**Reflexiones sobre el TAD BST**Al ejecutar el programa p4\_e1.c se puede comprobar que los tiempos de creación y búsqueda varían mucho según si el programa se ejecuta en modo normal o en modo ordenado.

**¿Por qué es así?**   
La diferencia en los tiempos de creación y búsqueda entre el modo normal y el modo ordenado se debe a cómo se inserta la información en el árbol. En el modo ordenado, los elementos se insertan de forma tal que el árbol se mantiene equilibrado, lo que resulta en una menor profundidad, esto hace que las operaciones de búsqueda y creación sean más eficientes, ya que el tiempo de ejecución está relacionado con la profundidad del árbol (por ejemplo, si un árbol tiene una profundidad de 16 para 50k elementos, las operaciones de búsqueda solo necesitan recorrer unos pocos niveles). Por el contrario, en el modo normal, la inserción no está optimizada, lo que puede dar lugar a árboles más desbalanceados y, por lo tanto, con mayor profundidad, lo que aumenta los tiempos de búsqueda.

**¿Hay alguna propiedad del árbol que permita explicar este comportamiento?**  
Sí, hay una propiedad de los árboles que puede ayudar a explicar este comportamiento, el factor de equilibrio de un árbol. Este factor se refiere a la diferencia en altura entre sus subárboles izquierdo y derecho. Un árbol se considera equilibrado si el factor de equilibrio de cada nodo en el árbol es -1, 0 o 1. Cuando un árbol está equilibrado, su profundidad se minimiza, lo que puede hacer que la búsqueda de elementos sea más rápida porque hay menos niveles para recorrer.

Cuando un árbol se construye en modo ordenado, los elementos se insertan de manera que equilibra el árbol, lo que resulta en una profundidad de árbol más baja. En contraste, cuando el árbol se construye en modo normal, los elementos pueden no insertarse de una manera que equilibre el árbol, lo que resulta en una profundidad de árbol más profunda y tiempos de búsqueda potencialmente más lentos.

**Reflexiones sobre el TAD SQ**  
 **¿Qué diferencias y similitudes hay entre el TAD SQ y el TAD Cola de la práctica anterior?**

Las SQ tienen la ventaja de estar organizadas gracias a su estructura de árbol, a diferencia de las colas que no siguen ningún orden específico. Esto hace que funciones típicas de las colas, como push\_front o push\_back no sean relevantes, ya que los elementos se insertan de forma ordenada, además, no es necesario tener punteros front o rear en las SQ, ya que se puede acceder a estos mediante funciones como find\_min o find\_max. En cuanto a la operación pop, aunque extrae el primer elemento, no siempre es el primero insertado, ya que los elementos están organizados de manera jerárquica. Además, las SQ no pueden implementarse de forma circular, a diferencia de las colas.

**¿Qué coste (aproximado) tiene la operación de extraer un elemento en el TAD SQ? ¿Sería posible hacer que esta operación fuera 𝒪(1)?**

El coste de extraer un elemento dependerá de la profundidad del árbol, dado que es necesario recorrer el subárbol izquierdo para encontrar el mínimo y luego realizar el proceso de eliminación, por lo tanto, el coste será aproximadamente 𝒪(2n). Si el árbol contiene un solo elemento, el coste sería 𝒪(1), ya que solo se debe acceder a ese nodo para identificarlo como mínimo. En un caso general, no es posible obtener una operación de 𝒪(1), pero se podría optimizar utilizando un doble puntero al mínimo, lo que permitiría eliminar ese elemento en 𝒪(1), aunque luego se tendría que buscar un nuevo mínimo, lo que llevaría el coste a 𝒪(n).