Universidad Abierta Interamericana

01 – PROGRAMACIÓN I

UAIOnline ultra

UAI Universidad Abierta Interamericana

UNIDAD 3

ESTRUCTURAS DINÁMICAS 2 – SEMANA 9 - ARBOLES

UAIOnline ultra

RECORRIDO DE UN ARBOL

- Se consideran dos tipos de recorrido:
  - Recorrido en profundidad
  - Recorrido en anchura o a nivel.
- Puesto que los árboles no son secuenciales como las listas, hay que buscar estrategias alternativas para visitar todos los nodos.

UAIOnline ultra

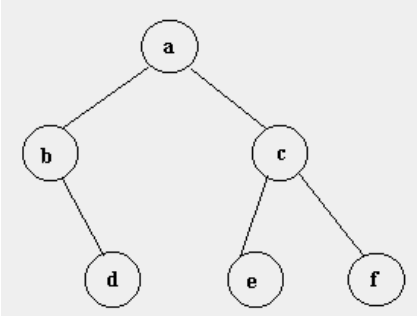
RECORRIDO EN PROFUNDIDAD

- Preorden: (raíz, izquierdo, derecho). Para recorrer un árbol binario no vacío en preorden, hay que realizar las siguientes operaciones recursivamente en cada nodo, comenzando con el nodo de raíz:
  - Visite la raíz
  - Atraviese el sub-árbol izquierdo
  - Atraviese el sub-árbol derecho
- Inorden: (izquierdo, raíz, derecho). Para recorrer un árbol binario no vacío en inorden (simétrico), hay que realizar las siguientes operaciones recursivamente en cada nodo:
  - Atraviese el sub-árbol izquierdo
  - Visite la raíz
  - Atraviese el sub-árbol derecho
- Postorden: (izquierdo, derecho, raíz). Para recorrer un árbol binario no vacío en postorden, hay que realizar las siguientes operaciones recursivamente en cada nodo:
  - Atraviese el sub-árbol izquierdo
  - Atraviese el sub-árbol derecho
  - Visite la raíz

En general, la diferencia entre preorden, inorden y postorden es cuándo se recorre la raíz

RECORRIDO EN AMPLITUD

- Consiste en ir visitando el árbol por niveles. Primero se visitan los nodos de nivel 1 (como mucho hay uno, la raíz), después los nodos de nivel 2, así hasta que ya no queden más. Si se hace el recorrido en amplitud del siguiente árbol se visitaría los nodos en el orden: a,b,c,d,e,f.
- En este caso el recorrido no se realizará de forma recursiva sino iterativa, utilizando una cola. como estructura de datos auxiliar. El procedimiento consiste en encolar (si no están vacíos) los subárboles izquierdo y derecho del nodo extraido de la cola, y seguir desencolando y encolando hasta que la cola esté vacía. (Ejemplo en C)

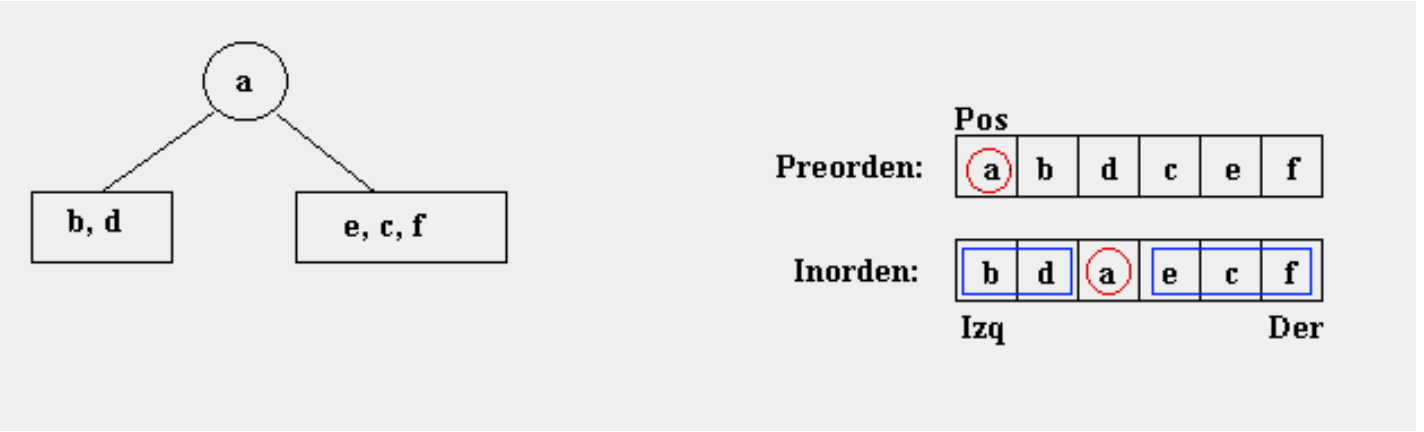



```
void amplitud(tarbol *a)
{
  tCola cola;   /* las claves de la cola serán de tipo árbol binario */
  arbol *aux;

  if (a != NULL) {
    CrearCola(cola);
    encolar(cola, a);
    while (!colavacia(cola)) {
      desencolar(cola, aux);
      visitar(aux);
      if (aux->izq != NULL) encolar(cola, aux->izq);
      if (aux->der != NULL) encolar(cola, aux->der)}}}
```

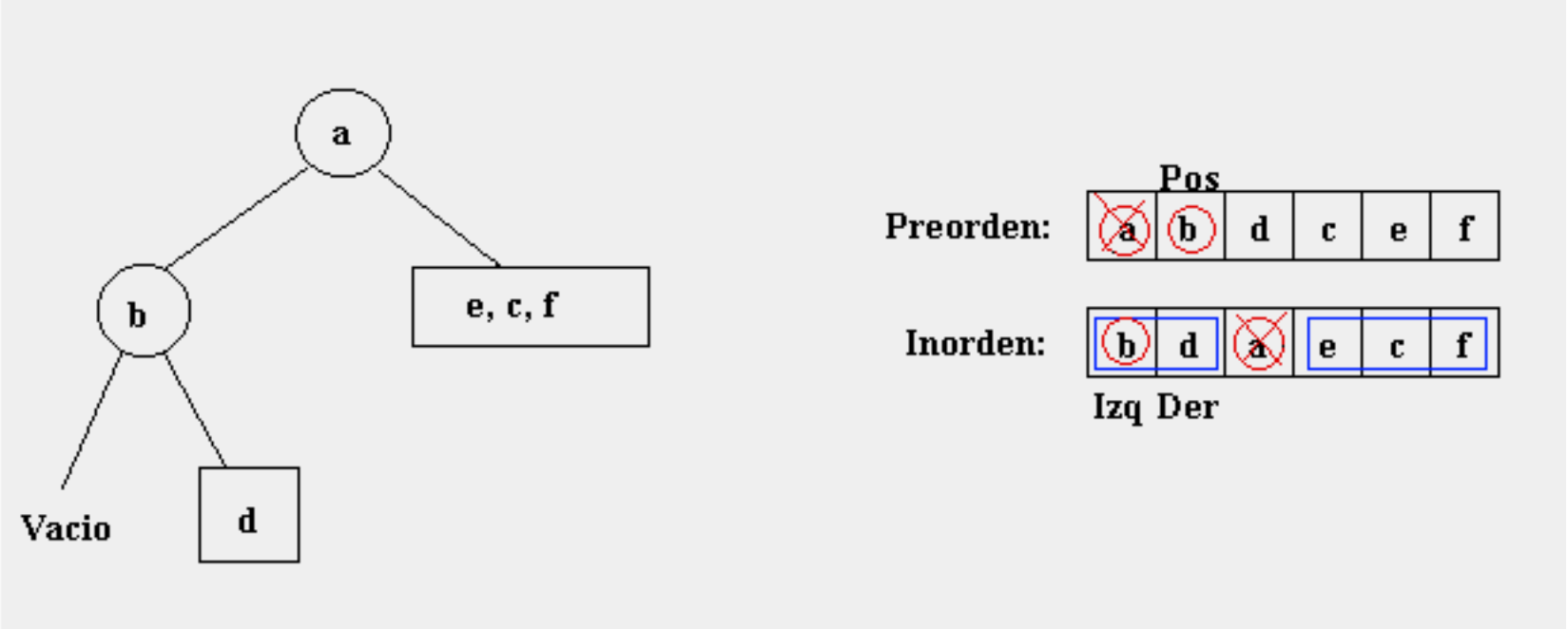
CREACIÓN DE UN ÁRBOL BINARIO

- Hasta el momento se ha visto la declaración y recorrido de un árbol binario. Sin embargo no se ha estudiado ningún método para crearlos. A continuación se estudia un método para crear un árbol binario que no tenga claves repetidas partiendo de su recorrido en preorden e inorden, almacenados en arrays.
- Partiendo de los recorridos preorden e inorden puede determinarse que la raíz es el primer elemento del recorrido en preorden. Ese elemento se busca en el array inorden. Los elementos en el array inorden entre **izq** y la raíz forman el subárbol izquierdo.
- Asimismo los elementos entre **der** y la raíz forman el subárbol derecho.



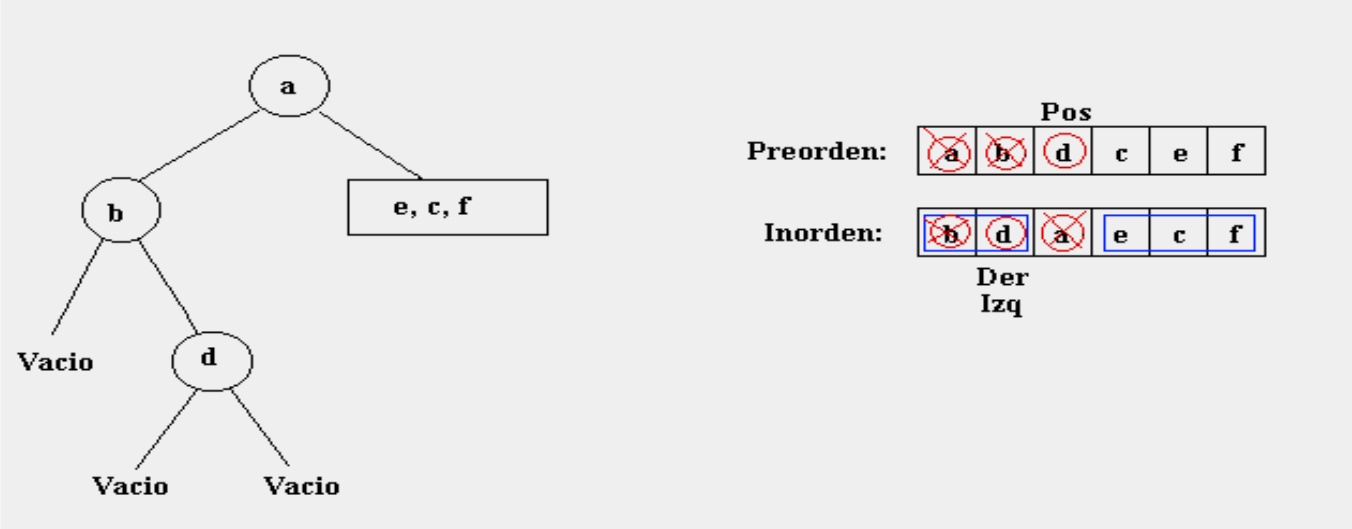
CREACIÓN DE UN ÁRBOL BINARIO

- A continuación comienza un proceso recursivo. Se procede a crear el subárbol izquierdo, cuyo tamaño está limitado por los índices **izq** y **der**.
- La siguiente posición en el recorrido en preorden es la raíz de este subárbol.



CREACIÓN DE UN ÁRBOL BINARIO

- El subárbol *b* tiene un subárbol derecho, que no tiene ningún descendiente, tal y como indican los índices **izq** y **der**.
- Se ha obtenido el subárbol izquierdo completo de la raíz *a*, puesto que *b* no tiene subárbol izquierdo.

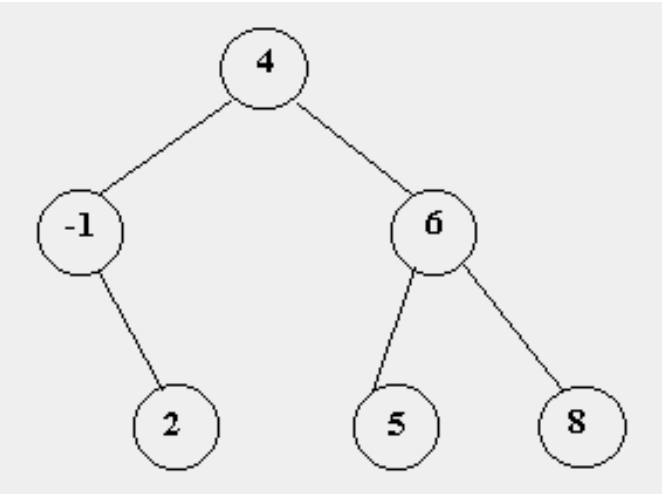


Después seguirá construyéndose el subárbol derecho a partir de la raíz *a*.

UAIOnline ultra

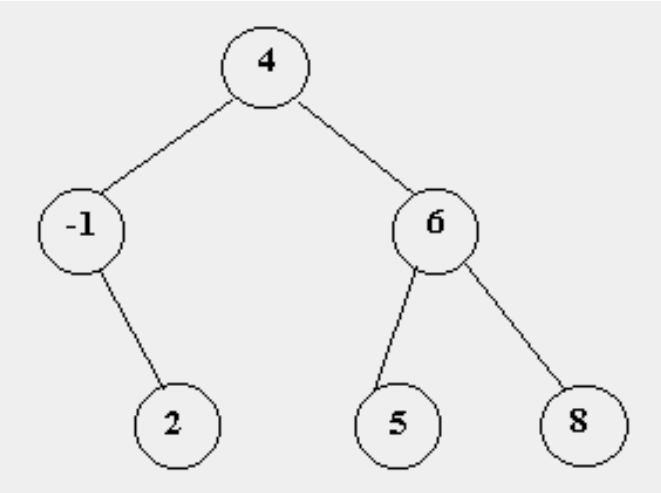
ÁRBOL BINARIO DE BÚSQUEDA

- Un árbol binario de búsqueda es aquel que es:
- Una estructura vacía o
- Un elemento o clave de información (nodo) más un número finito (a lo sumo dos) de estructuras tipo árbol, disjuntos, llamados subárboles y además cumplen lo siguiente:
    - Todas las claves del subárbol izquierdo al nodo son menores que la clave del nodo.
    - Todas las claves del subárbol derecho al nodo son mayores que la clave del nodo.
    - Ambos subárboles son árboles binarios de búsqueda.



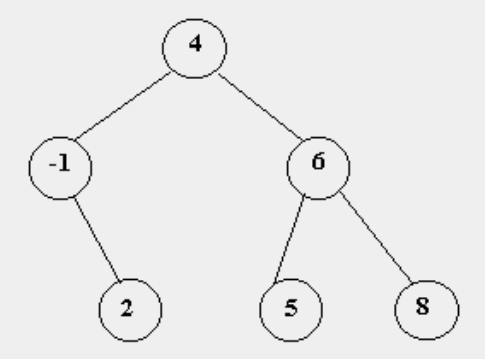
ÁRBOL BINARIO DE BÚSQUEDA

- En el ejemplo las claves son números enteros. Dada la raíz *4*, las claves del subárbol izquierdo son menores que 4, y las claves del subárbol derecho son mayores que 4. Esto se cumple también para todos los subárboles. Si se hace el recorrido de este árbol en orden central se obtiene una lista de los números ordenada de menor a mayor. [SEP]
- Cuestión: ¿Qué hay que hacer para obtener una lista de los números ordenada de mayor a menor?



ÁRBOL BINARIO DE BÚSQUEDA

- Cuestión: ¿Qué hay que hacer para obtener una lista de los números ordenada de mayor a menor?
- Una ventaja fundamental de los árboles de búsqueda es que son en general mucho más rápidos para localizar un elemento que una lista enlazada.
- Por tanto, son más rápidos para insertar y borrar elementos.
- Si el árbol está **perfectamente equilibrado** entonces el número de comparaciones necesarias para localizar una clave es aproximadamente de logN en el peor caso.



ÁRBOL BINARIO DE BÚSQUEDA

- Cuestión: ¿Qué hay que hacer para obtener una lista de los números ordenada de mayor a menor?
- Una ventaja fundamental de los árboles de búsqueda es que son en general mucho más rápidos para localizar un elemento que una lista enlazada.
- Por tanto, son más rápidos para insertar y borrar elementos.
- Si el árbol está **perfectamente equilibrado** entonces el número de comparaciones necesarias para localizar una clave es aproximadamente de logN en el peor caso.
- Aprovechando las propiedades del árbol de búsqueda se puede acelerar la localización. Simplemente hay que descender a lo largo del árbol a izquierda o derecha dependiendo del elemento que se busca.