Analizar la complejidad temporal de las operaciones: Evaluar el costo de tiempo de las operaciones básicas (inserción, búsqueda, eliminación) en un BST.

Inserción:
La complejidad temporal en el peor de los casos de las operaciones de inserción es O(h), donde h es la altura del árbol de búsqueda binaria.
En el peor de los casos, es posible que tengamos que viajar desde la raíz hasta el nodo de la hoja más profundo. La altura de un árbol sesgado puede llegar a ser n y la complejidad temporal de la operación de inserción puede llegar a ser O(n).

Búsqueda:
En el mejor caso, el valor buscado está en la raíz, por lo que solo se hace una comparación. Esto tiene una complejidad de O(1).
En el peor caso, el árbol está completamente desbalanceado, es decir, todos los nodos están alineados en una sola rama (como una lista enlazada). En este caso, la búsqueda recorrería todos los nodos, lo que tiene una complejidad de O(n), donde n es el número total de nodos.
En el caso de un árbol balanceado, la altura del árbol es aproximadamente O(logn), por lo que la búsqueda tendría una complejidad deO(logn).

Eliminación:

Buscar el nodo a eliminar tiene la misma complejidad que la búsqueda, que es O(h), donde h es la altura del árbol.

Eliminar el nodo implica realizar algunas operaciones adicionales, como encontrar el sucesor en el subárbol derecho (lo que también toma O(h) si es necesario. Sin embargo, la complejidad sigue dominada por la búsqueda del nodo.

https://www.geeksforgeeks.org/binary-search-tree-data-structure/

```
manuel@manuel-IdeaPadL7:~/Datastructures/binarytree$ ./tree
Inorden: 20 30 40 50 60 70 80
Preorden: 50 30 20 40 70 60 80
Postorden: 20 40 30 60 80 70 50
Valor encontrado: 40
Eliminando 20...
Inorden después de eliminar 20: 30 40 50 60 70 80
manuel@manuel-IdeaPadL7:~/Datastructures/binarytree$
```