Estruturas de dados elementares: Árvore Binária

Prof. Bruno de Castro Honorato Silva

February 10, 2021

1 Introdução

Uma Árvore Binária (AB) T é um conjunto finito de elementos, denominados nós ou vértices, tal que se $T = \emptyset$, a árvore é dita vazia, ou T contém um nó especial r, chamado raiz de T, e os demais nós podem ser subdivididos em dois sub-conjuntos distintos T_E e T_D, os quais também são árvores binárias (possivelmente vazias) T_E e T_D são denominados sub-árvore esquerda e sub-árvore direita de T, respectivamente. A Figura 1 ilustra uma árvore binária.

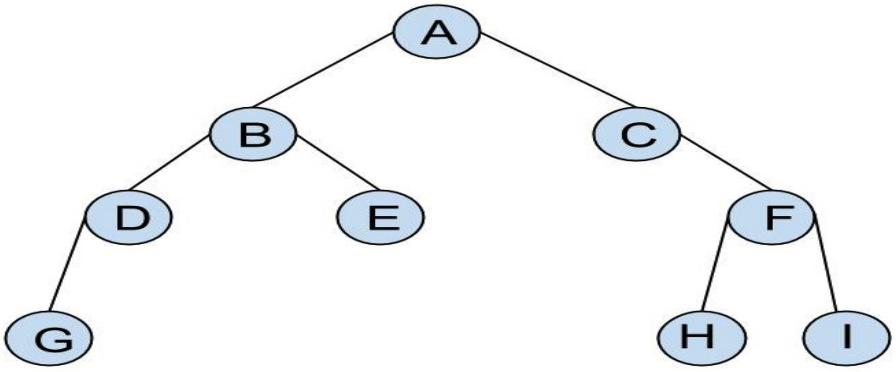


Figure 1: Representação gráfica de uma árvore binária.

2 Definições

A raiz da sub-árvore esquerda de um nó v, se existir, é denominada filho esquerdo de v. A raiz da sub-árvore direita de um nó v, se existir, é denominada filho direito de v. Pela natureza da árvore binária, o filho esquerdo pode existir sem o direito, e vice-versa. Uma Árvore Estritamente Binária (ou Árvore Própria) tem nós com zero (nenhum) ou dois filhos. Nós interiores (não folhas) sempre têm dois filhos. A Figura 2 ilustra uma Árvore Estritamente Binária.

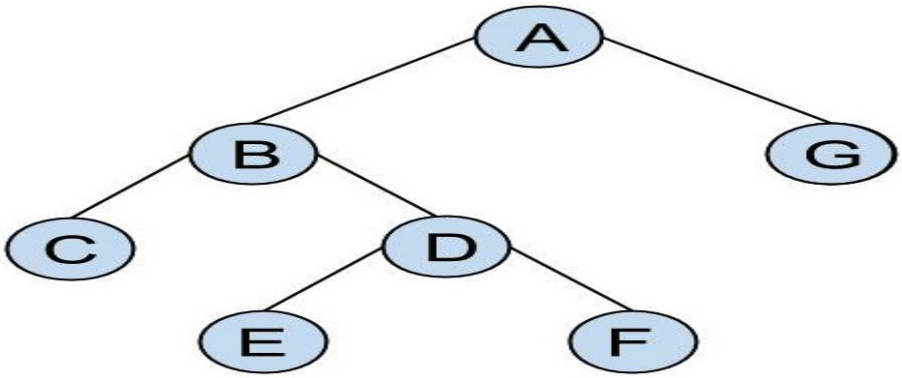


Figure 2: Representação gráfica de uma Árvore Estritamente Binária.

Uma Árvore Binária Completa (ABC) de profundidade d é uma Árvore Estritamente Binária onde cada nó folha está no nível $d-1$ ou no nível d (Figura 3). O nível $d-1$ deve estar totalmente preenchido e os nós folha no nível d devem estar mais à esquerda possível.

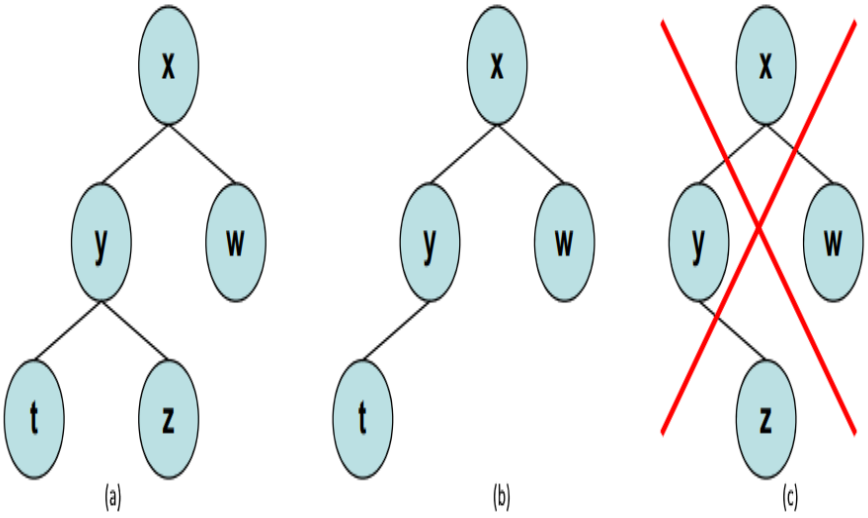


Figure 3: Exemplos gráficos de duas árvores binárias completas (itens a e b) e não completas (c).

Uma Árvore Binária Completa é classificada como Árvore Binária Completa Cheia (ABCC) se possuir todos os seus nós folha no mesmo nível (Figura 4).

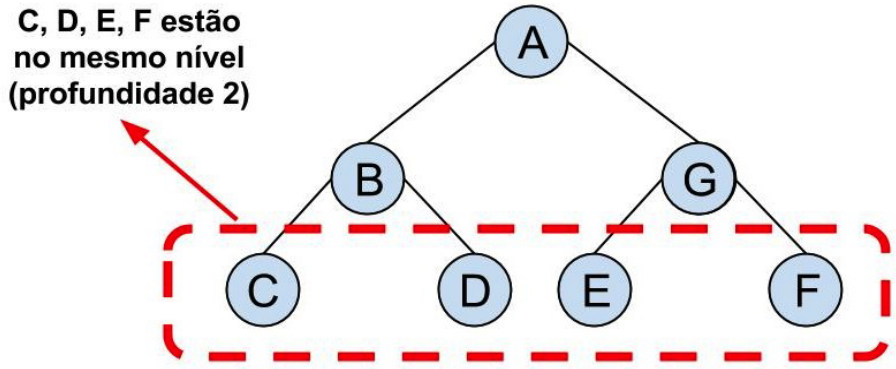


Figure 4: Exemplo gráfico de uma ÁBCC.

Dada uma ABCC e sua profundidade d, pode-se calcular o número total de nós na árvore

- $d = 0$: 1 nó (total 1 nó)
- $d = 1$: 2 nós (total 3 nós)
- $d = 2$: 4 nós (total 7 nós)
- ...
- Profundidade d: $2d$ nós (total $2^{d+1} - 1$).

Portanto, o número de nós, n, para uma ABCC de profundidade d é $n = 2^{d+1} - 1$. Já altura de uma ÁB é $\log_2 n$.

3 Implementações

As implementações desta seção consideram que estaríamos a implementar uma Árvore Binária para armazenar números inteiros. A implementação em C de uma Árvore Binária de inteiros pode ser feita com as estruturas (*structs*) apresentadas na Figura 5.

As operações de criação de um objeto do tipo Árvore Binária e de um nó raiz para está árvore tem suas implementações ilustradas na Figura 6.

Uma forma de se implementar a inserção para esta Árvore Binária é especificando três parâmetros: um valor w (número inteiro); um nó v já inserido na árvore o qual irá ter como filho o nó v_f (leia v índice f - a letra f é uma alusão ao termo *filho*, ou seja, filho de v) de que será criado para armazenar w; e, a informação do lado de v que irá apontar para v_f. A Figura 7 apresenta a implementação em C desta operação.

Vimos que em um vetor de dados, a ordem com que se dá a busca por um elemento pode ser linear ou binária (se o vetor está ordenado). Já a ordem de visitação dos elementos (nós) que compõem uma Árvore Binária varia conforme a aplicação. Denomina-se como *percurso* a sequência de nós visitados de um operação. Existem três ordens para se percorrer os nós de uma estrutura de dados do tipo Árvore Binária. São elas:

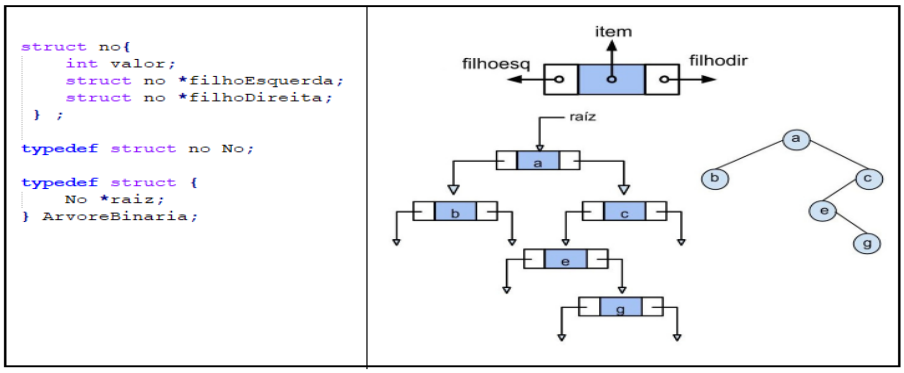


Figure 5: Exemplo de implementação em C dos tipos estruturados de uma AB.

```c
ArvoreBinaria *create() {
    ArvoreBinaria *arv = (ArvoreBinaria *)malloc(sizeof(ArvoreBinaria));
    if (arv != NULL) {
       arv->raiz = NULL;
    }
    return arv;
}

//criar raiz (root)
No *createRoot(ArvoreBinaria *arvore, int valor) {
    arvore->raiz = (No *) malloc(sizeof (No));
    if (arvore->raiz != NULL) {
        arvore->raiz->filhoEsquerda = NULL;
        arvore->raiz->filhoDireita = NULL;
        arvore->raiz->valor = valor;
    }
    return arvore->raiz;
}
```

Figure 6: Exemplo de implementação em C de operações de criação de objetos de árvores binárias.

- Pré-ordem: inicialmente, visita-se a raiz, em seguida, percorre-se a sub-árvore a esquerda em pré-ordem, e, posteriormente, percorre-se a sub-árvore a direita em pré-ordem;
- Em-ordem: inicialmente, percorre-se a sub-árvore a esquerda em em-ordem, em seguida, visita-se a raiz, e, posteriormente, percorre-se a sub-árvore a direita em em-ordem;
- Pós-ordem: inicialmente, percorre-se a subárvore a esquerda em pós-ordem, em seguida, percorre-se a subárvore a direita em pós-ordem, e, finalmente, visita-se a raiz.

Seja $T = \{A, B, C, D, E, F, G, H, I\}$ uma AB. É necessário as seguintes funções:

1. Recebe um árvore e imprime todos os seus elementos em pré-ordem.
2. Recebe um árvore e imprime todos os seus elementos em-ordem;

```c
#define LADO_ESQ 0 //A CRIAÇÃO DESTAS CONSTANTES EVITA O USO DE NÚMEROS
#define LADO_DIR 1 //MÁGICOS E PADRONIZA A REFERÊNCIA A ESTA INFORMAÇÃO.

No *inserir_filho(int lado, No *v, int w) {
    No *v_f = (No *) malloc(sizeof (No));

    if (v_f != NULL) {
        v_f->filhoDireita = NULL;
        v_f->filhoEsquerda = NULL;
        v_f->valor = w;

        if (lado == LADO_ESQ) {
            v->filhoEsquerda = v_f;
        } else {
            v->filhoDireita = v_f;
        }
    }

    return v_f;
}
```

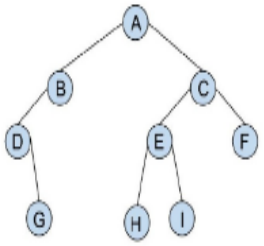
Figure 7: Exemplo de implementação em C da operação de inserção em árvore.

3. Recebe um árvore e imprime todos os seus elementos em pós-ordem.

As implementações para estas funções são apresentadas nas Figuras que se seguem.

```c
void pre_ordem_aux(No *raiz) {
    if (raiz != NULL) {
        printf("%i", raiz->valor);
        pre_ordem_aux(raiz->filhoEsquerda);
        pre_ordem_aux(raiz->filhoDireita);
    }
}

void pre_ordem(ArvoreBinaria *arvore) {
    pre_ordem_aux(arvore->raiz);
}
```

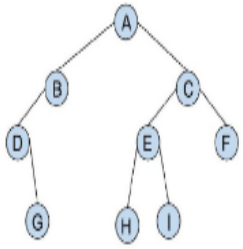


- Resultado: ABDGCEHIF

Figure 8: Exemplo de implementação em C da operação de impressão de elementos em uma AB em pré-ordem.

```c
void em_ordem_aux(No *raiz) {
    if (raiz != NULL) {
        em_ordem_aux(raiz->filhoEsquerda);
        printf("%i", raiz->valor);
        em_ordem_aux(raiz->filhoDireita);
    }
}

void em_ordem(ArvoreBinaria *arvore) {
    em_ordem_aux(arvore->raiz);
}
```

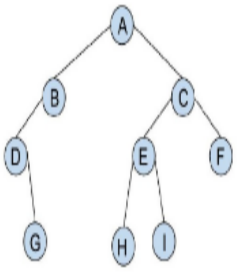


• Resultado: DGBAHEICF

Figure 9: Exemplo de implementação em C da operação de impressão de elementos em uma AB em-ordem.

```c
void pos_ordem_aux(No *raiz) {
    if (raiz != NULL) {
        pos_ordem_aux(raiz->filhoEsquerda);
        pos_ordem_aux(raiz->filhoDireita);
        printf("%i", raiz->valor);
    }
}

void pos_ordem(ArvoreBinaria *arvore) {
    pos_ordem_aux(arvore->raiz);
}
```



• Resultado: GDBHIEFCA

Figure 10: Exemplo de implementação em C da operação de impressão de elementos em uma AB pós-ordem.