Lista de Exercícios — Árvores Binárias de Busca

QXD0110 – Estrutura de Dados – 2025.2

Prof. Atílio Gomes

4 de janeiro de 2026

Aluno: [ ] Matrícula: [ ]

1. Escreva uma função que decida se uma dada árvore binária é ou não de busca.

2. Suponha que `(X->esq->chave)` $<$ `X->chave` $<$ `(X->dir->chave)` para cada nó `X` de uma árvore binária. Podemos dizer que toda chave na subárvore esquerda de `X` é menor que a chave de `X`? Podemos dizer que toda chave na subárvore direita de `X` é maior que a chave de `X`? Justifique sua resposta.

3. Escreva uma função que transforme um vetor ordenado em ordem crescente em uma árvore binária de busca balanceada.

4. Escreva uma função que transforme uma árvore de busca em um vetor crescente.

5. Suponha que os nós com chaves 50, 30, 70, 20, 40, 60, 80, 15, 25, 35, 45, 36 são inseridos, nesta ordem, numa árvore de busca inicialmente vazia. Desenhe a árvore que resulta. Em seguida, remova o nó que tem chave 30 de modo que a árvore continue sendo de busca.

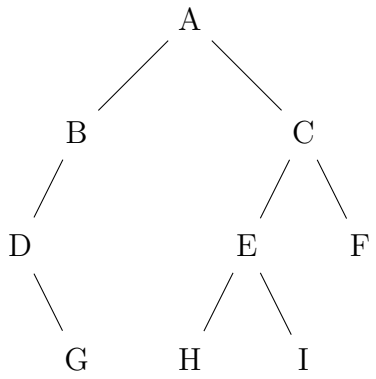
6. Mostre que: se um nó em uma árvore binária de busca tem dois filhos, então seu sucessor não tem filho esquerdo e seu antecessor não tem filho direito.

7. Considere uma árvore binária de busca T (lembre que as chaves são todas distintas). Mostre que: se a subárvore direita de um nó x em T é vazia e x tem um sucessor y, então é o ancestral mais próximo de x cujo filho esquerdo é também um ancestral de x (tenha em mente que todo nó é um ancestral dele mesmo).

8. [Percurso em ordem simétrica] Escreva um algoritmo **iterativo** que percorre todos os nós de uma árvore binária de busca em ordem simétrica (inorder traversal) e imprime na tela as chaves dos nós na ordem em que eles são visitados por esta busca.

9. [Percurso em pré-ordem] Escreva um algoritmo **iterativo** que percorre todos os nós de uma árvore binária de busca em pré-ordem (preorder traversal) e imprime na tela as chaves dos nós na ordem em que eles são visitados por esta busca.

10. [Percurso em Largura] Existem outras formas de percorrer uma árvore binária além do percurso em pré-ordem, in-ordem e pós-ordem. Por exemplo, o **percurso em nível (em largura)** é aquele em que os nós são dispostos em ordem não decrescente de seus níveis. Esse percurso é único quando se define a ordem em que os nós do mesmo nível são visitados, por exemplo, da esquerda para a direita. O percurso em nível, segundo esse critério, para a árvore da figura abaixo fornece a sequência `ABCDEFGHI`.



O percurso em nível difere, em essência, dos outros três percursos citados acima. Enquanto nesses últimos o percurso da árvore pode ser decomposto em percursos (contíguos) de suas subárvores, o mesmo não acontece com o percurso em nível. Por esse motivo, o percurso em nível é de caráter não-recursivo, isto é, um algoritmo para obter um percurso em nível não deve ser recursivo. De forma equivalente, o algoritmo não deve usar a pilha como estrutura de dados auxiliar.

11. Nós podemos ordenar um dado conjunto de n números do seguinte modo: primeiro construímos uma árvore binária de busca contendo estes números (usando a função `bst_insert` para inserir os números um por um). Depois, imprimimos os números usando o percurso em ordem simétrica (inordem). Quais são as complexidades de melhor caso e de pior caso para este algoritmo de ordenação?

12. A operação de remoção em uma árvore binária de busca é **comutativa** no sentido de que remover x e depois y resulta na mesma árvore que resultaria se primeiro removêssemos y e depois x? Prove sua resposta ou dê um contraexemplo.