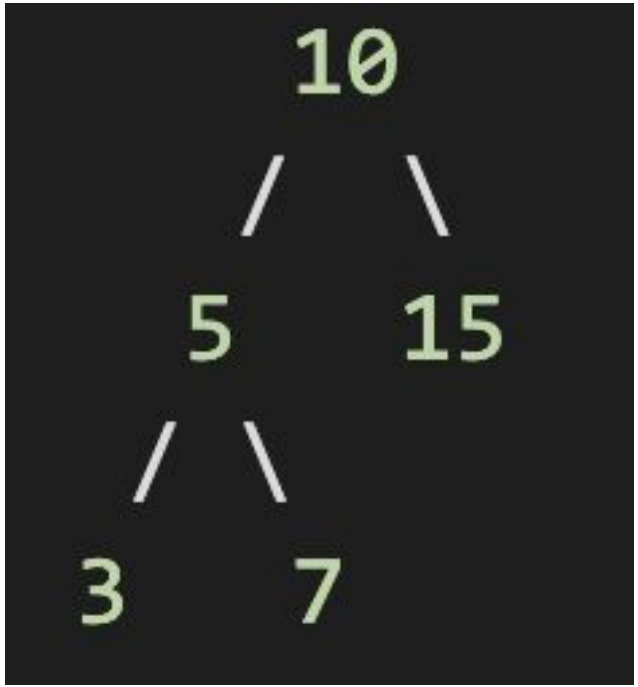
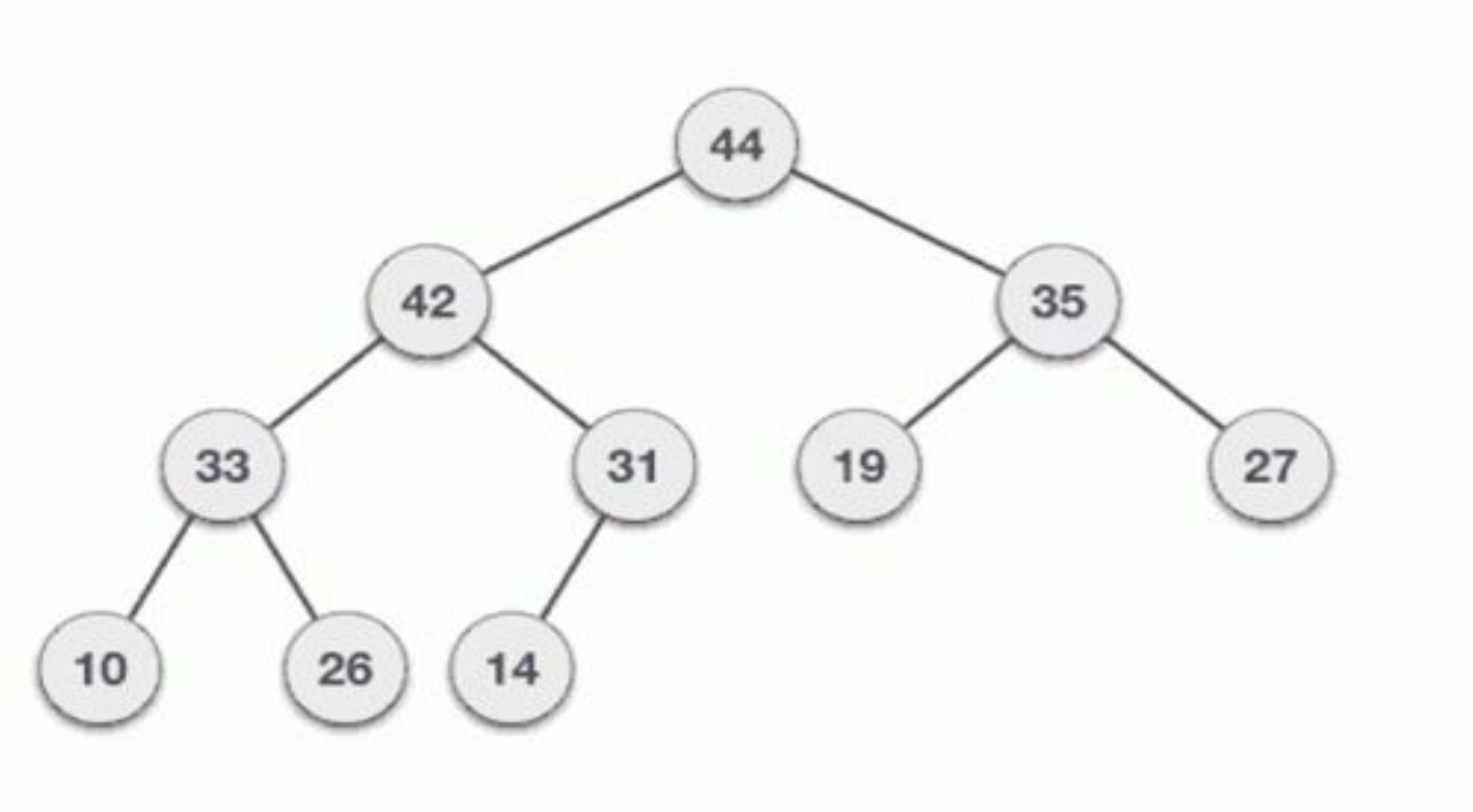
Unidad 4
Árboles

- El nodo **10** es la **raíz** (inicio del árbol).
- Tiene dos hijos: 5 (izquierdo) y 15 (derecho).
- El 5 tiene a su vez dos hijos: 3 y 7.
- Los nodos sin hijos se llaman **hojas**.



¿Para qué sirven los árboles binarios?

- Para ordenar datos rápidamente.
- Para búsquedas eficientes (como en Google).
- Para representar jerarquías o decisiones.


44
42
35
33
31
19
27
10
26
14

Input 35 33 42 10 14 19 27 44 26 31

```
// Estructura del nodo
struct Nodo {
    int dato;
    struct Nodo* izquierdo;
    struct Nodo* derecho;
};
```

```c
// Crear nuevo nodo
struct Nodo* nuevoNodo(int valor) {
    struct Nodo* nodo = (struct Nodo*)malloc(sizeof(struct Nodo));
    nodo->dato = valor;
    nodo->izquierdo = NULL;
    nodo->derecho = NULL;
    return nodo;
}
```

```c
// Insertar un valor en el árbol
struct Nodo* insertar(struct Nodo* raiz, int valor) {
    if (raiz == NULL) {
        return nuevoNodo(valor);
    }

    if (valor < raiz->dato) {
        raiz->izquierdo = insertar(raiz->izquierdo, valor);
    } else {
        raiz->derecho = insertar(raiz->derecho, valor);
    }

    return raiz;
}
```

```c
// Recorrido inorden
void inorden(struct Nodo* raiz) {
    if (raiz != NULL) {
        inorden(raiz->izquierdo);
        printf("%d ", raiz->dato);
        inorden(raiz->derecho);
    }
}
```

```c
struct Nodo* raiz = NULL;

// Insertamos algunos valores
raiz = insertar(raiz, 10);
insertar(raiz, 5);
insertar(raiz, 15);
insertar(raiz, 3);
insertar(raiz, 7);

printf("Recorrido inorden del árbol: ");
inorden(raiz);  // Debería imprimir: 3 5 7 10 15
```