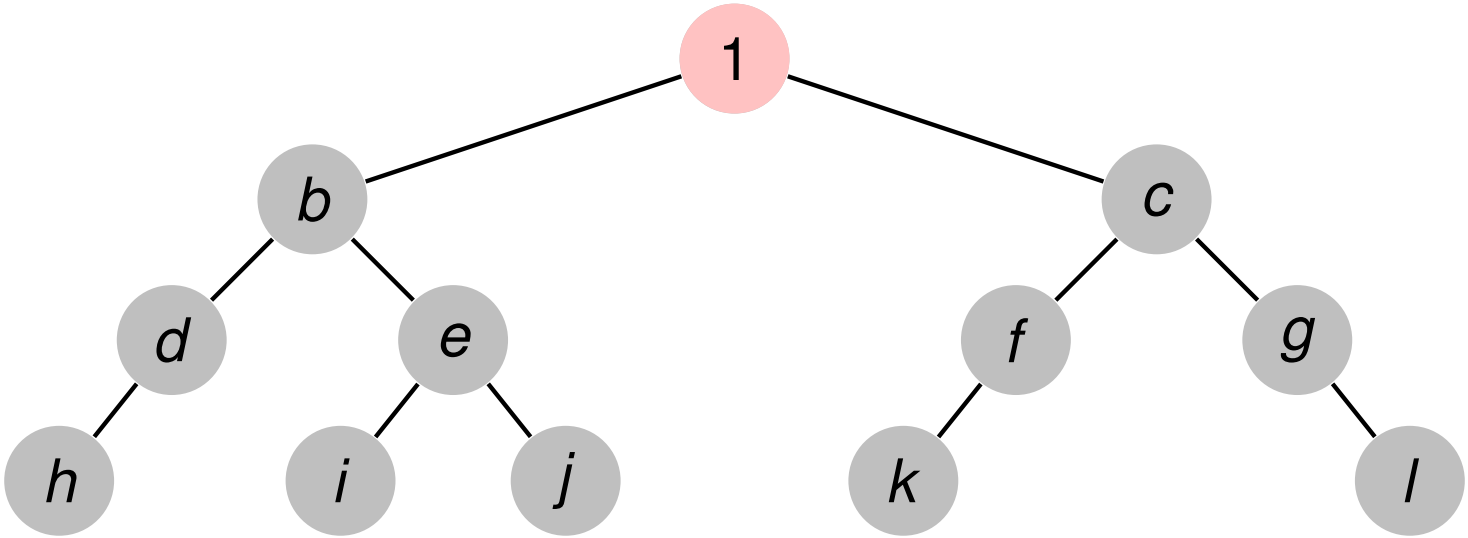
in-order Se **recorre** el subárbol izquierdo, luego se **visita** el elemento que se encuentra en la raíz y finalmente se **recorre** el subárbol derecho.

pos-order Se **recorre** el subárbol izquierdo, luego el derecho y finalmente se **visita** el elemento que se encuentra en la raíz.

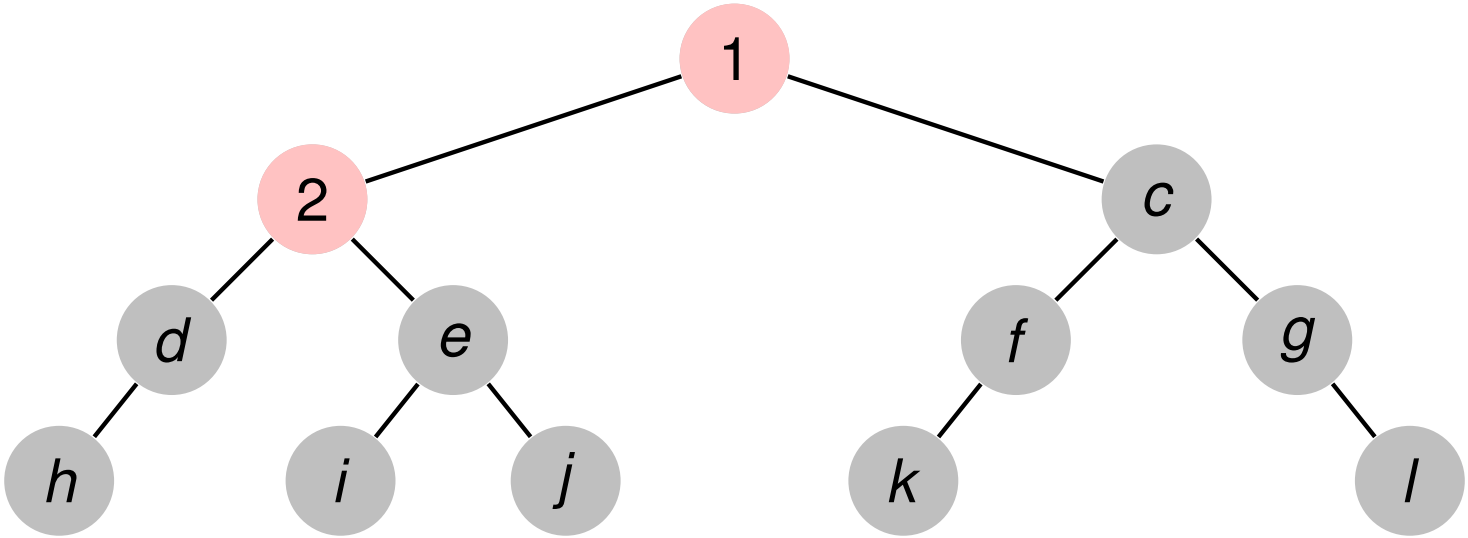
Ejemplo de árbol binario



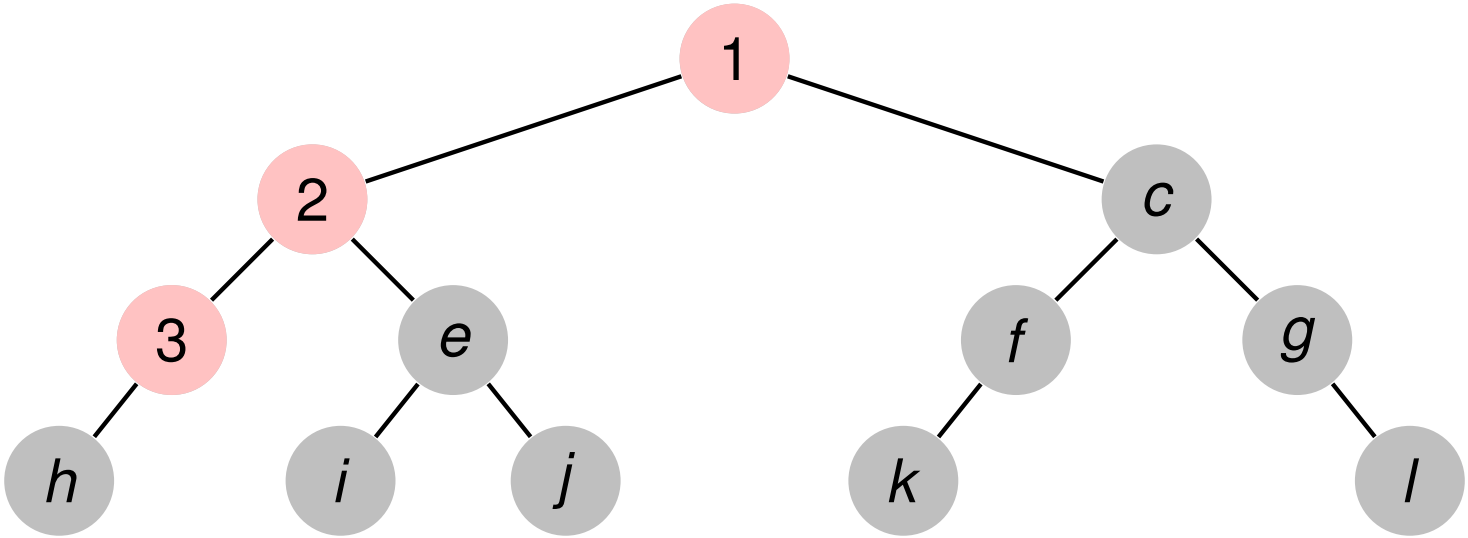
Ejemplo, recorrida pre-order



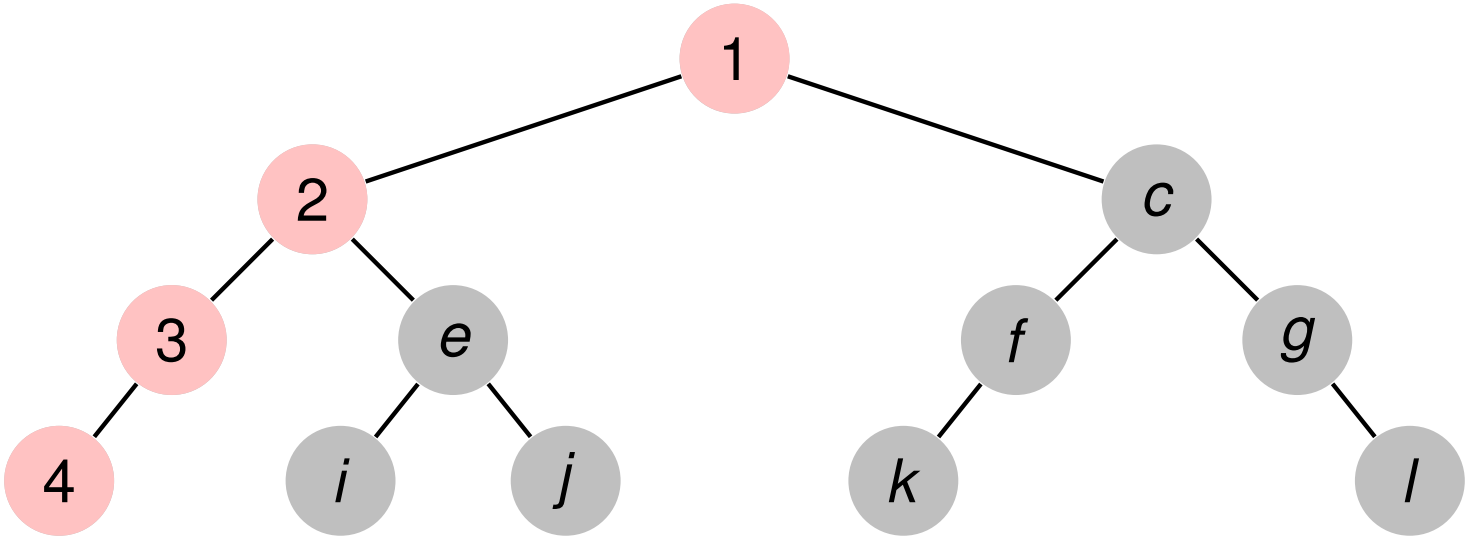
Ejemplo, recorrida pre-order



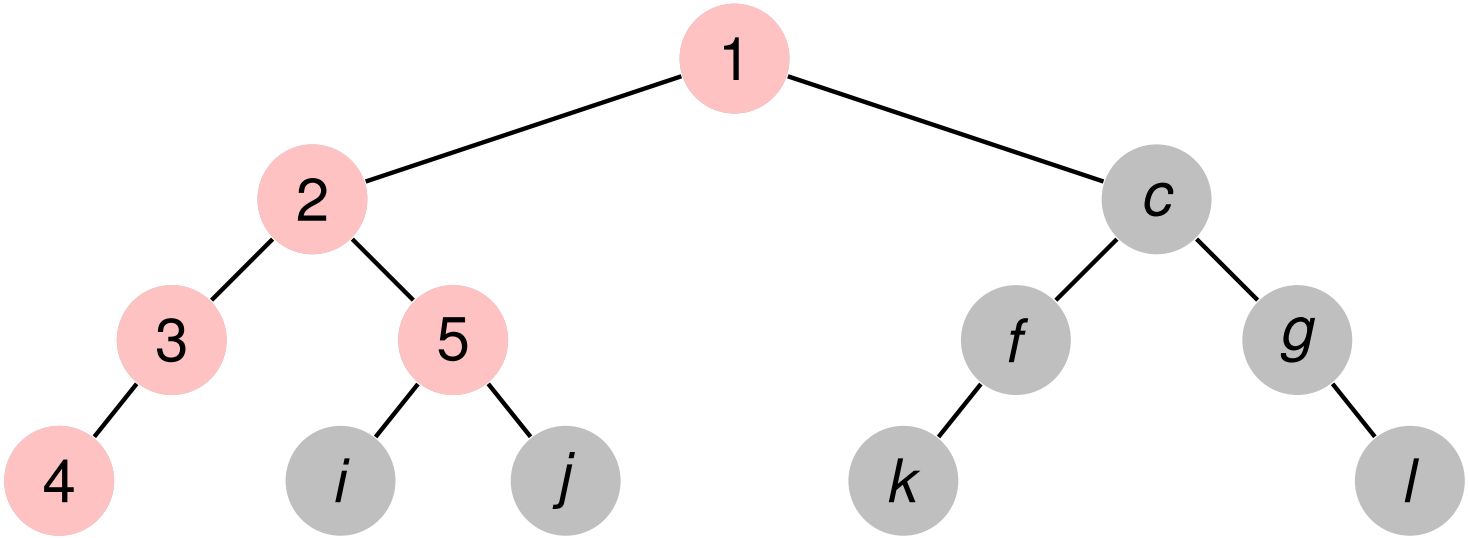
Ejemplo, recorrida pre-order



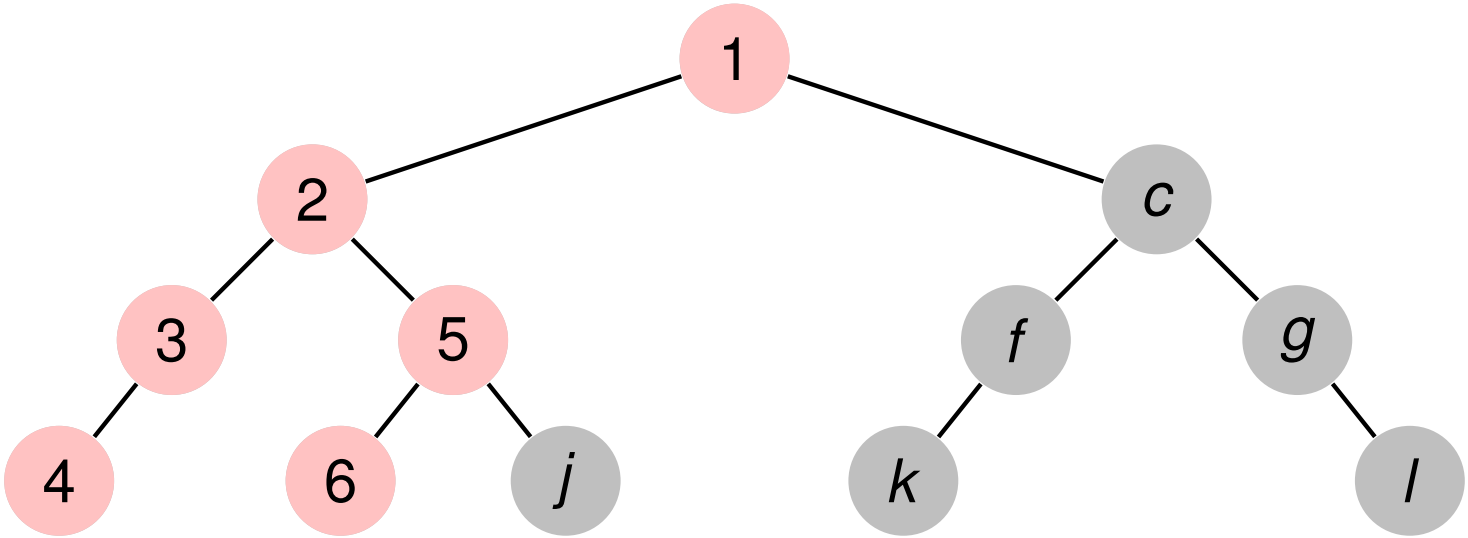
Ejemplo, recorrida pre-order



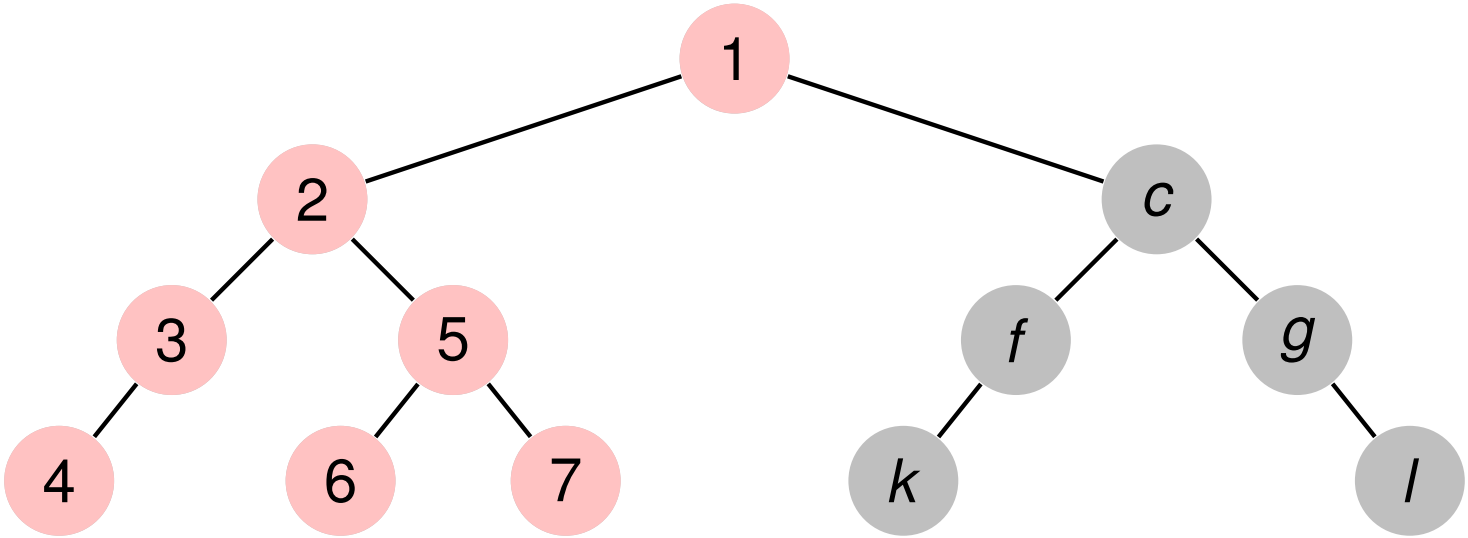
Ejemplo, recorrida pre-order



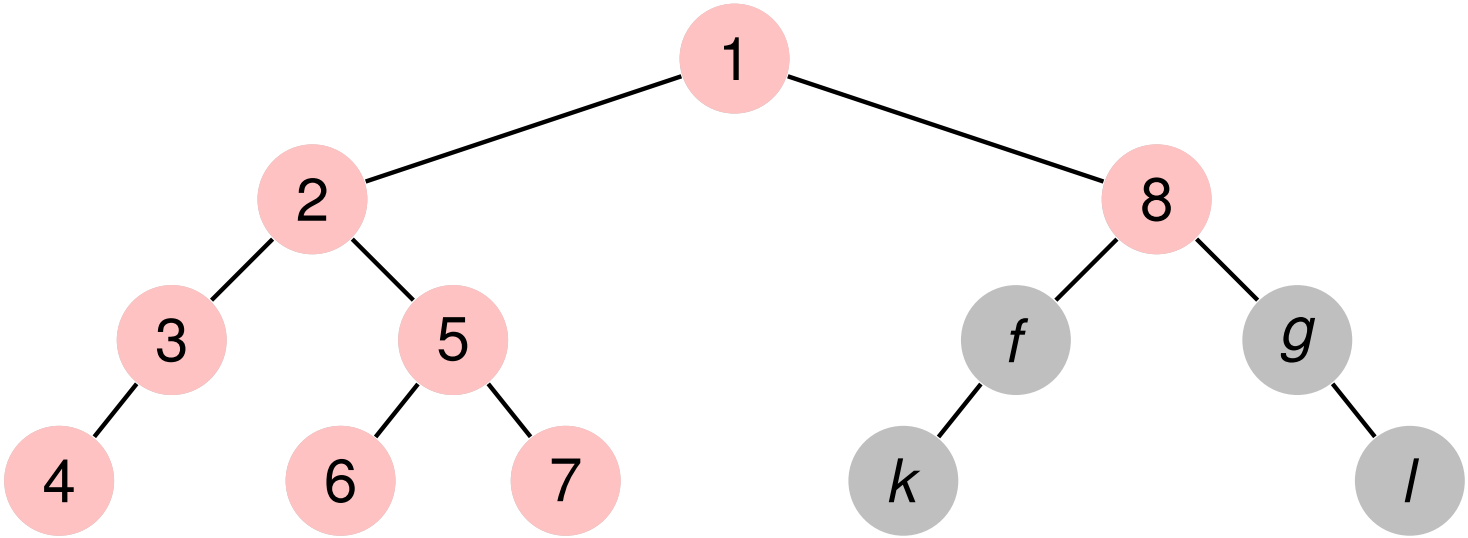
Ejemplo, recorrida pre-order



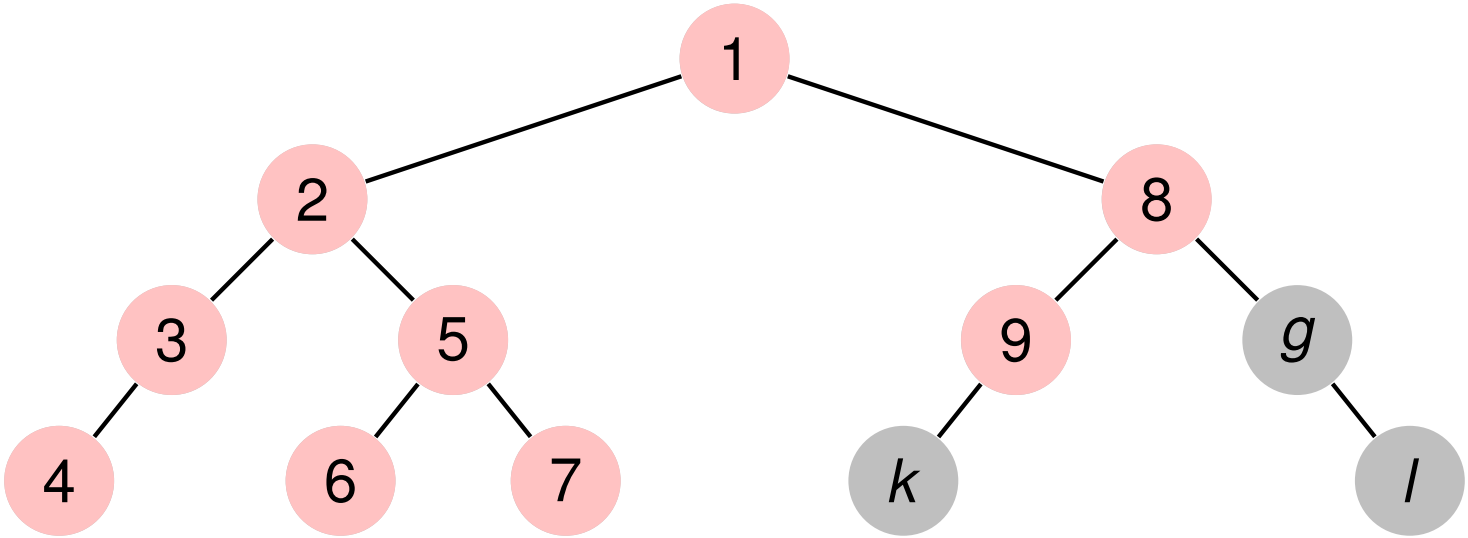
Ejemplo, recorrida pre-order



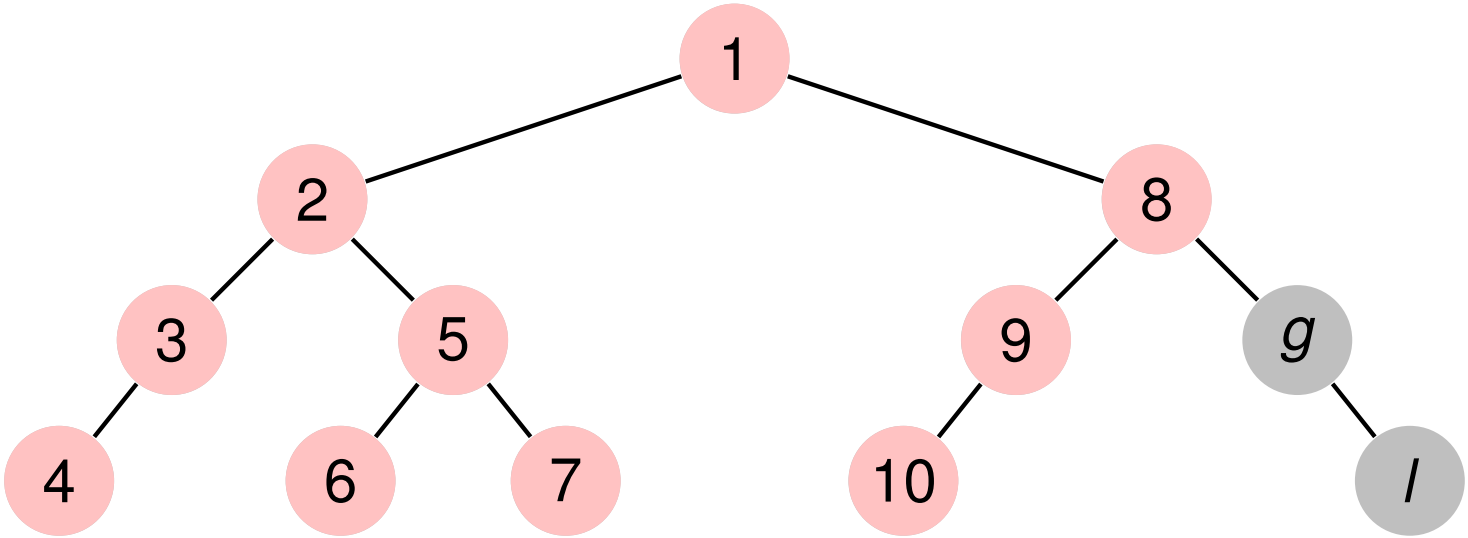
Ejemplo, recorrida pre-order



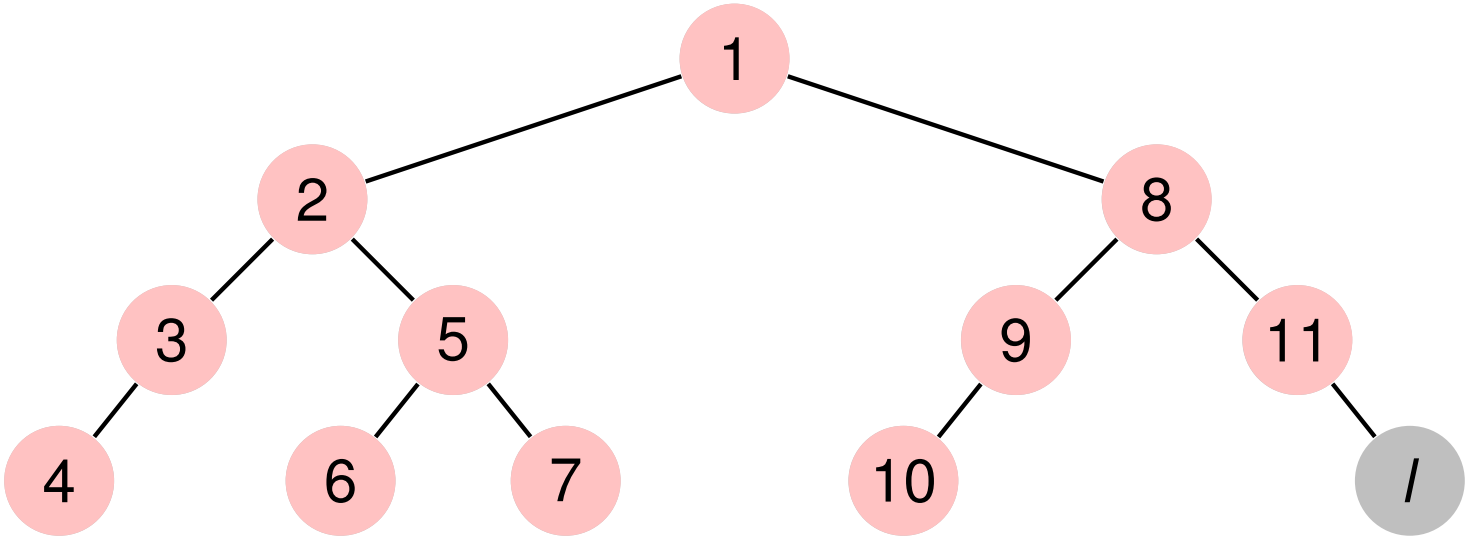
Ejemplo, recorrida pre-order



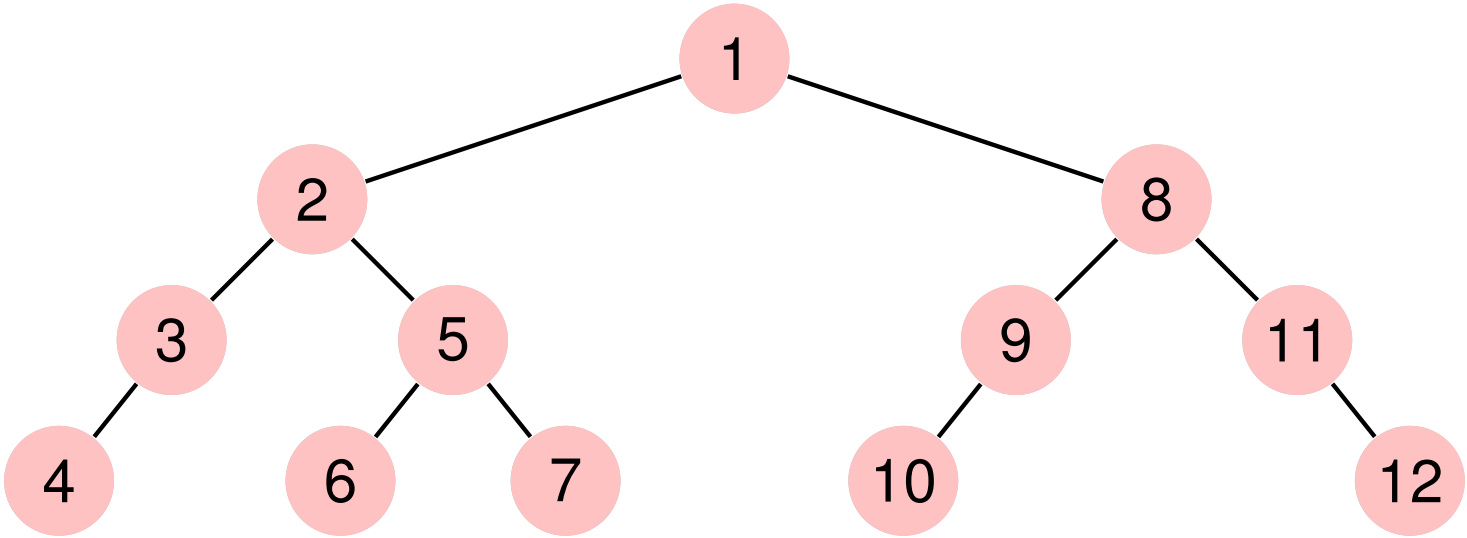
Ejemplo, recorrida pre-order



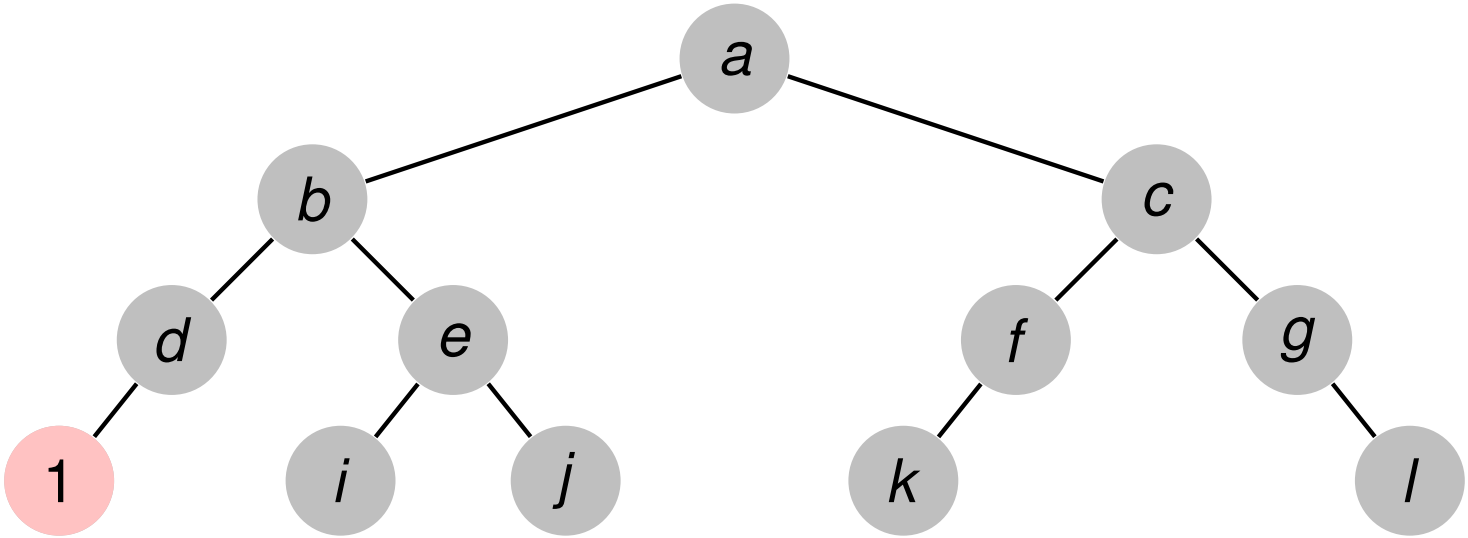
Ejemplo, recorrida pre-order



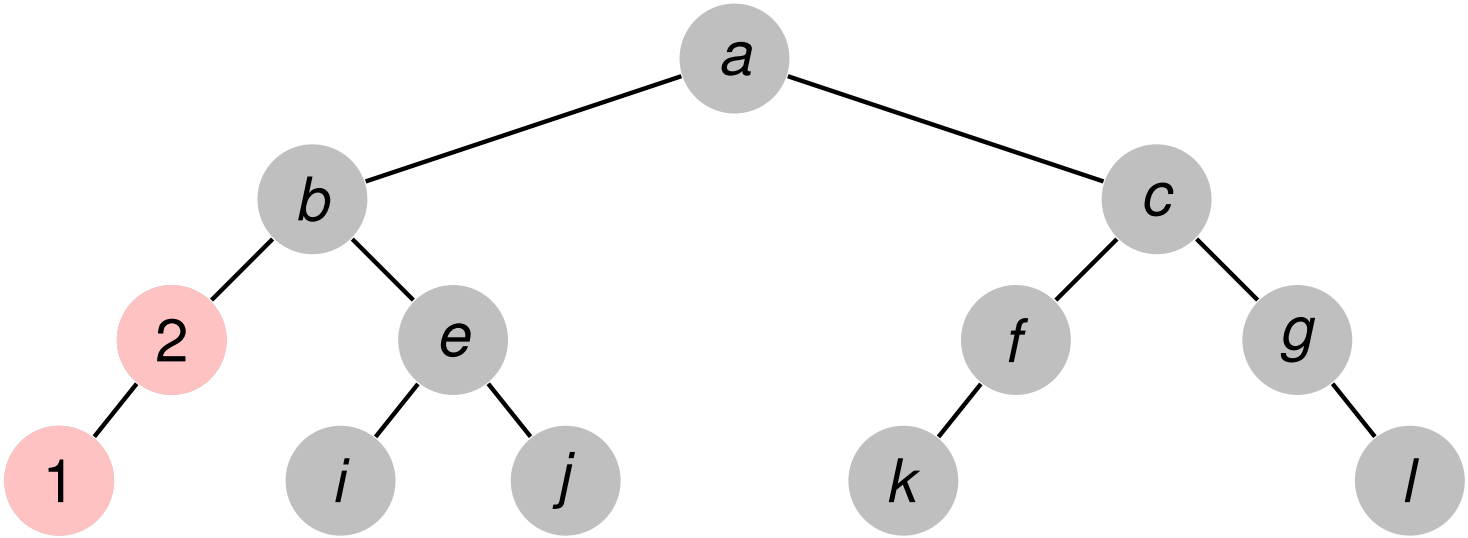
Ejemplo, recorrida pre-order



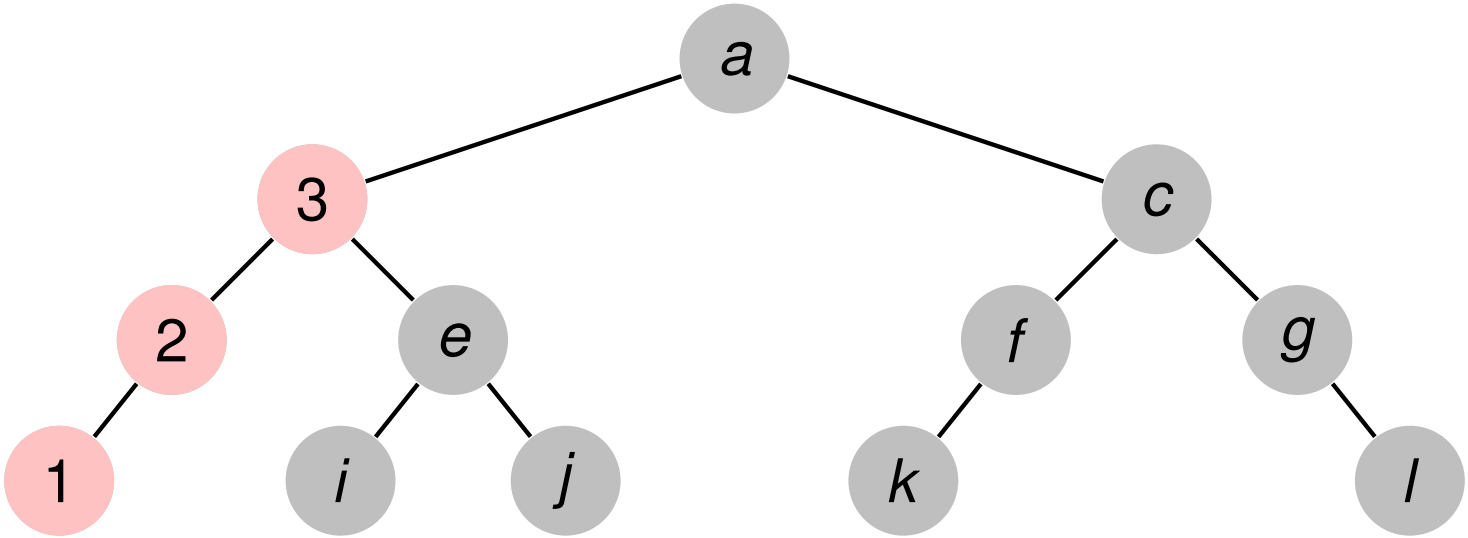
Ejemplo, recorrida in-order



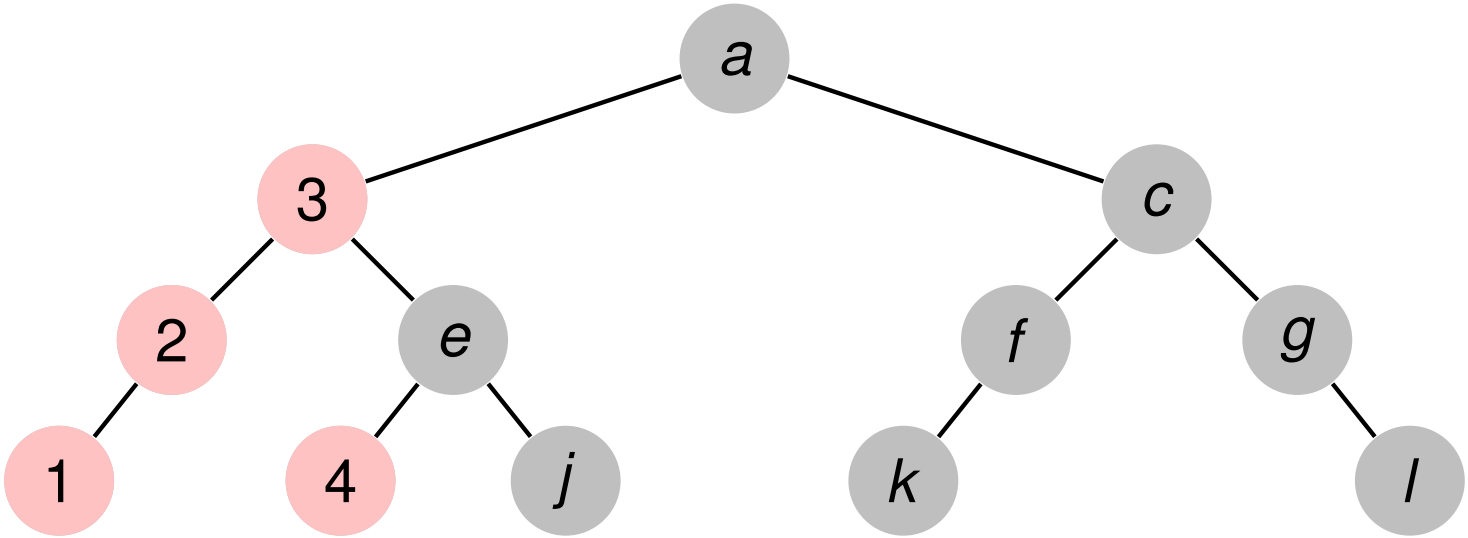
Ejemplo, recorrida in-order



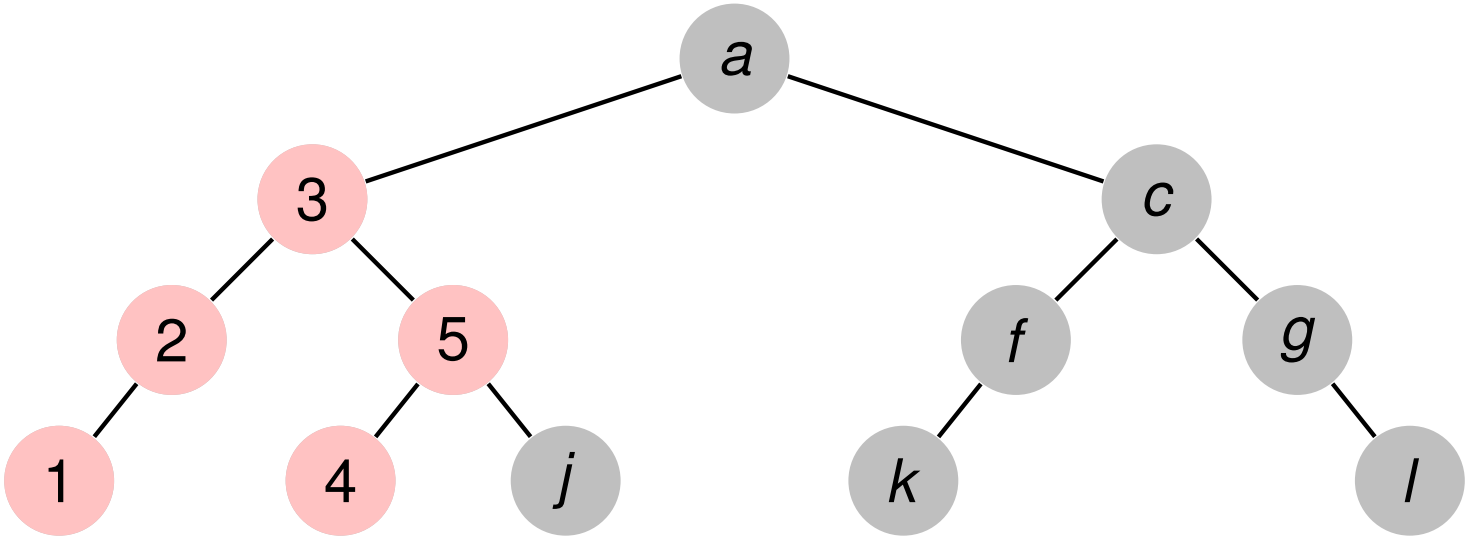
Ejemplo, recorrida in-order



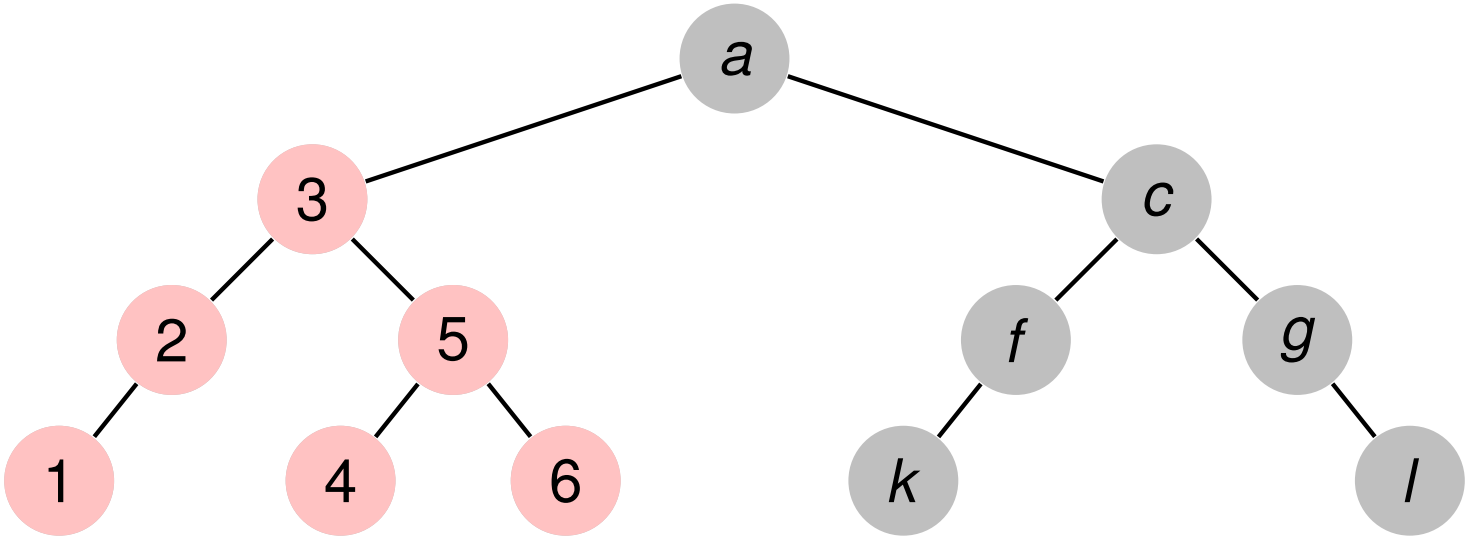
Ejemplo, recorrida in-order



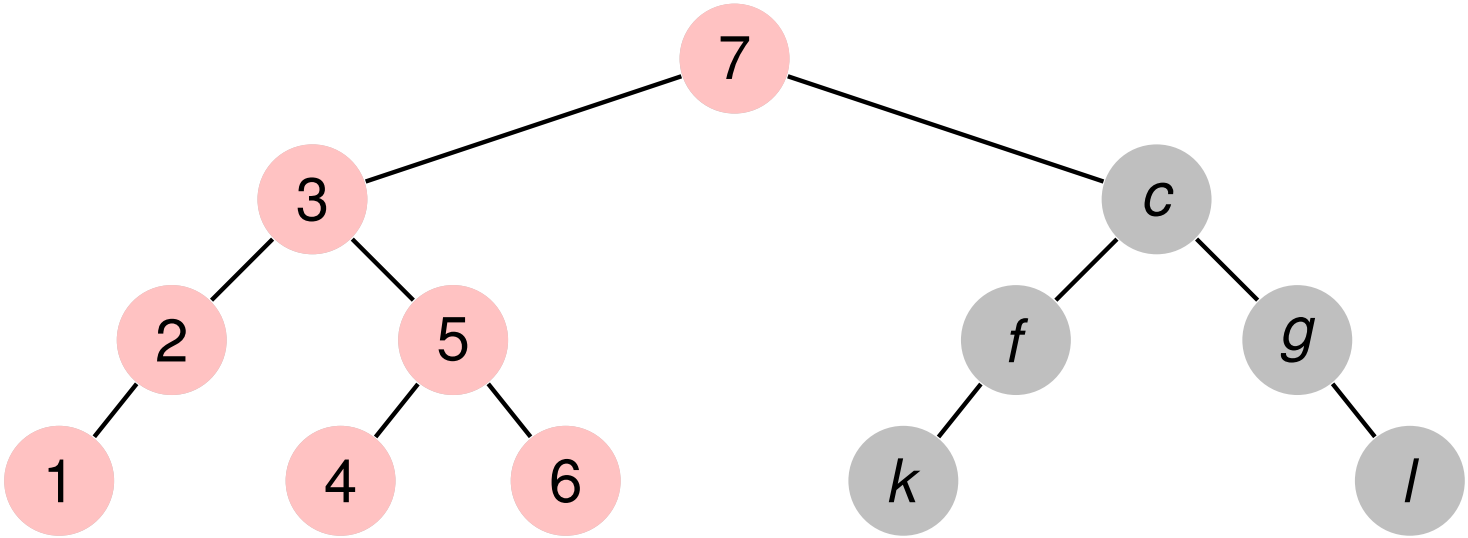
Ejemplo, recorrida in-order



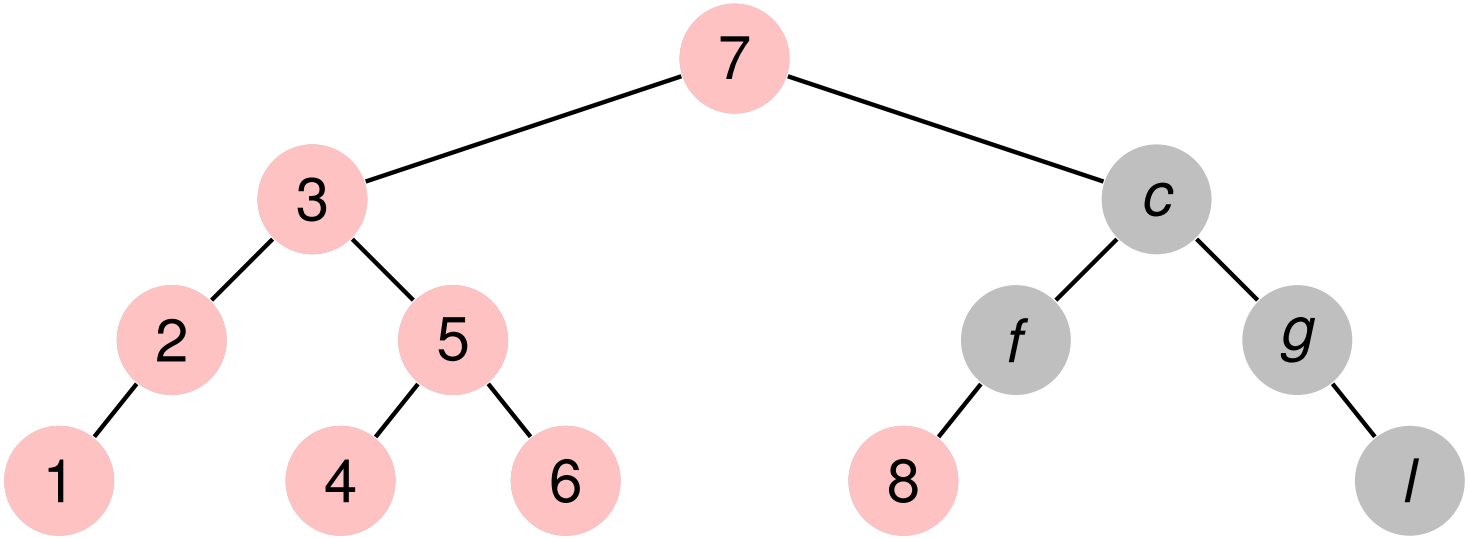
Ejemplo, recorrida in-order



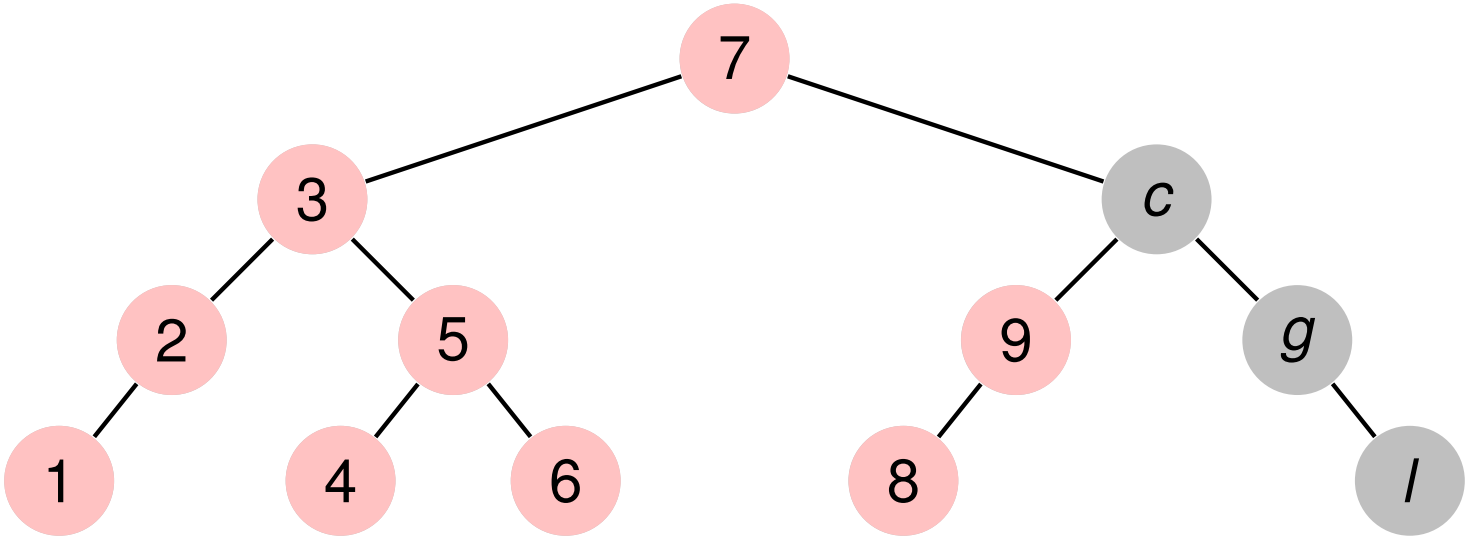
Ejemplo, recorrida in-order



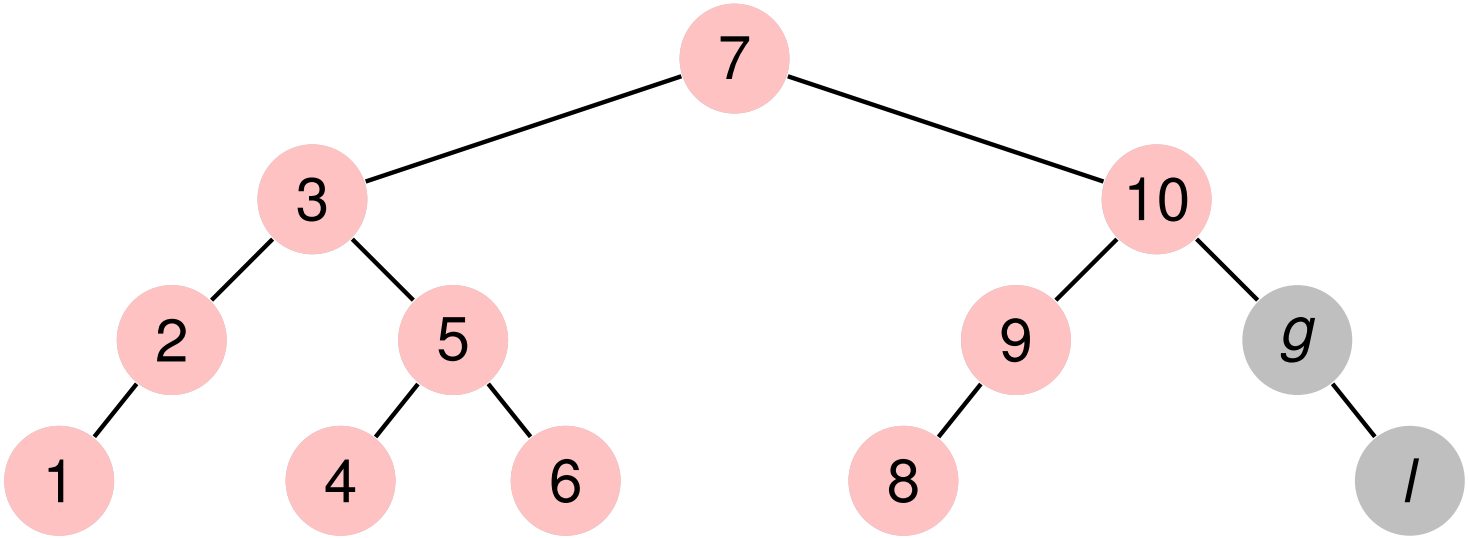
Ejemplo, recorrida in-order



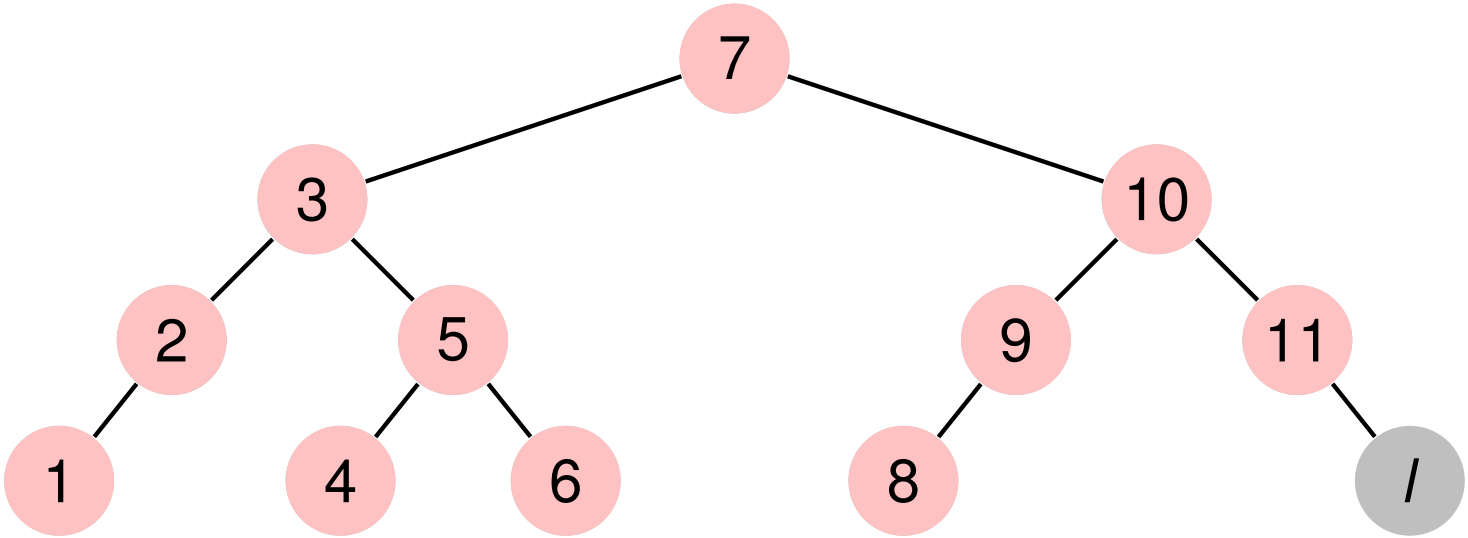
Ejemplo, recorrida in-order



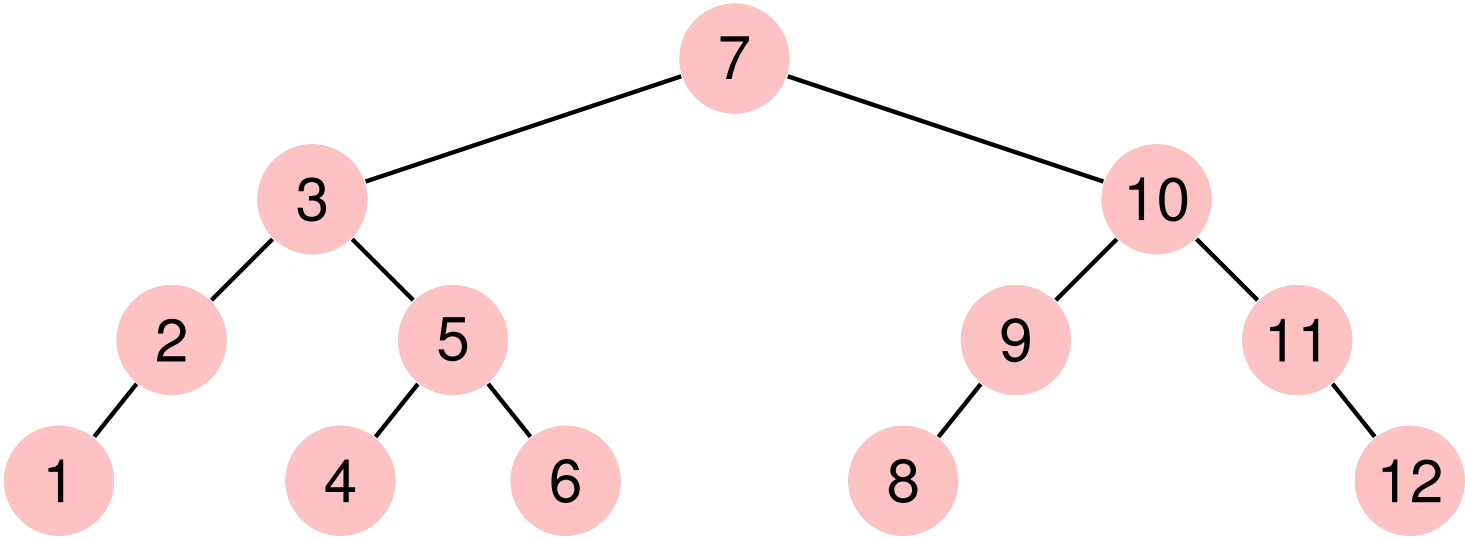
Ejemplo, recorrida in-order



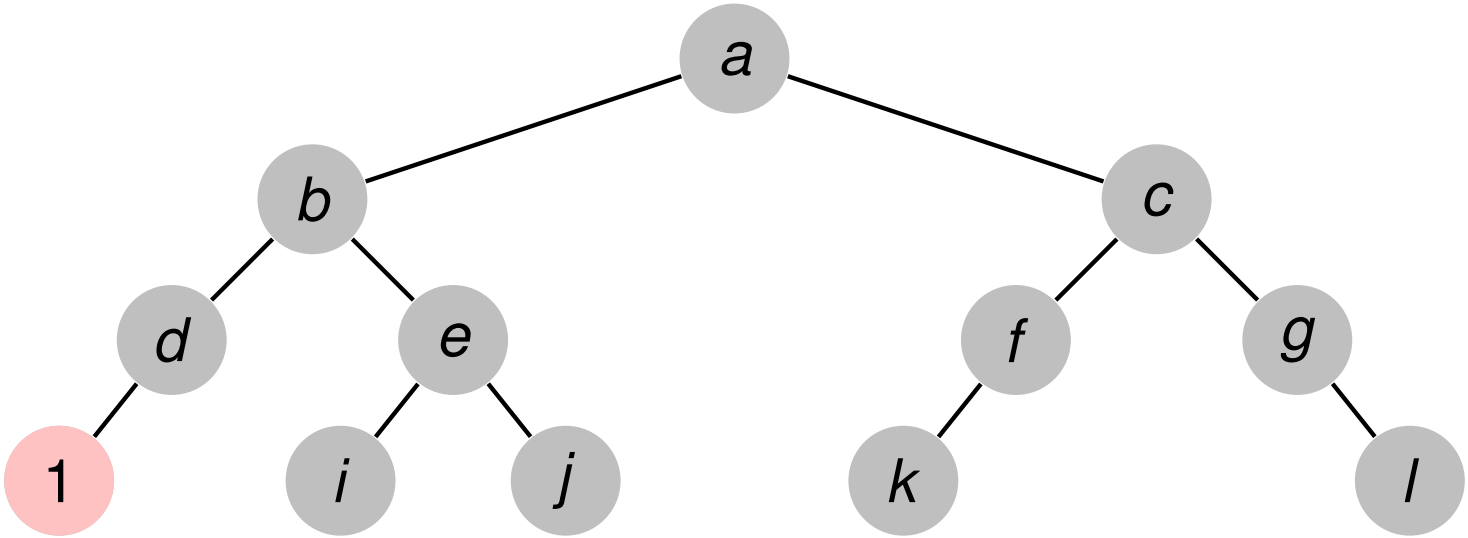
Ejemplo, recorrida in-order



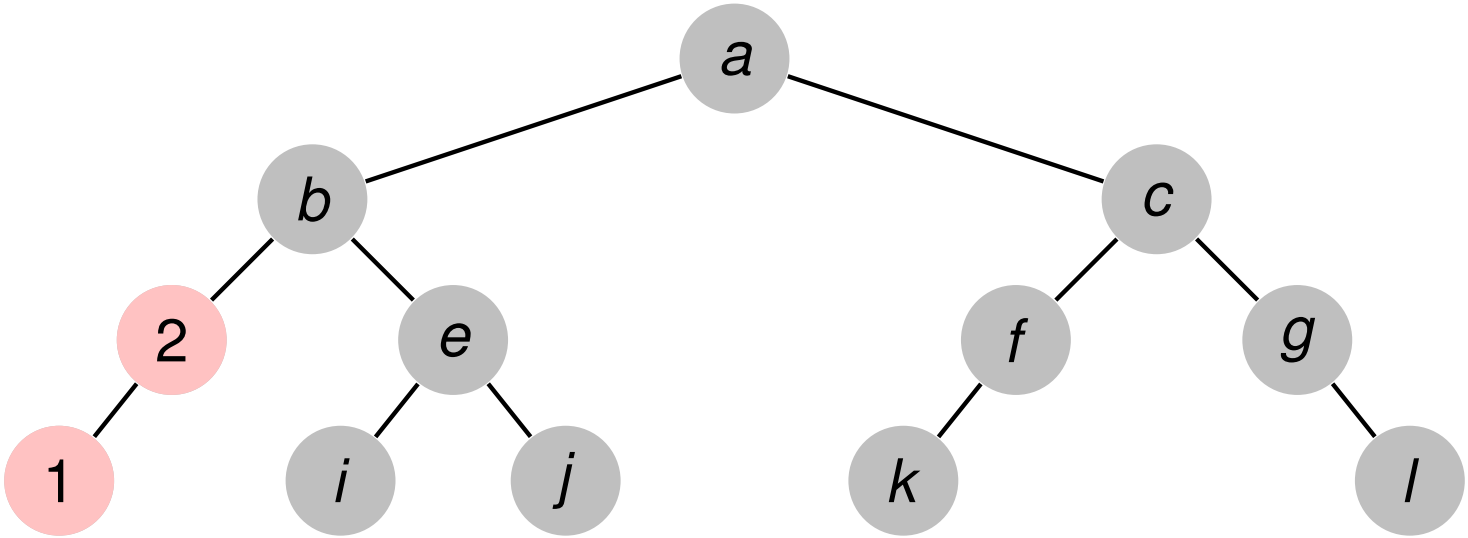
Ejemplo, recorrida in-order



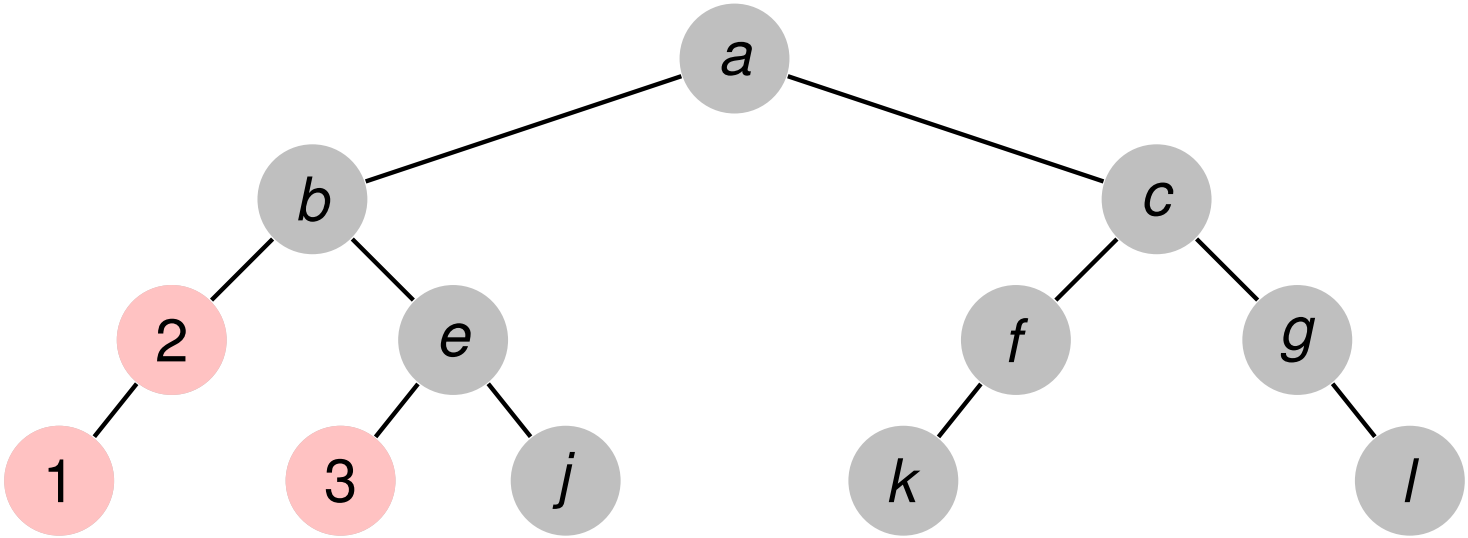
Ejemplo, recorrida pos-order



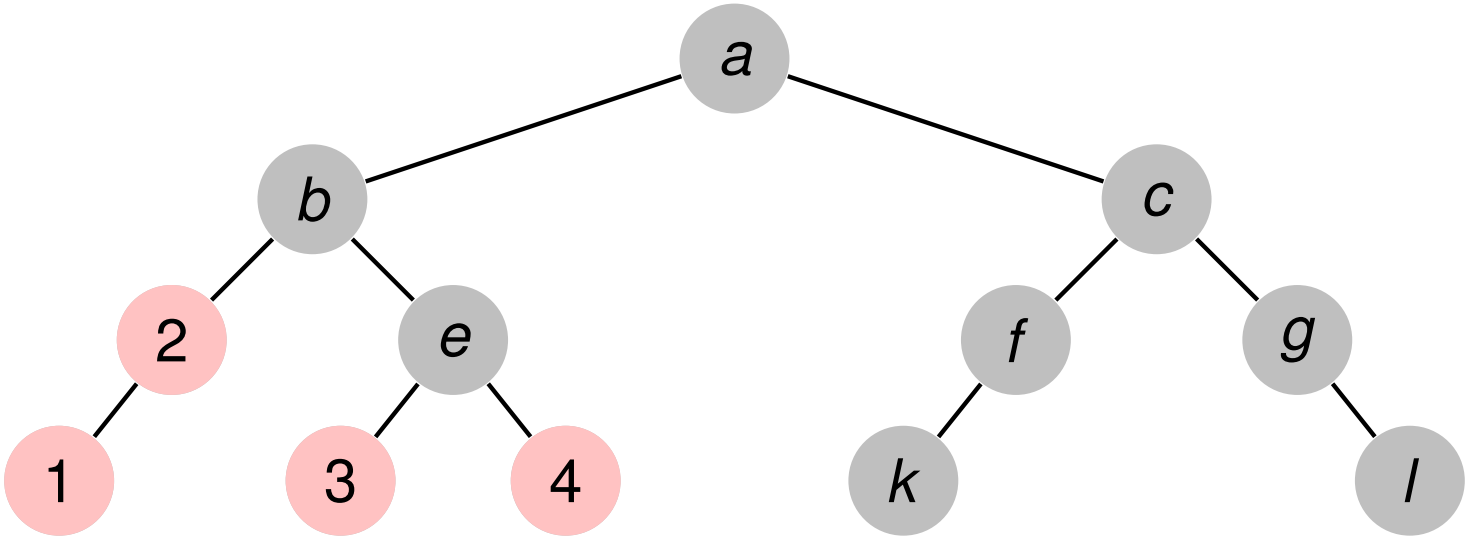
Ejemplo, recorrida pos-order



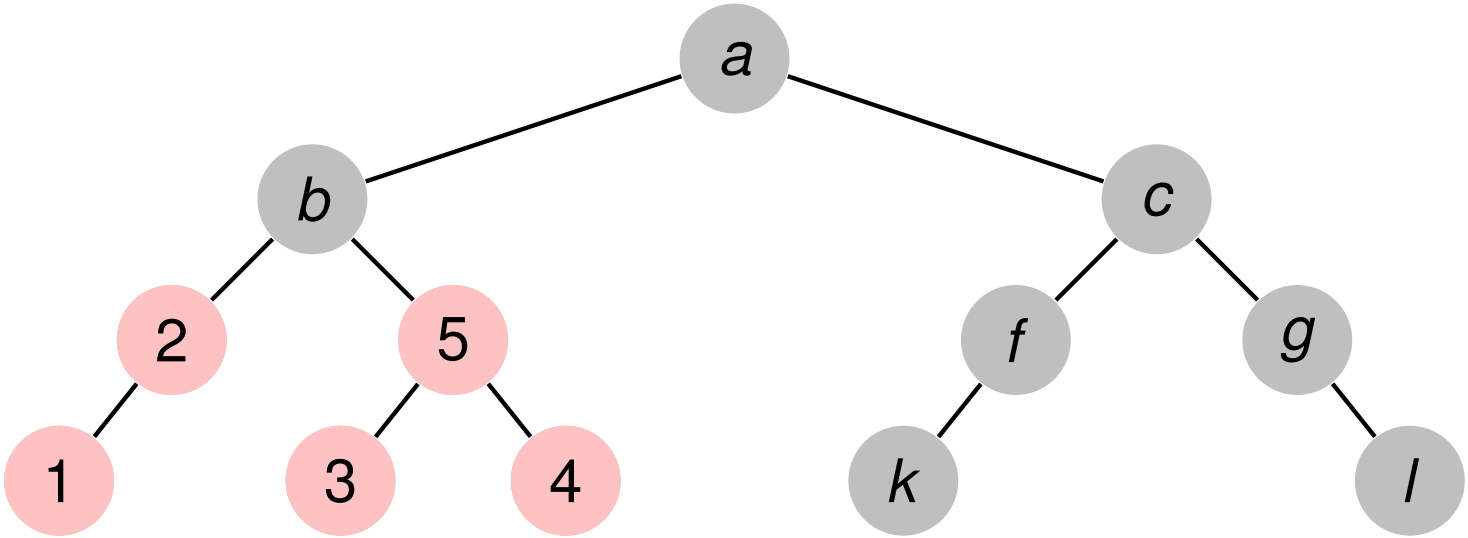
Ejemplo, recorrida pos-order



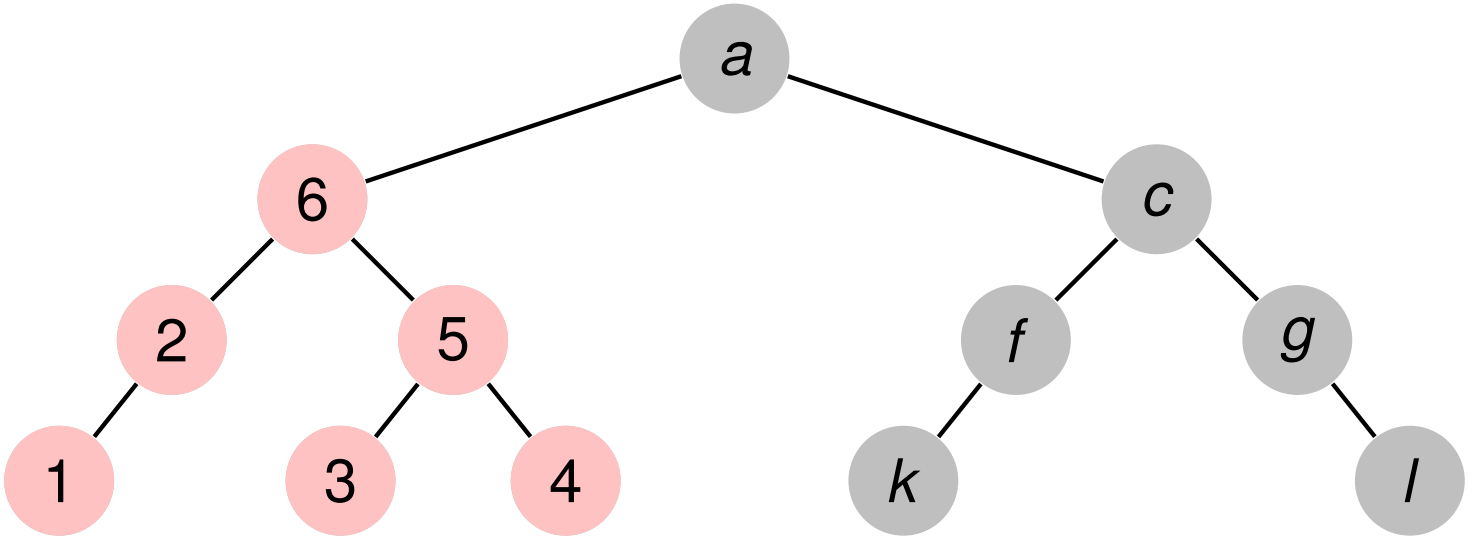
Ejemplo, recorrida pos-order



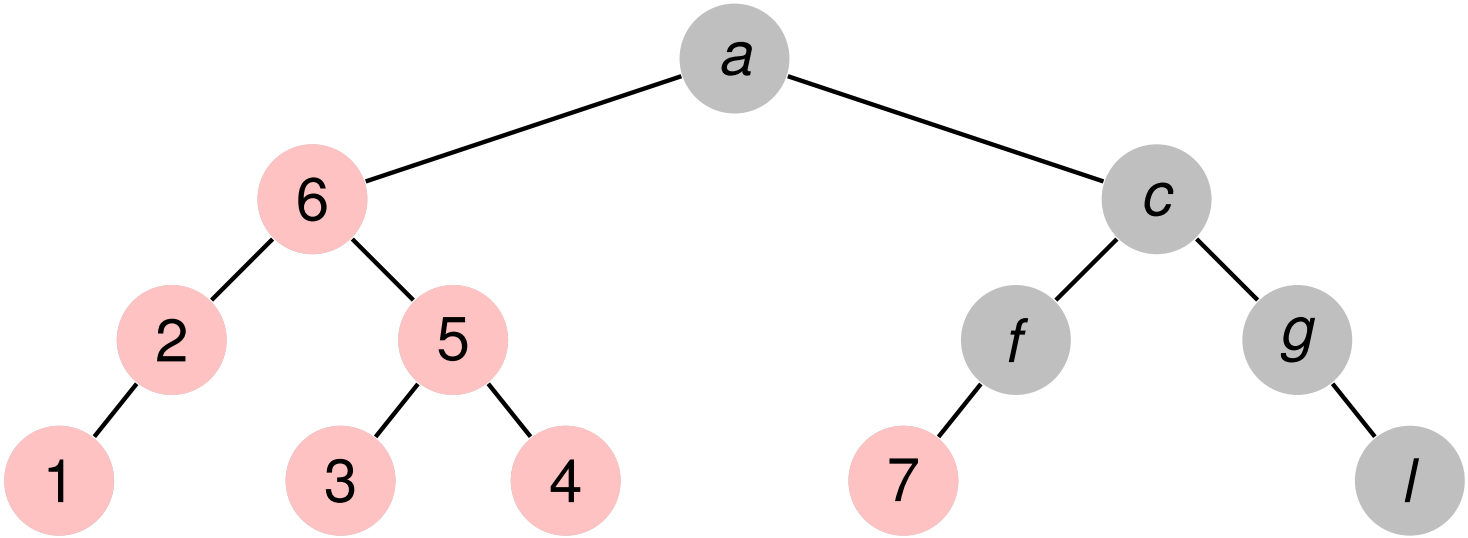
Ejemplo, recorrida pos-order



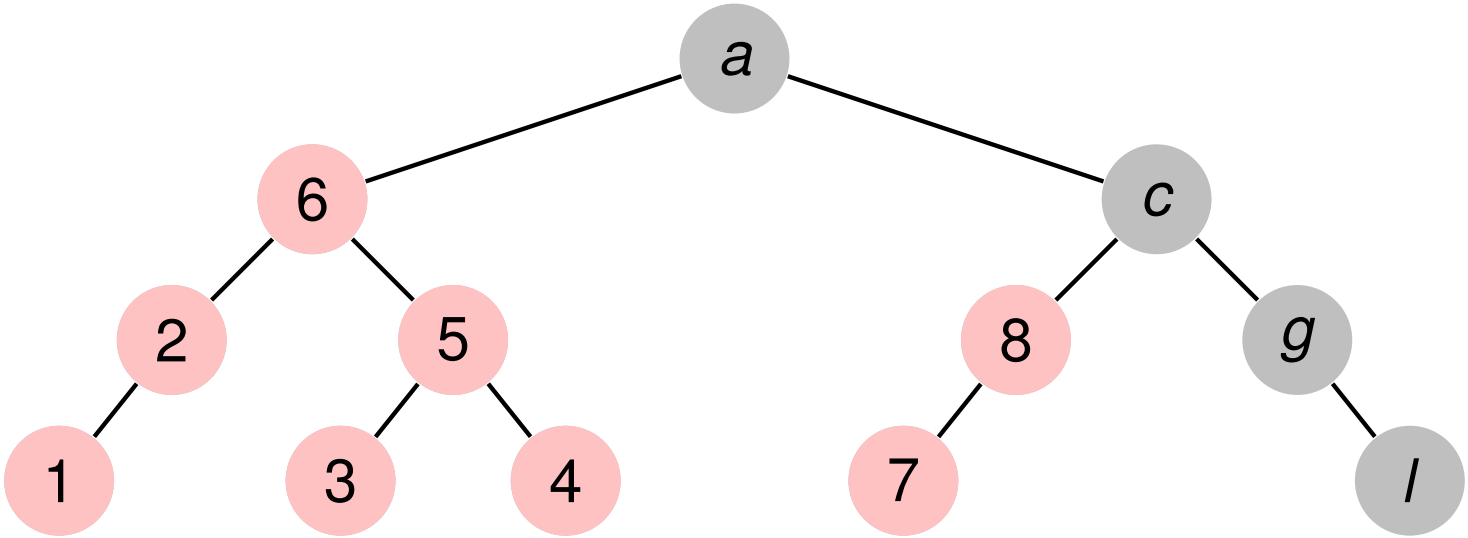
Ejemplo, recorrida pos-order



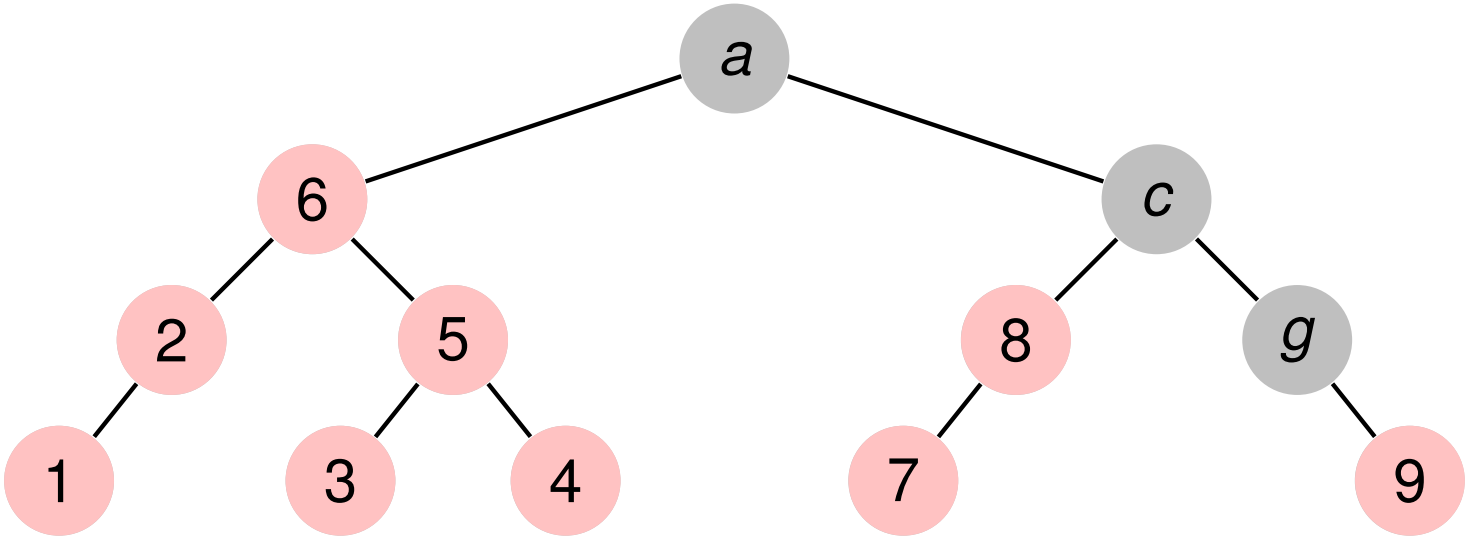
Ejemplo, recorrida pos-order



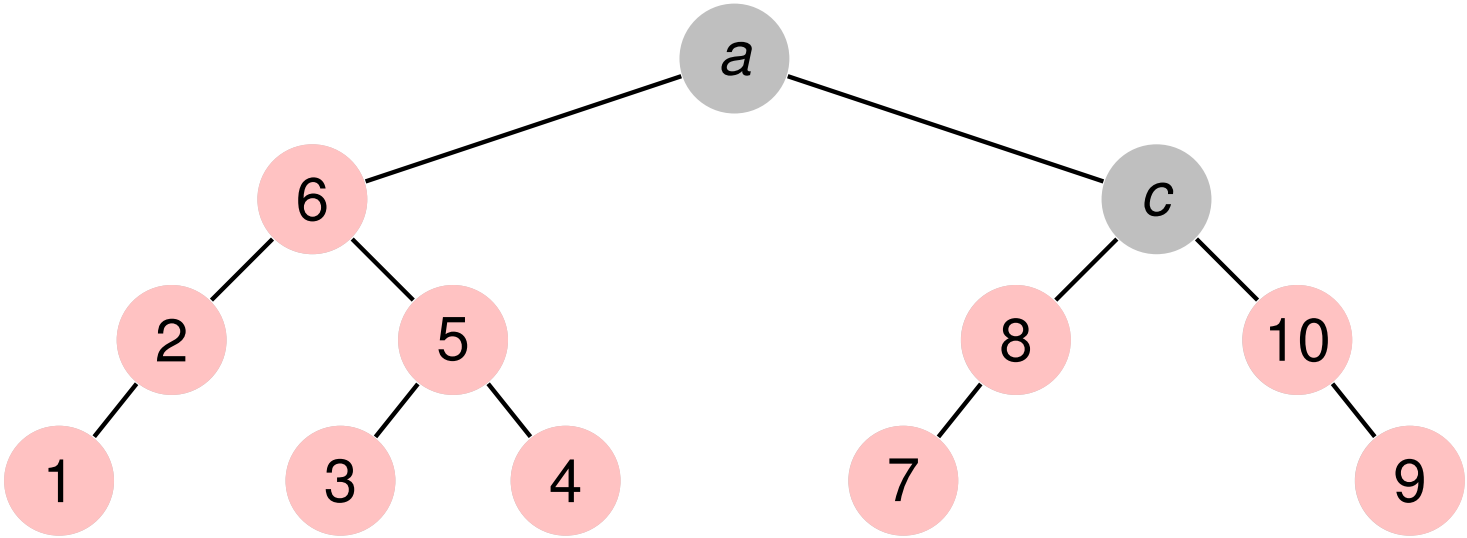
Ejemplo, recorrida pos-order



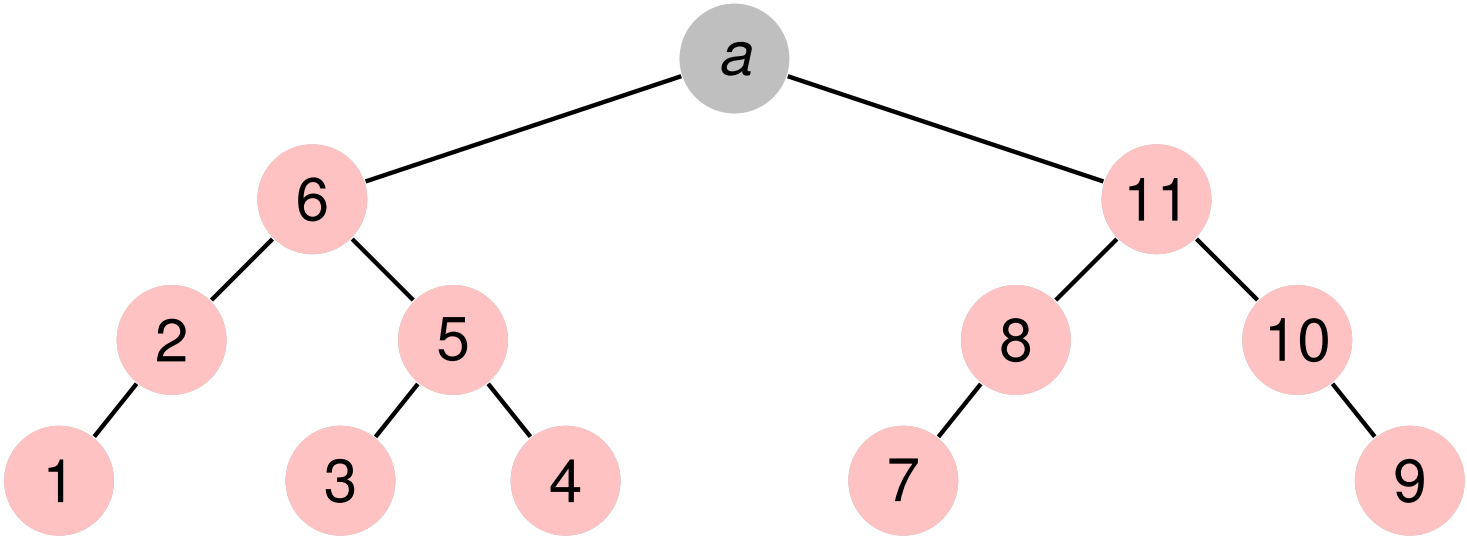
Ejemplo, recorrida pos-order



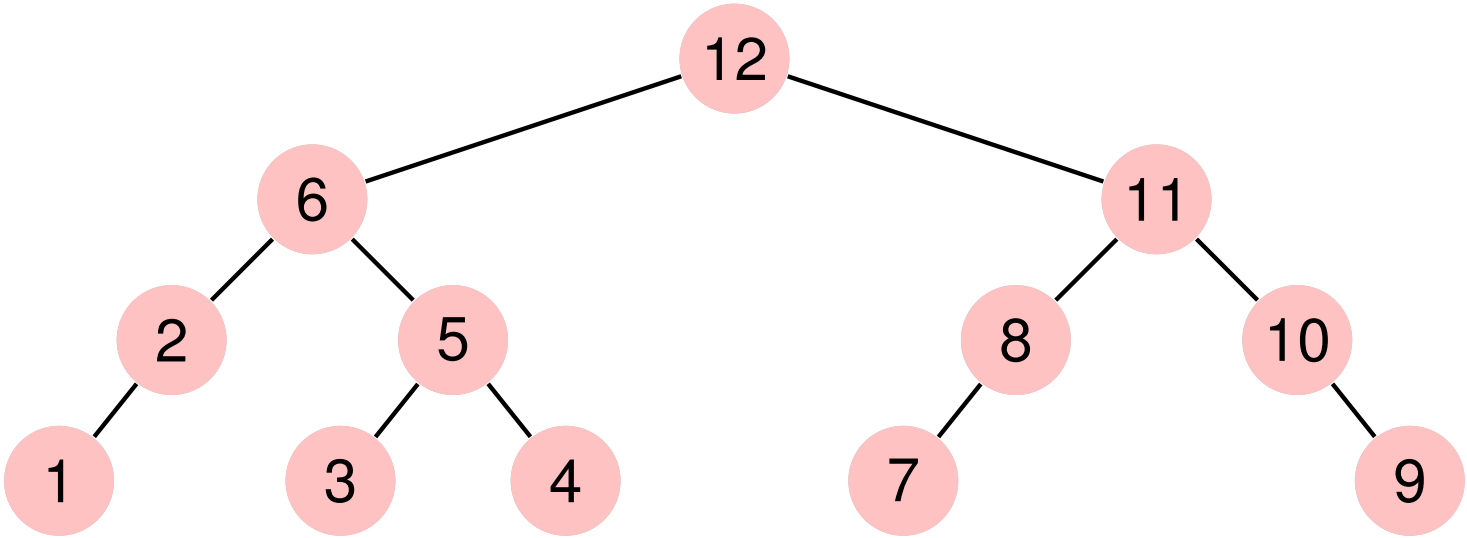
Ejemplo, recorrida pos-order



Ejemplo, recorrida pos-order



Ejemplo, recorrida pos-order

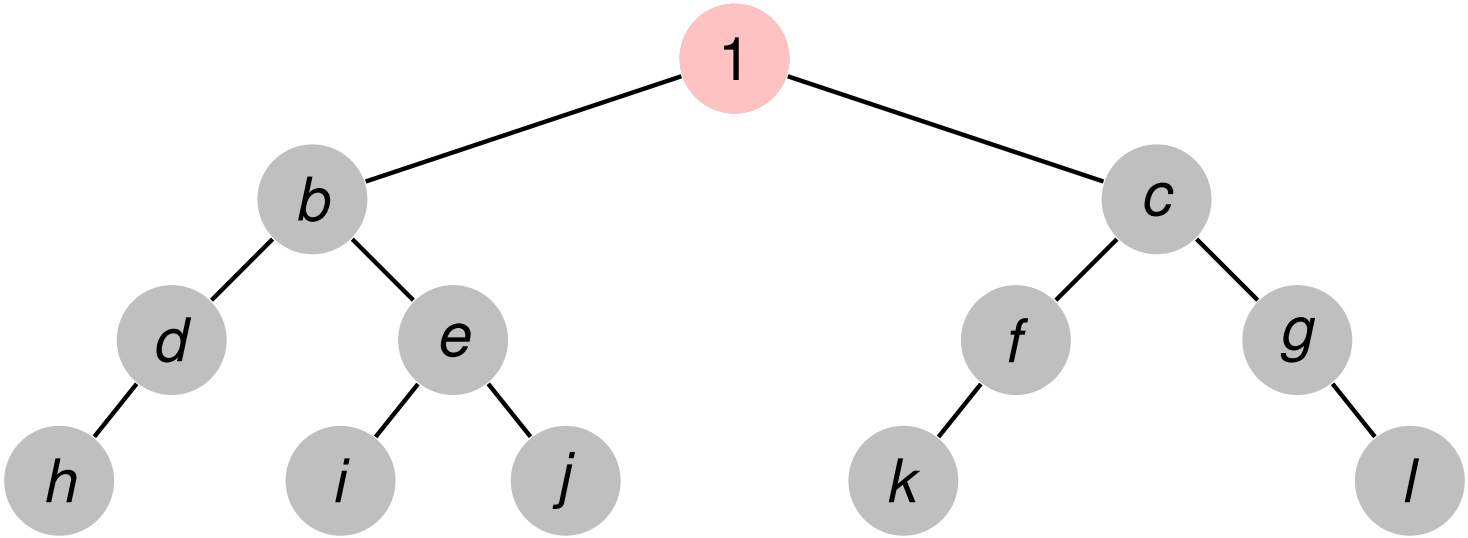


Otra manera de recorrer

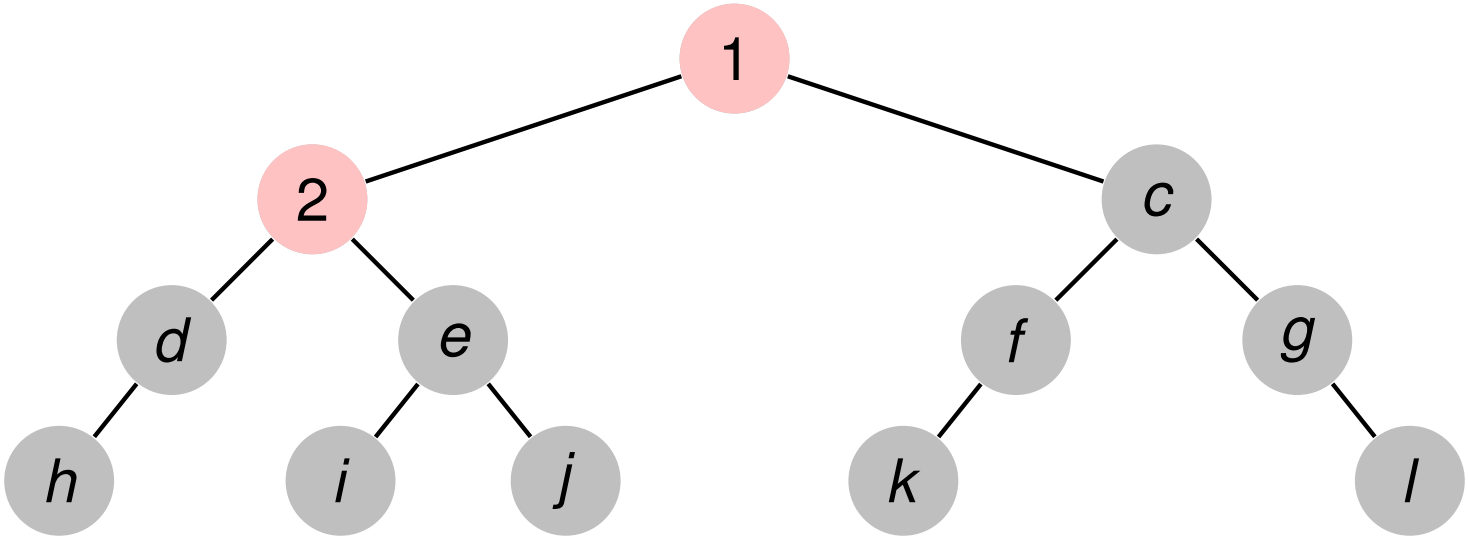
Las tres formas previas de recorrer un árbol comparten una característica: el recorrido es **en profundidad**. Es decir, el orden en que son visitados los nodos es yendo hacia las hojas o viniendo desde éstas.

La manera alternativa es recorrer por niveles: primero la raíz, luego los nodos del primer nivel, luego los del segundo, etcétera.

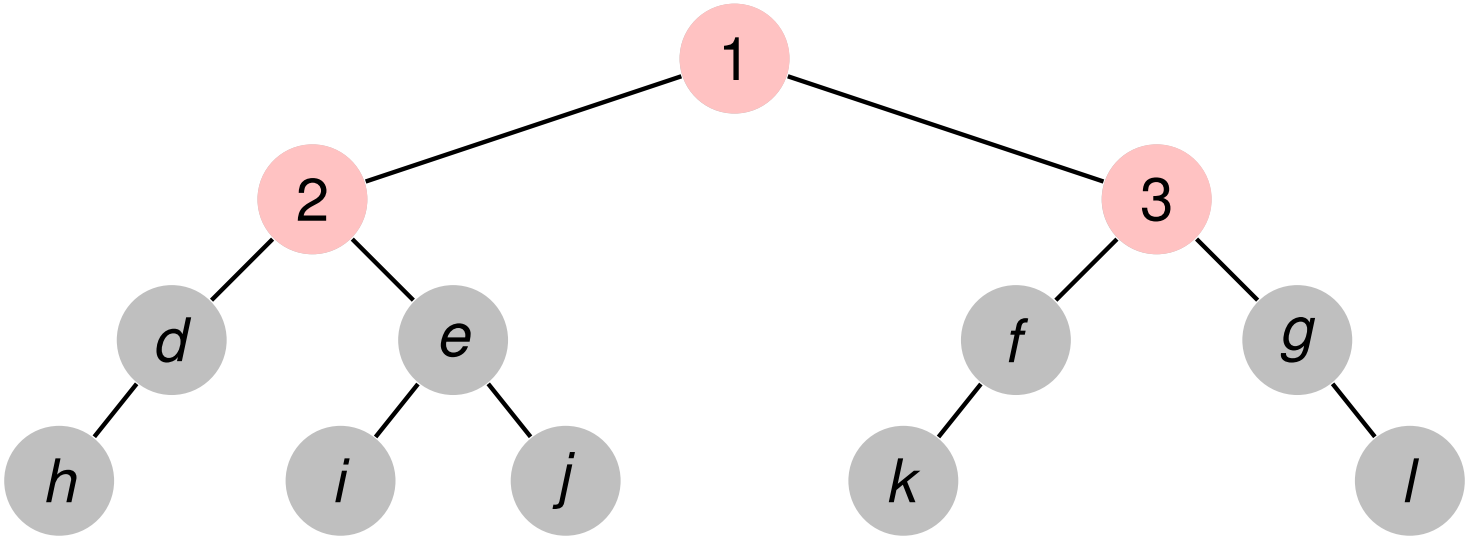
Otra manera más de recorrer árboles binarios



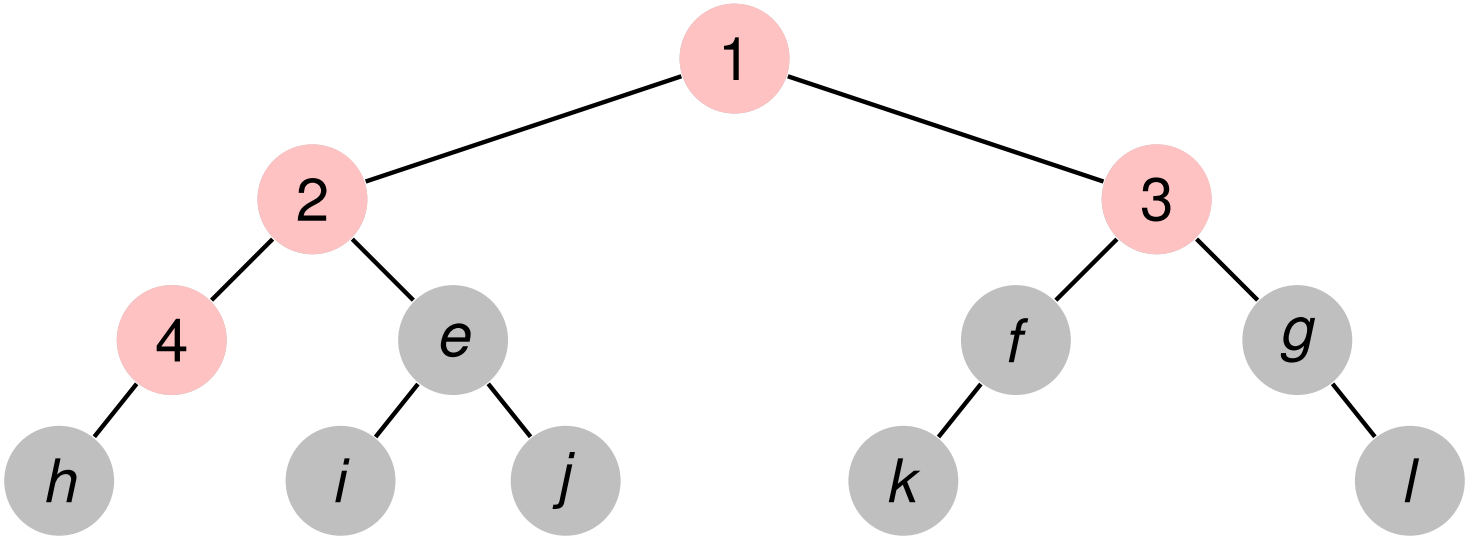
Otra manera más de recorrer árboles binarios



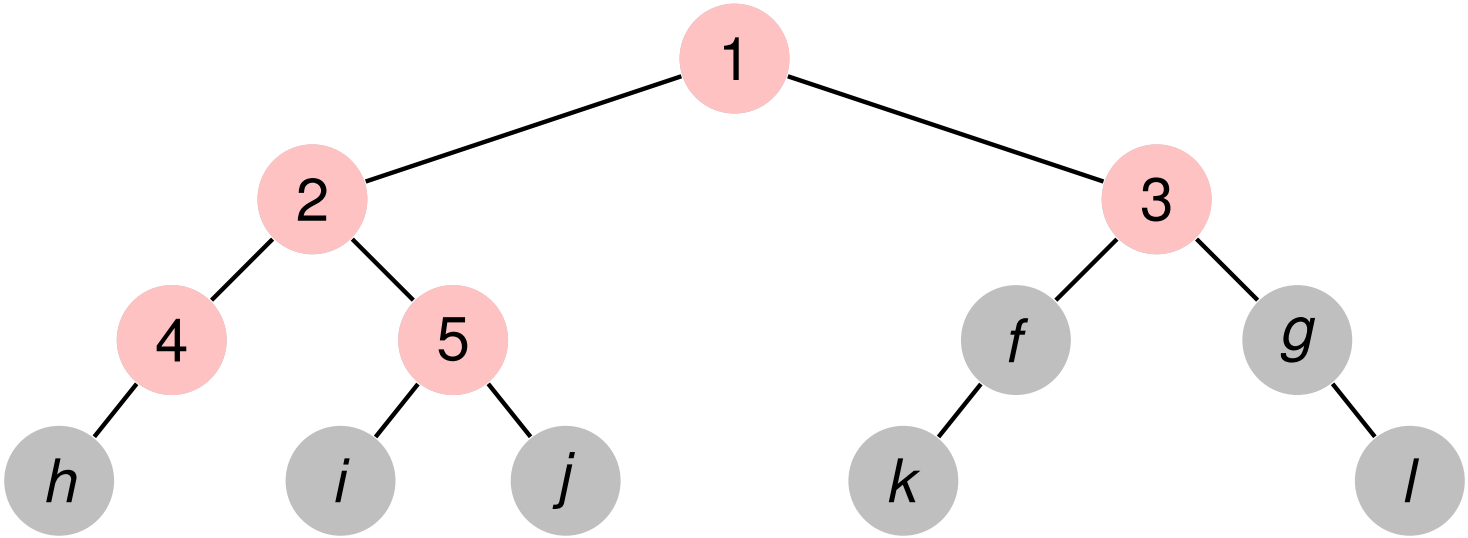
Otra manera más de recorrer árboles binarios



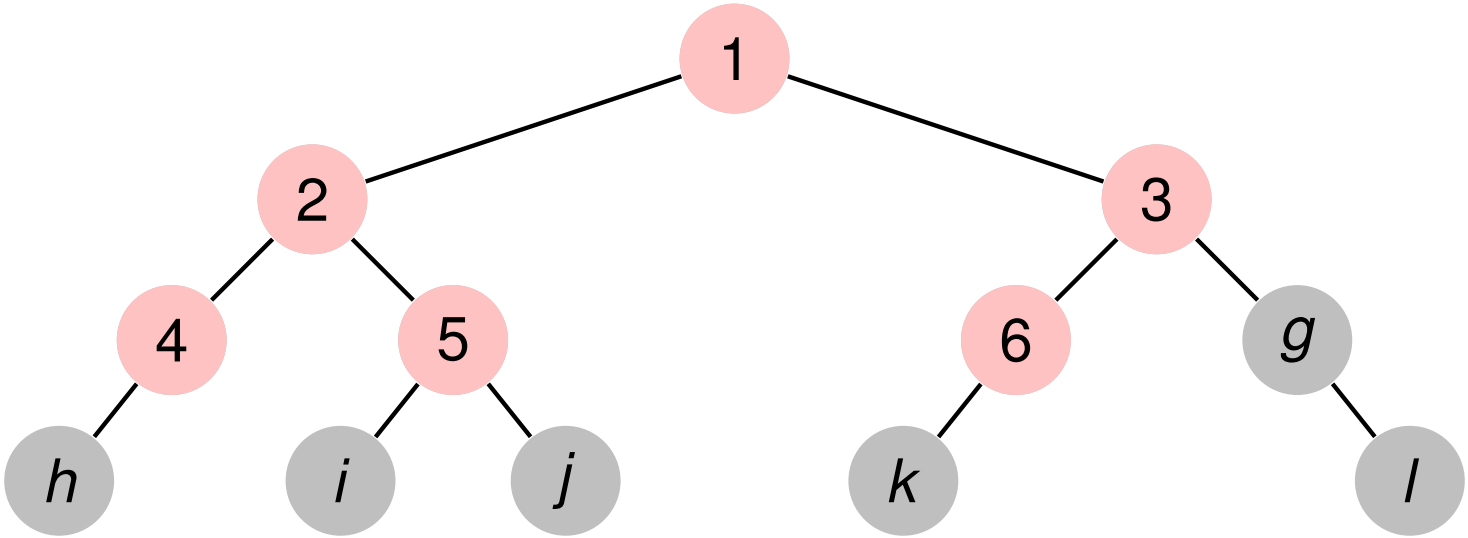
Otra manera más de recorrer árboles binarios



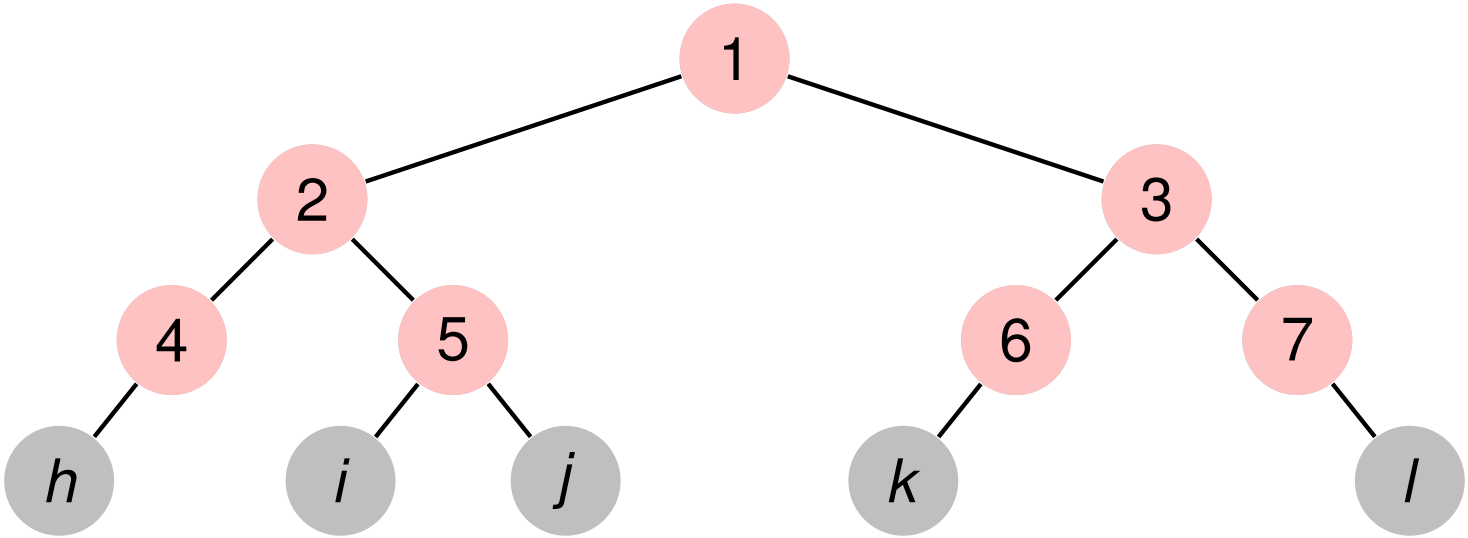
Otra manera más de recorrer árboles binarios



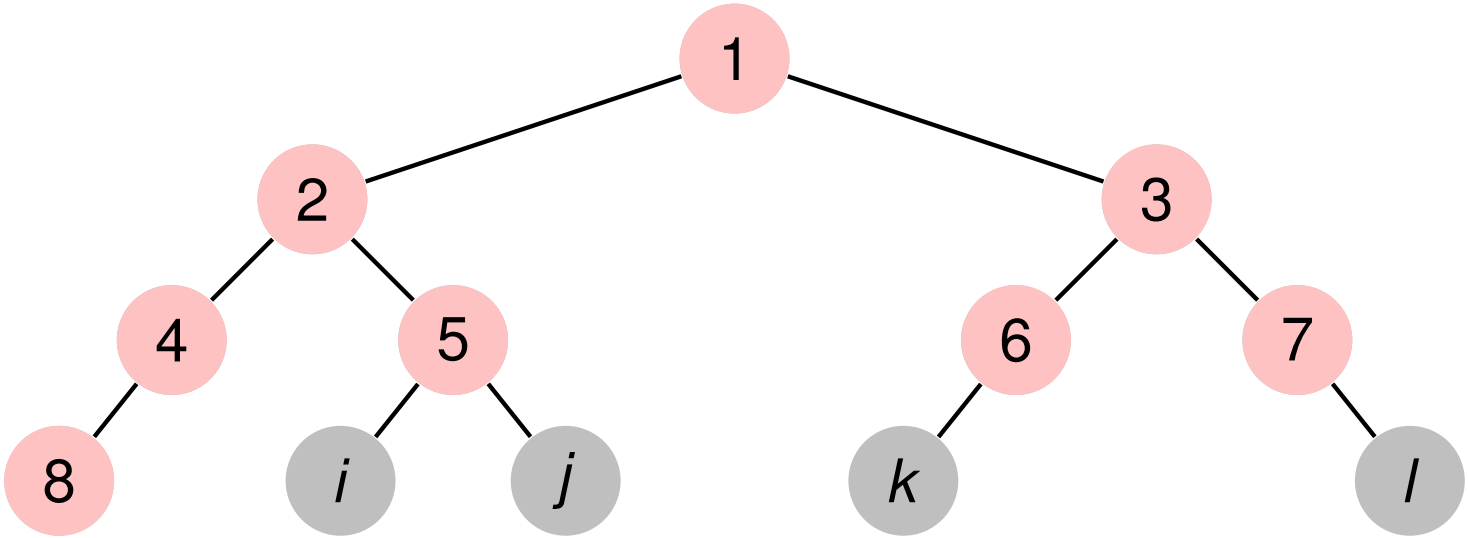
Otra manera más de recorrer árboles binarios



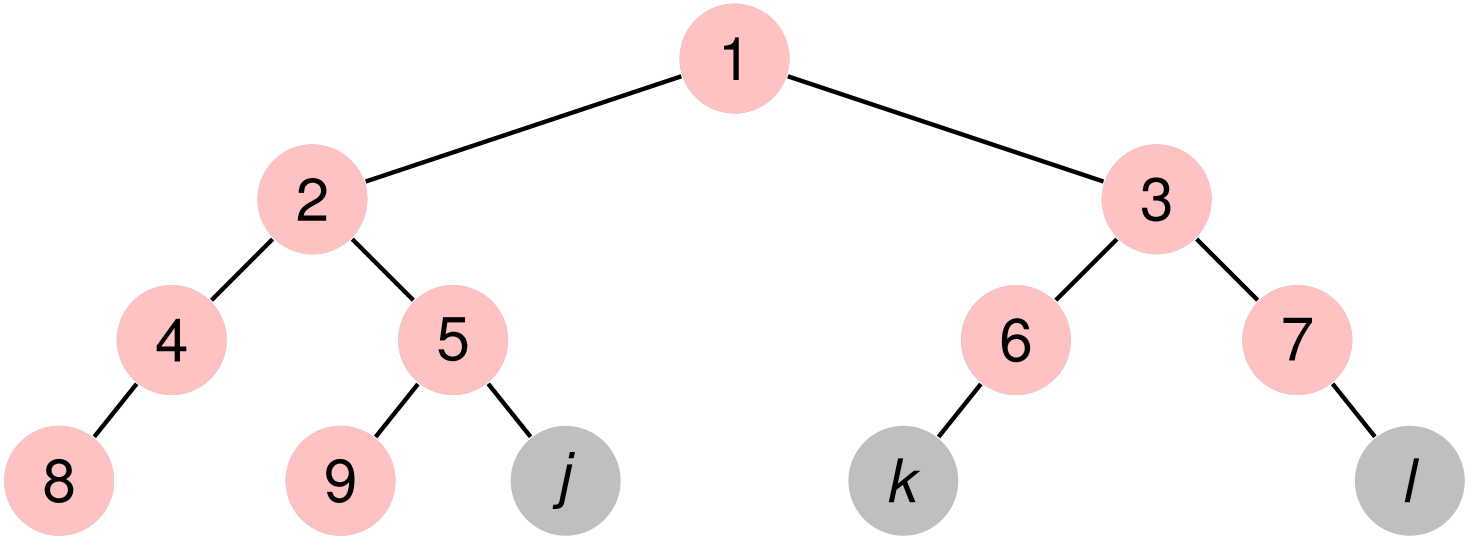
Otra manera más de recorrer árboles binarios



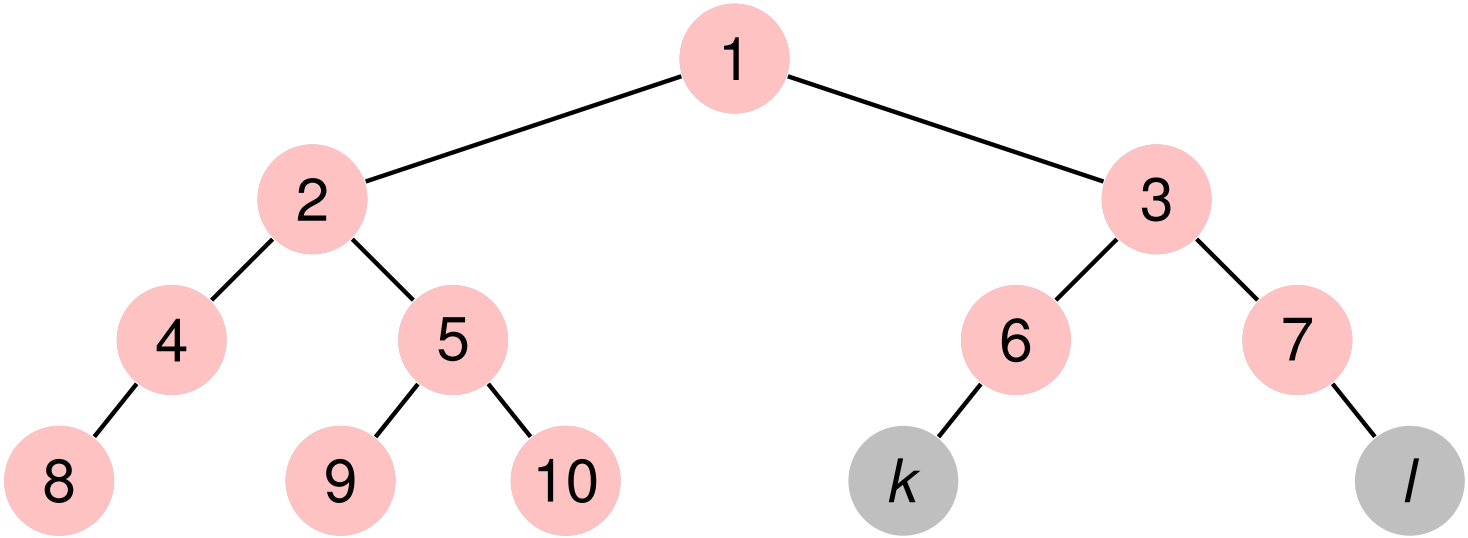
Otra manera más de recorrer árboles binarios



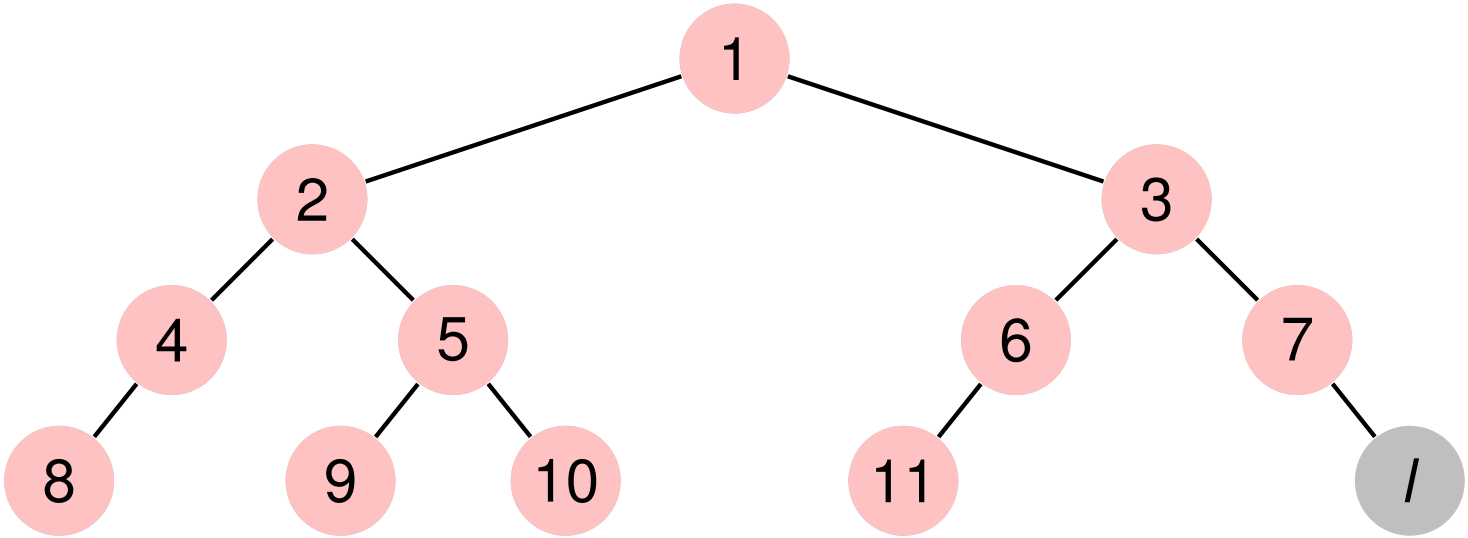
Otra manera más de recorrer árboles binarios



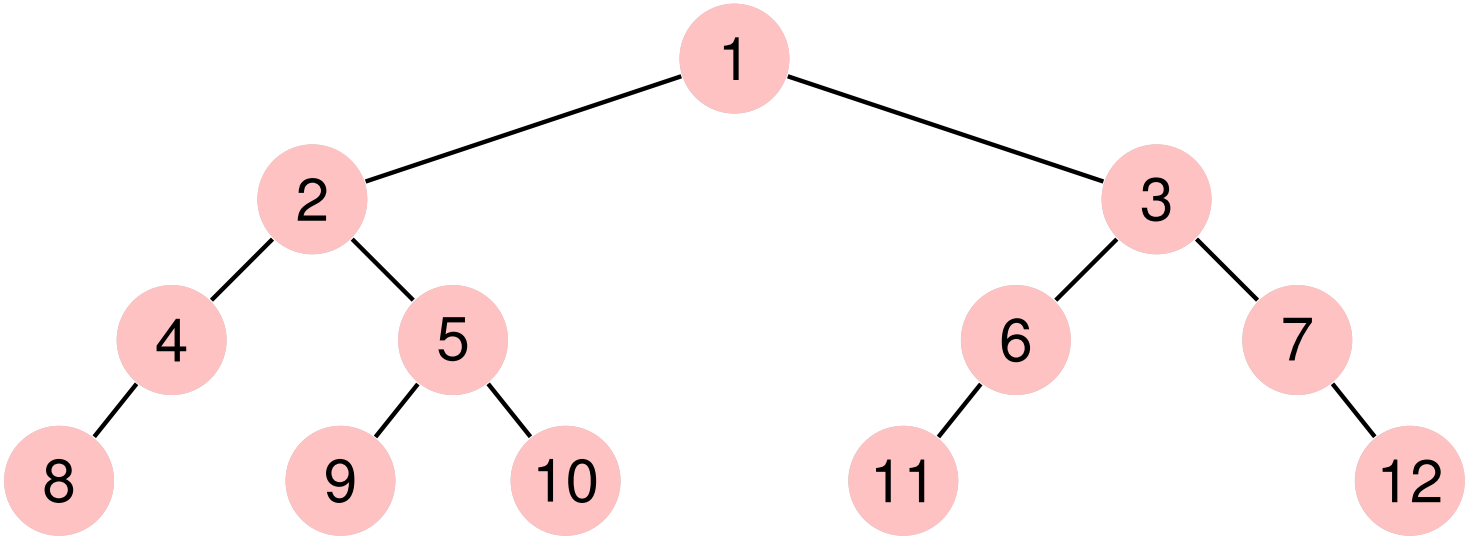
Otra manera más de recorrer árboles binarios



Otra manera más de recorrer árboles binarios



Otra manera más de recorrer árboles binarios



Recorrida de árboles

- Tanto in-order como pre-order y pos-order son recorridas *en profundidad*. Se las denomina DFS, por sus siglas en inglés Depth First Search.
- La última forma de recorrer el árbol es *a lo ancho*. Se la denomina BFS, por sus siglas en inglés Breadth First Search.
- A continuación vemos cómo implementar en el lenguaje imperativo estas recorridas.

Algoritmos
DFS recursivo

Las recorridas DFS pueden implementarse muy fácilmente utilizando recursión:

```
proc dfs(t : Tree of T)
      if  not is_empty_tree(t) then
         visit(root(t))
         dfs(left(t))
         dfs(rigth(t))
      end if
end proc
```

Las distintas variantes de DFS sobre árboles se obtienen cambiando el orden de las tres líneas de código que están dentro del if.

Algoritmos
DFS iterativo

Para implementar DFS de manera iterativa utilizamos una pila de árboles.

```
proc dfs(t : Tree of T)
    var subtrees : Stack of (Tree of T)
    var current_t : (Tree of T)

    subtrees := empth_stack()
    push(t,subtrees)
    do  (not is_empty_stack(subtrees)) →
        current_t := top(subtrees)
        pop(subtrees)
        if  (not is_empty_tree(current_t)) then
          visit(root(current_t))
          push(left(current_t) , subtrees)
          push(right(current_t) , subtrees)
        end if
    od
end proc
```

Algoritmos
BFS iterativo

Sorprendentemente la implementación de BFS es prácticamente igual a la del DFS. **Sólo debemos cambiar la pila por una cola**.

```
proc bfs(t : Tree of T)
    var subtrees : Queue of (Tree of T)
    var current_t : (Tree of T)
    subtrees := empth_queue()
    enqueue(t,subtrees)
    do  (not is_empty_queue(subtrees)) →
        current_t := first(subtrees)
        dequeue(subtrees)
        if  (not is_empty_tree(current_t)) then
          visit(root(current_t))
          enqueue(subtrees, left(current_t))
          enqueue(subtrees, right(current_t))
        end if
    od
end proc
```

Recorrida de grafos

- En el caso de grafos generales también tenemos estas dos formas de recorrer diferenciadas. Tienen una complejidad mayor, ya que hay que tener cuidado de no visitar dos veces un mismo nodo.
- Se deben adaptar las implementaciones que vimos previamente para el caso de grafos en general, y llevar un conjunto de nodos visitados que debe chequearse en cada paso. No lo veremos en detalle este año.