MAC–IME–USP

Carlos Eduardo Ferreira
Sala 108C Tel.: 3091 6079
e-mail cef@ime.usp.br

MAC0121 - Algoritmos e Estruturas de Dados I

Segundo semestre de 2018

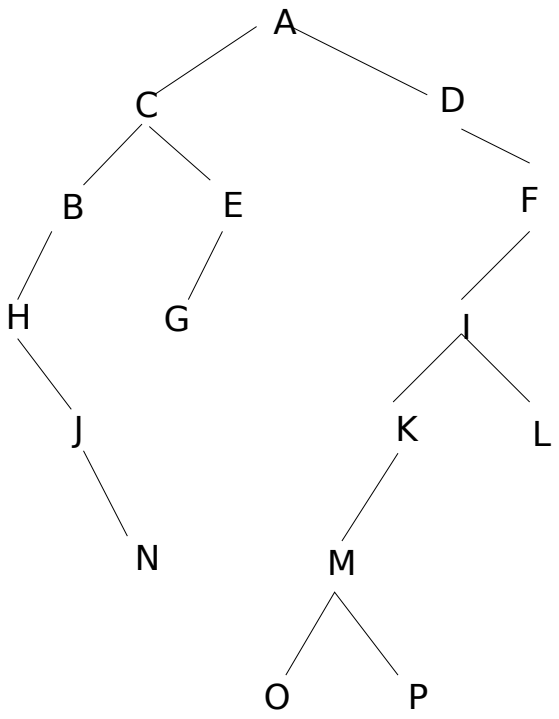
Lista de exercícios – Árvores.

1. Faça uma função que conta o número de nós de uma árvore binária.

2. Faça uma função que recebe dois apontadores para nós x e y de uma árvore binária e decide se x é ancestral de y.

3. Refaça o exercício anterior supondo que a implementação de árvore tenha, em cada nó, um ponteiro para o nó pai.

4. Considere a árvore binária abaixo:



   Liste os vértices visitados quando a árvore é percorrida em pré-ordem, in-ordem e pós-ordem.

5. Faça uma função que recebe dois vetores com a lista de nós visitados em uma árvore binária quando percorrida em pré-ordem e in-ordem e devolve uma cópia da árvore original.

6. Considere uma implementação de árvore binária em que temos um ponteiro para o pai de cada nó (o pai da raiz é `NULL`). A **profundidade** de um nó é a distância entre o nó e a raiz da árvore. Por exemplo, a profundidade da raiz é 0, dos seus filhos é 1, e assim por diante. Faça uma função que recebe um nó de uma árvore e determina sua profundidade.

7. Escreva uma função que imprima o conteúdo de cada nó de uma árvore binária precedido de um recuo em relação à margem esquerda do papel proporcional à sua profundidade.

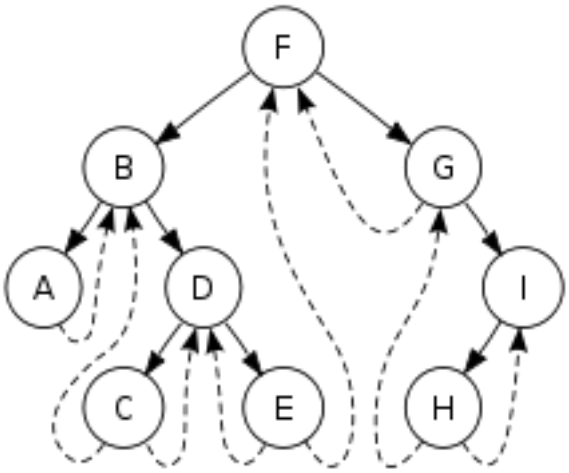
8. Faça uma função que recebe um nó de uma árvore binária e devolve um ponteiro para o próximo nó quando a árvore é percorrida em in-ordem. Pode considerar a implementação com o ponteiro para o pai, se você preferir.

9. Considere uma árvore binária com n elementos. Mostre que $n + 1$ dos ponteiros nos nós da árvore são iguais a `NULL`.

10. Considere a seguinte implementação de árvores binárias:

```
typedef struct cel {
              elemento info;
              int efio, dfio;
              struct cel * esq;
              struct cel * dir;
        } no;
typedef no * apontador;
```

A ideia é utilizar os apontadores `NULL` (que são muitos, como vimos no último exercício) para apontar, por exemplo, para o sucessor/antecessor do nó em in-ordem. Neste caso, a variável `efio` (resp. `dfio`) teria valor 1 se o apontador `esq` (resp. `dir`) apontar para o antecessor (resp. sucessor) em in-ordem e 0 se for o filho esquerdo (resp. direito) do nó. A figura abaixo (extraída da wikipedia) mostra uma árvore de busca binária com fios (veja *threaded binary tree* na wikipedia).



Faça uma função que recebe uma árvore binária e coloca os fios conforme descrito acima.

11. Ao buscar o elemento 555 em uma árvore de busca binária é possível ter passado pelos seguintes nós? Justifique detalhadamente.
    a. 444, 333, 765, 513, 525, 555;
    b. 500, 700, 680, 515, 600, 535, 571, 566, 550, 555;
    c. 700, 340, 398, 610, 380, 412, 580, 555.

12. Considere a implementação de árvore que tem um apontador para o pai de cada nó. Escreva a função:

    `apontador insere (apontador raiz, elemento x)`

    que recebe uma árvore e devolve um apontador para a raiz da árvore em que `x` é inserido, **atualizando o apontador para o pai**.

13. Suponha que os elementos são inseridos em uma árvore de busca binária inicialmente vazia na seguinte ordem:

    $$50, 30, 70, 20, 40, 60, 80, 15, 25, 35, 45, 36$$

    Desenhe a árvore resultante. Em seguida, remova o nó de conteúdo 30.

14. Faça funções de protótipo

    `int menorQue (apontador raiz, elemento x)`

    `int maiorQue (apontador raiz, elemento x)`

    que devolve 1 se todos os elementos da árvore apontada por raiz forem menores (resp. maiores) que `x`.

15. Faça uma função de protótipo

    `int ehABB (apontador raiz)`

    que recebe um apontador para uma raiz de uma árvore binária e devolve 1 se a árvore é de busca binária e 0 caso contrário. Escreva versões iterativa e recursiva.