Video19 (1)

Conceptos Principales

1. a
2. b
3. c

Notas

Introducción a Árboles

Concepto general

- Un **árbol** es una **estructura de datos no lineal**.
- Se usa principalmente para **búsquedas rápidas** y para representar jerarquías.
- En computación se representa **invertido**:
  - **Raíz** arriba.
  - **Hojas** abajo.

Elementos principales

- **Nodo raíz**: nodo principal, desde el que se parte.
- **Nodos internos**: los que tienen hijos.
- **Hojas**: nodos sin hijos.
- **Aristas**: conexiones entre nodos.
- **Subárbol**: cualquier nodo con sus descendientes.
  - Se habla de **subárbol izquierdo** y **subárbol derecho**.

Representación en memoria

Estructura típica de un nodo de árbol binario:

```
struct nodoArbol {
    int valor;                  // puede almacenar más información
    struct nodoArbol *izq;      // puntero a hijo izquierdo
    struct nodoArbol *der;      // puntero a hijo derecho
};
```

> Similar a una lista doble, pero en lugar de “anterior/siguiente” se usa “izquierdo/derecho”.

Tipos de árboles

- **Árbol binario**: cada nodo tiene como máximo 2 hijos (`izq` y `der`).
- **Árbol n-ario**: cada nodo puede tener hasta *n* hijos.
- **Árbol binario de búsqueda (ABB)**:
  - Los valores **menores** van al lado izquierdo.
  - Los valores **mayores** van al lado derecho.
  - Muy usado porque imita la búsqueda binaria.

Propiedades

- **Altura**: distancia desde la raíz hasta la hoja más profunda.
- **Nivel**: profundidad de un nodo específico (cuánto baja desde la raíz).
- **Grado**: número máximo de hijos que puede tener un nodo (en un binario, 2).
- **Relaciones jerárquicas**:
  - Padre ↔ Hijo.
  - Hermanos (nodos con el mismo padre).

Recorridos (traversals)

Los árboles no se "imprimen" en forma gráfica, sino en una versión **lineal** usando recorridos:

1. **Preorden**:
   - Visitar raíz → Subárbol izquierdo → Subárbol derecho.
2. **Inorden**:
   - Subárbol izquierdo → Raíz → Subárbol derecho.
3. **Postorden**:
   - Subárbol izquierdo → Subárbol derecho → Raíz.

Ejemplo en C (inorden):

```c
void inorden(struct nodoArbol *raiz) {
    if (raiz == NULL) return;
    inorden(raiz->izq);
    printf("%d ", raiz->valor);
    inorden(raiz->der);
}
```

Nota: la mayoría de algoritmos en árboles se hacen con recursividad.

Operaciones básicas

- **Insertar**: colocar nodo siguiendo reglas del árbol (ej: en ABB, menor izquierda, mayor derecha).
- **Buscar**: se baja recursivamente hasta encontrar el valor.
- **Eliminar**: más complejo (distintos casos: nodo hoja, con un hijo o con dos hijos).

Idea central:

Un **árbol binario de búsqueda** es como una “búsqueda binaria en estructura enlazada”, donde cada nodo tiene un subárbol izquierdo y derecho. Se trabaja principalmente con **recursividad** para recorridos, búsqueda e inserción.