b) Dê a ordem do pior caso e do caso esperado de tempo de execução para cada método.

c) Qual é a eficiência de utilização de memória (relação entre o espaço necessário para dados e o espaço total necessário) para cada método?

2. Suponha uma lista ordenada contendo n itens e um item x que **não** está presente na lista. O problema consiste em determinar entre qual par de itens na lista está o item x, isto é, encontrar $a[i]$ e $a[i+1]$ de tal forma que $a[i] < x < a[i+1]$, para $1 \leq i < n$, ou que $x < a[1]$ ou que $x > a[n]$.

a) Encontre o limite inferior para essa classe de problemas quanto ao número de comparações.

b) Apresente uma prova informal para o limite inferior.

c) Você conhece algum algoritmo que seja ótimo para resolver o problema?

3. Qual é a principal propriedade de uma árvore binária de pesquisa?

4. Árvore Binária de Pesquisa.

a) Desenhe a árvore binária de pesquisa que resulta da inserção sucessiva das chaves Q U E S T A O F C I L em uma árvore inicialmente vazia.

b) Desenhe as árvores resultantes das retiradas dos elementos E e depois U da árvore obtida no item anterior.

5. Implemente uma função que conta quantos elementos existem em uma árvore binária de pesquisa (Guedes Neto, 2010).

6. O **caminhamento por nível** em árvores visita primeiro a raiz, depois todos os nós no nível 1, depois todos os nós no nível 2, e assim por diante (Loureiro, 2010). Escreva um algoritmo $O(n)$ para listar os n nós de uma árvore por nível, do nó mais a esquerda para o mais a direita em cada nível. Sugestão: não usar recursividade.

7. Suponha que você tenha uma árvore binária de pesquisa na qual estão armazenadas uma chave em cada nó. Suponha também que a árvore foi construída de tal maneira que, ao caminhar nela na ordem central, as chaves são visitadas em *ordem crescente*.

a) Qual propriedade entre as chaves deve ser satisfeita para que isso seja possível?

b) Dada uma chave k, descreva sucintamente um algoritmo que procure por k em uma árvore com essa estrutura.

c) Qual é a complexidade do seu algoritmo no melhor e no pior casos? Justifique.

8. Considere o algoritmo para pesquisar e inserir registros em uma **árvore binária de pesquisa sem balanceamento**. Em razão de sua simplicidade e eficiência, a árvore binária de pesquisa é considerada uma estrutura de dados muito útil. Considerando-se que a altura da árvore corresponde ao tamanho da pilha necessária para pesquisar na árvore, é importante conhecer o seu valor. Assim sendo,

a) determine empiricamente a altura esperada da árvore;

b) mostre analiticamente o melhor caso e o pior caso para a altura da árvore;

c) compare os resultados obtidos no item (a) com resultados analíticos publicados na literatura.

9. Para pesquisar um elemento em um arranjo ordenado de tamanho n usando **pesquisa binária**, o elemento é comparado com o elemento na posição $\lfloor n/2 \rfloor$ do arranjo. Para pesquisar um elemento no mesmo arranjo usando pesquisa ternária, o elemento é comparado com os elementos nas posições $\lfloor n/3 \rfloor$ e $\lfloor 2n/3 \rfloor$.

a) Determine o número de comparações necessárias para encontrar um elemento usando pesquisa binária e pesquisa ternária.

b) Qual dos dois métodos é preferível: pesquisa binária ou pesquisa ternária?

c) Qual é o limite inferior para o problema de realizar busca em um arranjo ordenado?

10. Árvore SBB.

a) Desenhe a **árvore SBB** que resulta da inserção sucessiva das chaves Q U E S T A O F C I L em uma árvore inicialmente vazia.

b) Desenhe as árvores resultantes das retiradas dos elementos E e depois U da árvore obtida no item anterior.

11. Árvore SBB.

Um novo conjunto de transformações para a **árvore SBB** foi proposto por Olivié (1980). O algoritmo de inserção usando as novas transformações produz árvores SBB com menor altura e demanda um número menor de transformações de divisão de nós para construir a árvore, conforme comprovado em Ziviani e Tompa (1982) e Ziviani, Olivié e Gonnet (1985). A Figura 5.21 mostra as novas transformações. A operação divide esquerda-esquerda requer modificação de três apontadores, a operação divide esquerda-direita requer a alteração de cinco apontadores, e a operação aumenta altura requer apenas a modificação de dois *bits*. Transformações simétricas também podem ocorrer.

Quando ocorre uma transformação do tipo aumenta altura, a altura da subárvore transformada é um nível maior do que a da subárvore original, o que pode provocar outras transformações ao longo do caminho de pesquisa até a raiz da árvore. Geralmente, o retorno ao longo do caminho de pesquisa termina quando um apontador vertical é encontrado ou uma transformação do tipo divide é realizada. Como a altura da subárvore que sofreu a divisão é a mesma que a altura da subárvore original, apenas uma transformação do tipo divide é suficiente para restaurar a propriedade SBB da árvore.