Binary Tree

1. Inserción

Mejor caso: O(logn), si el árbol está balanceado.
Peor caso: O(n), si el árbol está degenerado (todos los nodos están en un solo lado).

2. Búsqueda

- **Descripción**:
  - Comienza en la raíz y, dependiendo del valor buscado:
    - Se mueve al subárbol izquierdo si el valor buscado es menor que el nodo actual.
    - Se mueve al subárbol derecho si el valor buscado es mayor.
  - Finaliza si encuentra el nodo o llega a un nodo nulo.
- **Complejidad**:
  - **Mejor caso**: O(logn), si el árbol está balanceado.
  - **Peor caso**: O(n), si el árbol está degenerado.

3. Eliminación

- **Descripción**:
  - Localiza el nodo a eliminar.
  - Hay tres casos posibles:
    1. **El nodo es una hoja**: Se elimina directamente.
    2. **El nodo tiene un hijo**: Se reemplaza con su único hijo.
    3. **El nodo tiene dos hijos**:
       - Encuentra el sucesor (mínimo en el subárbol derecho) o el predecesor (máximo en el subárbol izquierdo).
       - Intercambia valores y elimina el sucesor/predecesor.
- **Complejidad**:
  - **Mejor caso**: O(logn), si el árbol está balanceado.
  - **Peor caso**: O(n), si el árbol está degenerado.