Lista de Exercícios 1 — Árvores Binárias

QXD0115 – Estrutura de Dados Avançada – Turma 01A – 2023.1

Prof. Atílio Gomes

9 de março de 2023

Aluno: [ ] Matrícula: [ ]

1. Escreva uma função que calcule o número de nós de uma árvore binária.
2. Escreva uma função que conte o número de folhas de uma árvore binária.
3. Escreva uma função que exclua todas as folhas de uma árvore binária.
4. Escreva uma função que conte o número de nós internos de uma árvore binária.
5. Escreva uma função que encontre o nó com maior número em uma árvore binária.
6. Escreva uma função que encontre um nó com chave k em uma árvore binária. Sua função deve retornar um ponteiro para o nó, caso ele exista, ou `nullptr`, caso contrário.
7. Escreva uma função recursiva que apaga todas as folhas de uma árvore que tenham a chave igual a um valor dado.
8. Escreva algoritmos recursivos e não-recursivos para determinar:
   (a) O número de nós com valores pares em uma árvore binária.
   (b) A soma dos conteúdos de todos os nós em uma árvore binária, considerando que cada nó contém um inteiro.
   (c) A profundidade de uma árvore binária.
9. Mostrar que toda árvore pode ser representada por uma sequência binária. Quantos 0's e 1's possui essa sequência?
10. Representar, através de uma árvore, a seguinte expressão aritmética:

$$[(a+b)(c+d)/e] - [(f+g)h]$$

11. Provar ou dar contraexemplo: Se v é o pai de um nó w de uma árvore T, então:
    (i) $\text{nivel}(v) = \text{nivel}(w)+1$
    (ii) $\text{altura}(v) = \text{altura}(w)+1$
    (iii) $\max_{v \in T}\{\text{altura}(v)\} = \max_{v \in T}\{\text{nivel}(v)\}$

12. Mostrar que toda árvore com $n > 1$ nós possui no mínimo 1 e no máximo $n - 1$ folhas.

13. Uma árvore m-ária T, $m > 2$, é um conjunto finito de elementos, denominados nós, tais que:

    - $T = \emptyset$ e a árvore é dita vazia, ou
    - $T \neq \emptyset$ e T contém um nó especial r, chamado raiz de T, e os restantes podem ser sempre divididos em m subconjuntos disjuntos, as i-ésimas subárvores de r, $1 \leq i \leq m$, as quais são também árvores m-árias.

    Mostrar que o número de subárvores vazias de uma árvore m-ária com $n > 0$ nós é $(m - 1)n + 1$.

14. Mostrar que uma árvore m-ária completa é aquela que possui altura mínima dentre todas as árvores m-árias, $m > 1$, com $n > 0$ nós.

15. O *percurso em altura* de uma árvore binária é aquele em que os nós são dispostos em ordem não decrescente de suas alturas. Descrever um algoritmo para efetuar um percurso em altura de uma árvore binária.

16. O percurso de uma árvore em pré-ordem resultou na impressão da sequência ABCFHDLMPNEGI, e o percurso da mesma árvore em ordem simétrica resultou em FCHBDLPMNAIGE. Construa uma árvore que satisfaça esses percursos. Ela é única?

17. Escreva uma função que crie uma cópia de uma árvore binária. Essa função deve obedecer ao protótipo: `Node* bt_copia(Node *root);`

18. Escreva uma função que retorne a quantidade de nós de uma árvore binária que possuem apenas um filho. Essa função deve obedecer ao protótipo:
    `int um_filho(Node *root);`

19. Escreva funções que determinem se uma árvore binária é:

    (a) estritamente binária.

    (b) cheia.

    (c) completa.

20. [Árvores AVL] Uma árvore é balanceada no sentido AVL se, para todo nó x, as alturas das subárvores esquerda e direita de x diferem em no máximo uma unidade. Escreva uma função que decida se uma dada árvore é balanceada no sentido AVL. Sua função deve respeitar o protótipo `bool eh_avl(NoArv *raiz);`

21. Escreva uma função não-recursiva que realize o percurso em pré-ordem de uma árvore binária. Sugestão: utilizar uma pilha.

22. Escreva uma função não-recursiva que realize o percurso em in-ordem (ordem simétrica) de uma árvore binária. Sugestão: utilizar uma pilha.

23. Escreva uma função não-recursiva que realize o percurso em pós-ordem de uma árvore binária. Sugestão: faça uma versão que utiliza duas pilhas, depois faça outra que utiliza apenas uma pilha.

24. Escreva uma função **não-recursiva** que calcula o número de nós de uma árvore binária dada como entrada. Sua função deve obedecer o seguinte protótipo:
`int numNos_iterativo(NoArv *no);`

25. Escreva uma função que compara se duas árvores binárias dadas como entrada são iguais.

26. A árvore binária vista em sala não possui campo para o nó pai. Adicione o campo `pai` ao `struct NoArv`. Esse campo é um ponteiro para o nó pai de cada nó. Então, escreva uma função que recebe como parâmetro a raiz da árvore e preencha corretamente todos os campos pai de cada um dos nós da árvore binária. Note que o nó raiz não tem pai e o seu campo pai deve apontar para `nullptr` ao final da execução da função.

27. Escreva um algoritmo que receba uma expressão matemática (composta por operandos compostos por um único algarismo, operações de $+$, $-$, $*$, / e parênteses) representada por um string e retorne uma árvore binária representando esta expressão.

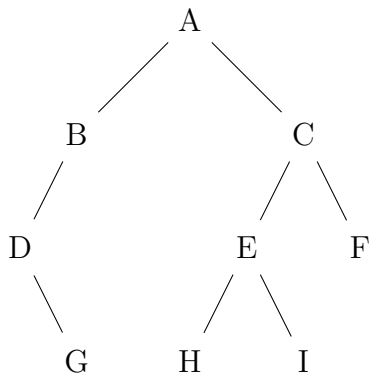
28. Dada uma árvore binária que represente uma expressão matemática, construa um algoritmo que imprima a versão infixa (in-ordem) da expressão.

29. Dada uma árvore binária que represente uma expressão matemática, construa um algoritmo que imprima a versão posfixa (ou pós-ordem) da expressão.

30. Desenhe a árvore binária correspondente às seguintes sequências em pré-ordem e inordem: [1 2 3 4 5 6 7 8 9] e [3 2 6 5 4 1 7 8 9], respectivamente.

31. O percurso de uma árvore em pré-ordem resultou na impressão da sequência ABCFHDLMPNEGI, e o percurso da mesma árvore em ordem simétrica resultou em FCHBDLPMNAIGE. Construa uma árvore que satisfaça esses percursos. Ela é única?

32. [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.

Escreva uma função que recebe uma árvore binária como parâmetro e percorre em nível a árvore binária. Sugestão: utilizar a estutura de dados fila.

33. Uma árvore **quase-cheia** é uma árvore em que todos os níveis, exceto o último, possuem o máximo número de nós possível. No caso em que o último nível não possua o máximo número de nós possíveis, para a árvore ser considerada quase-cheia, todos os nós deste nível devem estar o mais à esquerda possível.

    Considere que uma árvore vazia é representada como um ponteiro nulo. (Uma árvore vazia é, por default, uma árvore quase-cheia.) Implemente uma função em linguagem C++ que receba uma árvore como parâmetro e retorne um inteiro diferente de 0 se a árvore é quase-cheia ou retorne 0 se a árvore não é quase-cheia. Sua função deve obedecer o seguinte protótipo: `int bt_quaseCheia(Node *root);`

    Não é permitido implementar funções auxiliares, ou seja, a função `quaseCheia` deve ser autocontida (porém é permitido chamadas recursivas, se necessário).

    Tarefa adicional: Note que a função deve retornar um inteiro diferente de 0 caso a árvore seja quase-cheia. Portanto, o retorno pode ser **tanto positivo quanto negativo**, o que permite representar se a árvore é quase-cheia e cheia ao mesmo tempo ou se ela é apenas quase-cheia. (Uma árvore **cheia** é uma árvore em que todos os níveis possuem o máximo número de nós possível.)