**Lista de Exercícios de árvores binárias**

1) Suponha que T é a árvore binária armazenada na memória, como na figura. Desenhe o diagrama de T.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INFO | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | | | 35 | 45 | 55 | 95 |
| LEFT | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 7 | | | 0 | 3 | 11 | 0 |
| RIGHT | 0 | 13 | 0 | 0 | 6 | 8 | 0 | 14 | | | 12 | 4 | 0 | 0 |

RAIZ $\boxed{5}$

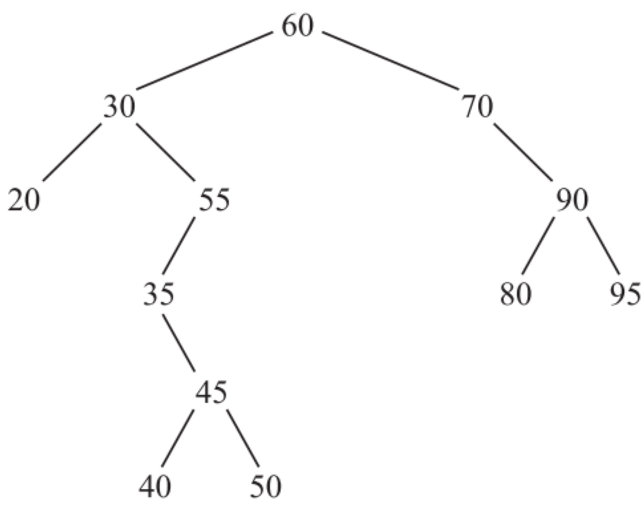
A árvore T é esboçada a partir de sua raiz R para baixo como se segue:

(a) A raiz R é obtida a partir do valor do apontador ROOT. Note que ROOT = 5. Logo, INFO[5] = 60 é a raiz R de T.

(b) O filho à esquerda de R é conseguido a partir do campo apontador esquerdo de R. Note que LEFT[5] = 2. Logo, INFO[2] = 30 é o filho à esquerda de R.

(c) O filho à direita de R é conseguido a partir do campo apontador direito de R. Note que RIGHT[5] = 6. Logo, INFO[6] = 70 é o filho à direita de R.

Podemos agora esboçar a parte superior da árvore e, então, repetindo o processo com cada novo nó, finalmente obtemos a árvore T inteira na Fig.

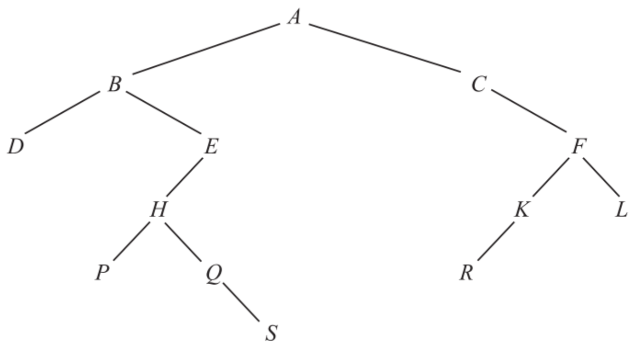



(*a*)

2) Seja T a árvore binária armazenada na memória, como na Fig. 10-34, onde ROOT = 14.
   a. Esboce o diagrama de T.
   b. Percorra T em: (i) pré-ordem; (ii) inordem; (iii) pós-ordem.
   c. Determine a profundidade d de T.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INFO | *H* | *R* | | *P* | *B* | | *E* | | *C* | *F* | *Q* | *S* | | *A* | *K* | *L* | | *D* |
| LEFT | 4 | 0 | | 0 | 18 | | 1 | | 0 | 15 | 0 | 0 | | 5 | 2 | 0 | | 0 |
| RIGHT | 11 | 0 | | 0 | 7 | | 0 | | 10 | 16 | 12 | 0 | | 9 | 0 | 0 | | 0 |

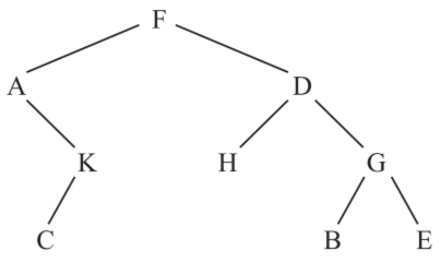
(a) Diagrama T




(b) Pré-Ordem (RED): raiz, esquerda, direita - ABDEHPQSCFKRL
In-Ordem(ERD): esquerda, raiz, direita - DBPHQSEACRKFL
Pós-Ordem(EDR): esquerda, direita, raiz - DPSQHEBRKLFCA;

(c) d = 6 – (Tamanho do maior ramo)

3) Considere a árvore binária T na Figura.
   a. Encontre a profundidade d de T.
   b. Percorra T, usando o algoritmo de pré-ordem.
   c. Percorra T, usando o algoritmo inordem.
   d. Percorra T, usando o algoritmo pós-ordem.
   e. Encontre os nós terminais de T e a ordem em que eles são percorridos em (b), (c) e (d).




a) A profundidade d é o número de nós no mais longo ramo de T; logo, d = 4.
b) O percurso pré-ordem de T é um algoritmo recursivo NLR, ou seja, primeiro, ele processa um nó N, em seguida, sua subárvore L à esquerda e, finalmente, sua subárvore à direita R. Denotando [A1,..., Ak] como uma subárvore com nós A1,..., Ak, a árvore T é percorrida como se segue:
   F −[A, K, C][D, H, G, B, E] ou F − A −[K, C]−D−[H][G, B, E]
   ou, finalmente, F −A−K−C−D−H−G−B−E
c) O percurso inordem de T é um algoritmo recursivo LNR, isto é, primeiro processa uma subárvore L à esquerda, em seguida, seu nó e, finalmente, sua subárvore à direita R. Assim, T é percorrida como se segue:
   [A, K, C]−F −[D, H, G, B, E] ou
   A−[K, C]−F −[H]−D−[G, B, E] ou,
   finalmente, A−K−C−F −H−D−B−G−E
d) O percurso pós-ordem de T é um algoritmo recursivo LRN, ou seja, primeiro, processa uma subárvore à esquerda L, em seguida, sua subárvore à direita R e, finalmente, seu nó N. Logo, T é percorrida como se segue:
   [A, K, C][D, H, G, B, E]−F ou
   [K, C]−A−[H][G, B, E]−D−F ou,
   finalmente, C−K−A−H−B−E−G−D−F
e) Os nós terminais são aqueles sem filhos. Eles são percorridos na mesma ordem em todos os três algoritmos de percurso: C, H, B, E.

4) Seja T a árvore binária do exercício anterior. Encontre a representação sequencial de T na memória.

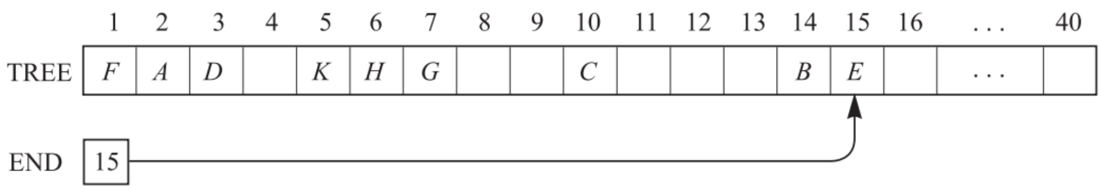
A representação sequencial de T emprega apenas um array TREE e uma variável apontadora END.

(a) A raiz R de T é armazenada em TREE[1]; logo, R = TREE[1] = F.

(b) Se o nó N ocupa TREE[K], então seus filhos à esquerda e à direita são armazenados em TREE[2∗K] e TREE[2∗K + 1], respectivamente. Logo, TREE[2] = A e TREE[3] = D, uma vez que A e D são os filhos à esquerda e à direita de F, e assim por diante.

Observe que TREE[10] = C, pois C é o filho à esquerda de K, que é armazenado em TREE[5]. Além disso, TREE[14] = B e TREE[15] = E, pois B e E são os filhos à esquerda e à direita de G, que é armazenado em TREE[7].

(c) END aponta para a localização do último nó de T; logo, END = 15.



5) Uma árvore binária T tem nove nós. Esboce uma representação visual de T se o percurso pré-ordem e inordem de T levam às seguintes sequências de nós:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Pré-ordem: | $G$ | $B$ | $Q$ | $A$ | $C$ | $P$ | $D$ | $E$ | $R$ |
| Inordem: | $Q$ | $B$ | $C$ | $A$ | $G$ | $P$ | $E$ | $D$ | $R$ |

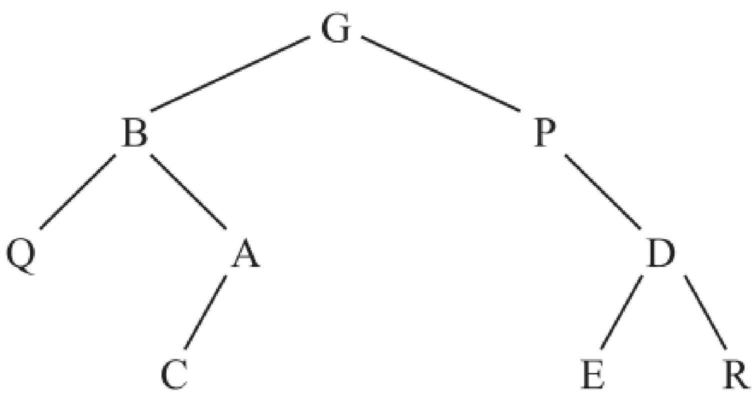
A árvore T é esboçada a partir de sua raiz R para baixo como se segue.

(a) A raiz de T é obtida, escolhendo o primeiro nó em sua pré-ordem. Assim, G é a raiz de T.

(b) O filho à esquerda do nó G é conseguido como se segue. Primeiro, empregamos a inordem de T para encontrar os nós na subárvore à esquerda T1 de G. Assim, T1 consiste nos nós Q, B, C, A, que estão à esquerda de G na inordem de T. Em seguida, o filho à esquerda de G é obtido, escolhendo o primeiro nó (raiz) na pré-ordem de T1, que aparece na pré-ordem de T. Portanto, B é o filho à esquerda de G.

(c) analogamente, a subárvore à direita T2 de G consiste nos nós P, E, D, R; e P é a raiz de T2, isto é, P é o filho à direita de G.

Repetindo o processo acima com cada novo nó, finalmente conseguimos a árvore T exigida na Fig.

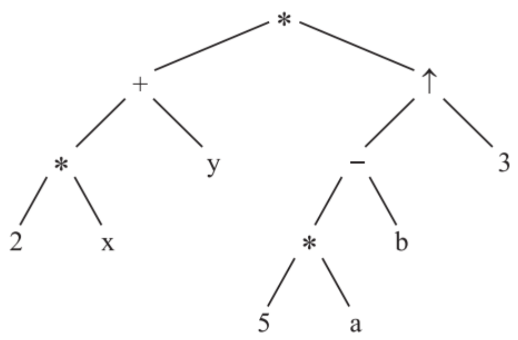


6) Considere a expressão algébrica $E = (2x + y)(5a - b)^3$

a. Desenhe a 2-árvore correspondente.

b. Use T para escrever E na forma prefixa polonesa.

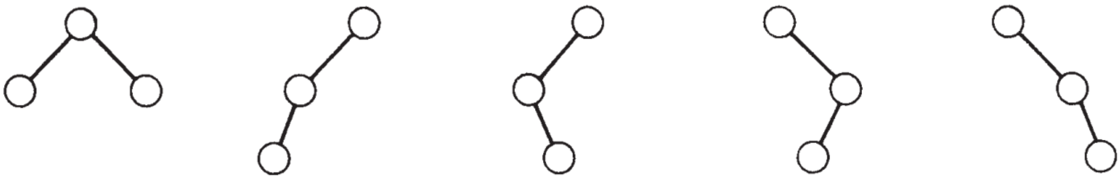
(a) Use uma flecha (↑) para potência, um asterisco (∗) para multiplicação e uma barra (/) para divisão, para obter a árvore da Fig.



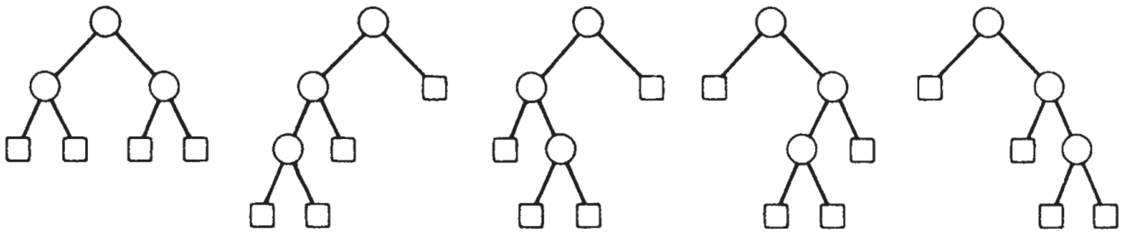

(b) Escaneie a árvore a partir da esquerda, para obter: ∗ + ∗ 2 x y ↑ − ∗ 5 a b 3

7) Esboce todas as possíveis e não semelhantes:
    a. Árvores binárias T com três nós
    b. 2-árvores T’ com quatro nós externos

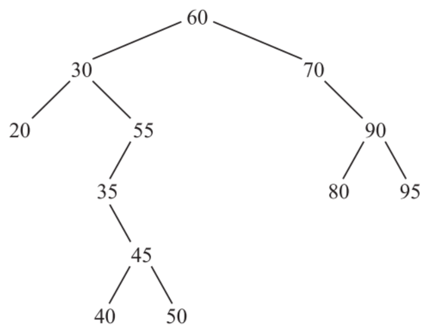
(a) Existem cinco árvores como essas, as quais são representadas na Fig.



(b) Cada 2-árvore T com quatro nós externos é determinada por uma árvore binária T com três nós, ou seja, por uma árvore T no item (a). Logo, há cinco 2-árvores T como essas, as quais são representadas na Fig.



8) Considere a árvore binária T na Figura




(a)

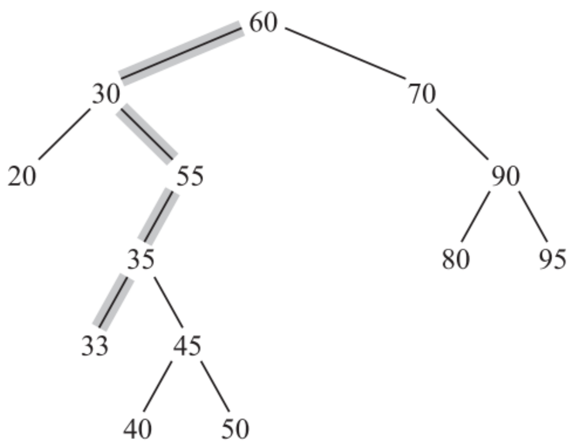
(a) Por que T é uma árvore binária de busca?

(b) Suponha que ITEM = 33 é adicionado à árvore. Encontre a nova árvore T

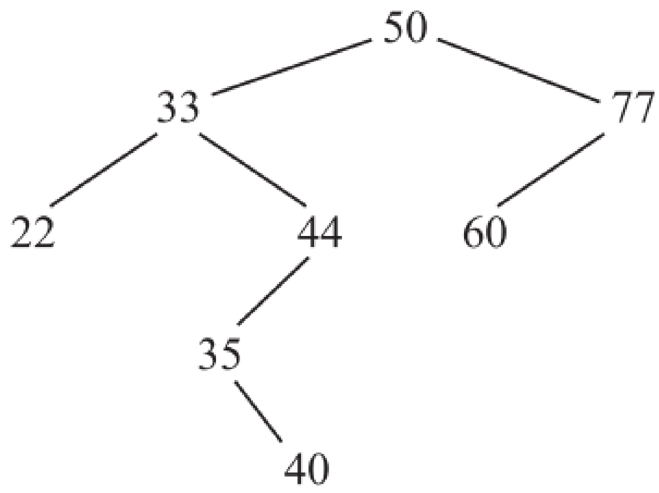
(a) T é uma árvore binária de busca, porque cada nó N é maior do que os valores em sua subárvore à esquerda e menor do que os valores em sua subárvore à direita.

(b) Compare ITEM = 33 com a raiz 60. Como 33 < 60, mova para o filho à esquerda, 30. Como 33 > 30, mova para o filho à direita, 55. Uma vez que 33 < 55, mova para o filho à esquerda, 35. Agora, 33 < 35, mas 35 não admite filho à esquerda.

Logo, adicione ITEM = 33 como um filho à esquerda do nó 35, para termos a árvore da Fig. As arestas sombreadas indicam o caminho para baixo, através da árvore, durante a inserção.



9) Encontre a árvore final T se os seguintes números são inseridos em uma árvore binária vazia de busca T: 50, 33, 44, 22, 77, 35, 60, 40

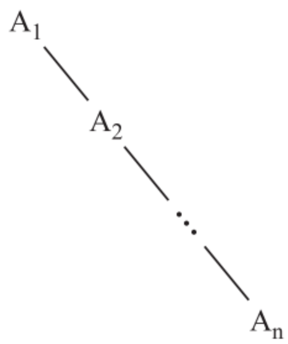


10) Suponha que n itens de dados $A_1, A_2, ..., A_N$ já foram ordenados, isto é, $A_1 < A_2 < \cdots < A_N$
    a. Se os itens são inseridos em ordem em uma árvore binária vazia T, descreva a árvore final T.
    b. Qual a profundidade d da árvore final T?
    c. Compare d com a profundidade média d∗ de uma árvore binária com n nós, para (i) n = 50; (ii) n = 100; (iii) n = 500

(a) A árvore T consiste em um ramo que se estende para a direita, como retratado na Fig.

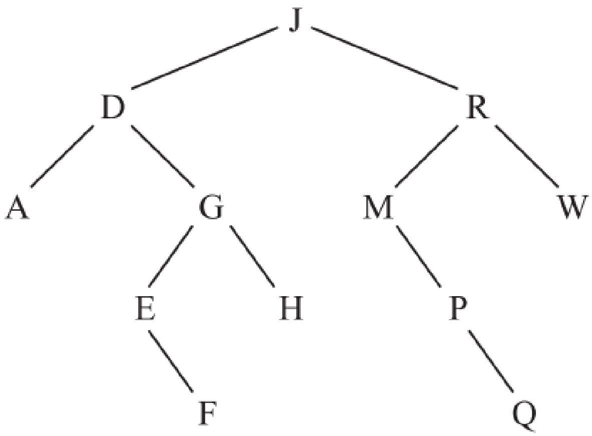
(b) O ramo de T tem n nós; logo, d = n.

(c) Sabe-se que $d* = c \log_2 n$, onde $c \approx 1{,}4$. Logo:
(i) $d(50) = 50$, $d*(50) \approx 8$;
(ii) $d(100) = 100$, $d*(100) \approx 9$;
(iii) $d(500) = 500$, $d*(500) \approx 12$.



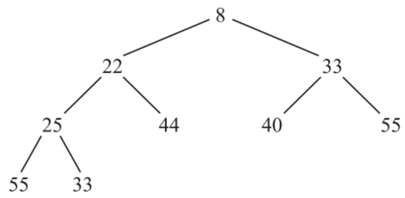
11) Suponha que a seguinte lista de letras é inserida em uma árvore binária vazia: J, R, D, G, W, E, M, H, P, A, F, Q
    a. Encontre a árvore final T
    b. Encontre o percurso inordem de T.

(a) insira os nós, um após o outro, para obter a árvore T da Fig.

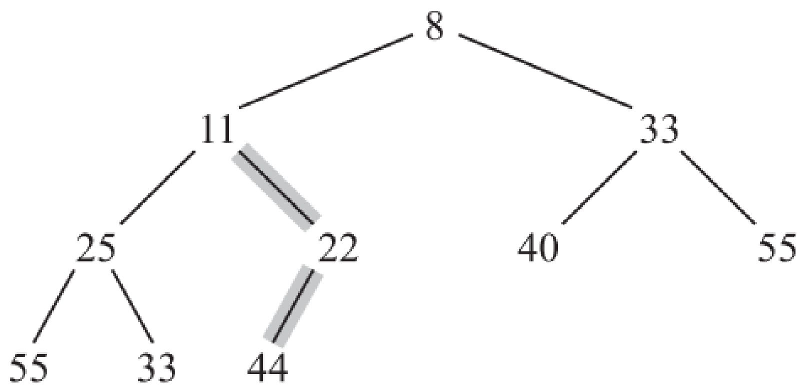



(b) O percurso inordem de T é o que se segue: A, D, E, F, G, H, J, M, P, Q, R, W
Observe que essa é a listagem alfabética das letras. (O percurso inordem de qualquer árvore binária de busca T conduz a uma lista ordenada dos nós.)

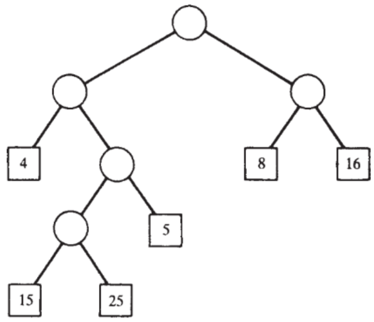
12) Seja H o heap mínimo na Figura. (H é um heap mínimo, uma vez que os elementos menores estão no topo do heap, em vez dos maiores.) Descreva o heap depois que ITEM = 11 é inserido em H.




Primeiro, insira ITEM como o próximo nó da árvore completa, isto é, como o filho à esquerda do nó 44. Em seguida, compare repetidamente ITEM com seu PAI e permute ITEM e PAI enquanto ITEM < PAI. Como 11 < 44, permute 11 e 44. Como 11 < 22, permute 11 e 22. Como 11 > 8, ITEM = 11 encontrou seu lugar apropriado no heap H. A Figura mostra o heap final H. As arestas sombreadas indicam o caminho de ITEM à medida que ele se move sobre a árvore.



13) Seja T a 2-árvore ponderada. Determine o comprimento do caminho ponderado P da árvore T.




Suponha que T é uma 2-árvore com n nós externos e que cada um deles é assinalado a um peso (não negativo). O comprimento do caminho ponderado (ou simplesmente comprimento do caminho) P da árvore T é definido como a soma $P = W_1L_1 + W_2L_2 + \cdots + W_nL_n$

onde $W_i$ é o peso em um nó externo $N_i$, e $L_i$ é o comprimento do caminho da raiz R ao nó $L_i$.

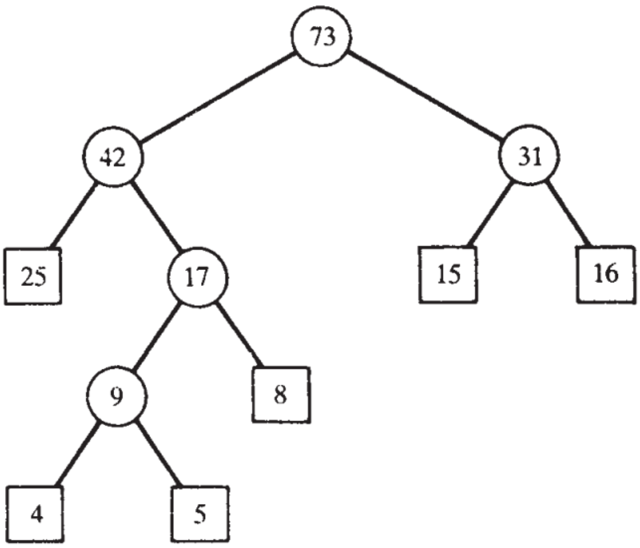
$$P = 4(2) + 15(4) + 25(4) + 5(3) + 8(2) + 16(2) = 8 + 60 + 100 + 15 + 16 + 32 = 231$$

14) Suponha que seis pesos, 4, 15, 25, 5, 8, 16, sejam dados. Encontre uma 2-árvore T com os pesos dados e com um comprimento de caminho P mínimo. (Compare T com a árvore da Figura do exercício anterior)

Use o algoritmo de Huffman. Ou seja, combine repetidamente as duas subárvores com pesos mínimos em uma única subárvore como se segue:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $(a)$ | 4, | 15, | 25, | 5, | 8, | 16; | $(d)$ | 25, | 17, | 31; |
| $(b)$ | | 15, | 25, | 9, | 8, | 16; | $(e)$ | | 42, | 31; |
| $(c)$ | | 15, | 25, | | 17, | 16; | $(f)$ | | | 73, |

(O número circulado indica a raiz da nova subárvore no passo.) A árvore T é esboçada a partir do passo (f) para trás.




P = 25(2) + 4(4) + 5(4) + 8(3) + 15(2) + 16(2) = 50 + 16 + 20 + 24 + 30 + 32 = 172

15) Suponha que itens de dados A, B, C, D, E, F, G ocorram com a seguinte distribuição de probabilidades:

| Itens de dados: | *A* | *B* | *C* | *D* | *E* | F | *G* |
|---|---|---|---|---|---|---|---|
| Probabilidade: | 10 | 30 | 5 | 15 | 20 | 15 | 5 |

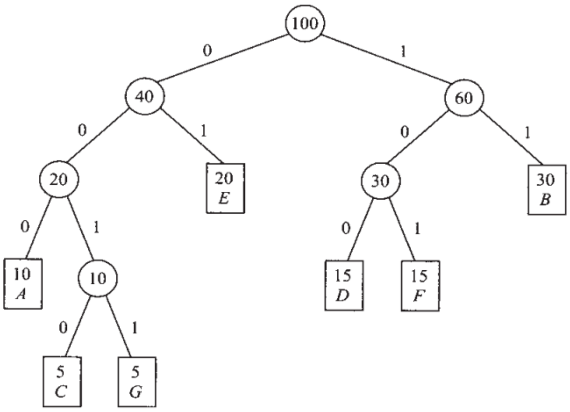
Encontre um código de Huffman para os itens de dados.

Aplique o algoritmo de Huffman para econtrar uma 2-árvore com comprimento de caminho ponderado mínimo P. (Novamente o número circulado indica a raiz da nova subárvore no passo.) A árvore T é esboçada a partir do passo (g) para trás, levando à Figura. Assinale rótulos de bit às arestas da árvore T, 0 para uma aresta à esquerda e 1 para uma aresta à direita. A árvore T nos leva ao seguinte código de Huffman:

A:000; B :11; C:0010; D:100; E:01; F :101; G:0011

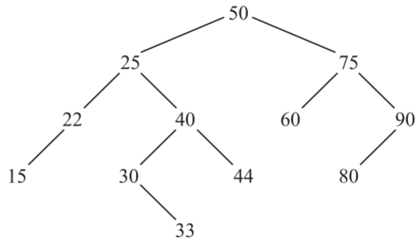
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| (*a*) | 10, | 30, | 5, | 15, | 20, | 15, | 5 |
| (*b*) | 10, | 30, | 10, | 15, | 20, | 15, | |
| (*c*) | 20, | 30, | | 15, | 20, | 15 | |
| (*d*) | 20, | 30, | | 30, | 20 | | |
| (*e*) | 40, | 30, | | 30 | | | |
| (*f*) | 40 | 60, | | | | | |
| (*g*) | | 100 | | | | | |

(*a*)




(*b*)

16) Seja T a árvore binária de busca na Figura. Suponha que os nós 20, 55 e 88 são adicionados, um após o outro, a T. Determine a árvore final T.




**Algoritmo 10.1:** Uma árvore binária de busca $T$ e um ITEM de informação são dados. O algoritmo encontra a localização de ITEM em $T$, ou insere ITEM como novo nó na árvore.

***Passo 1.*** Compare ITEM com a raiz $N$ da árvore.

(*a*) Se ITEM $< N$, proceda para o filho à esquerda de $N$.

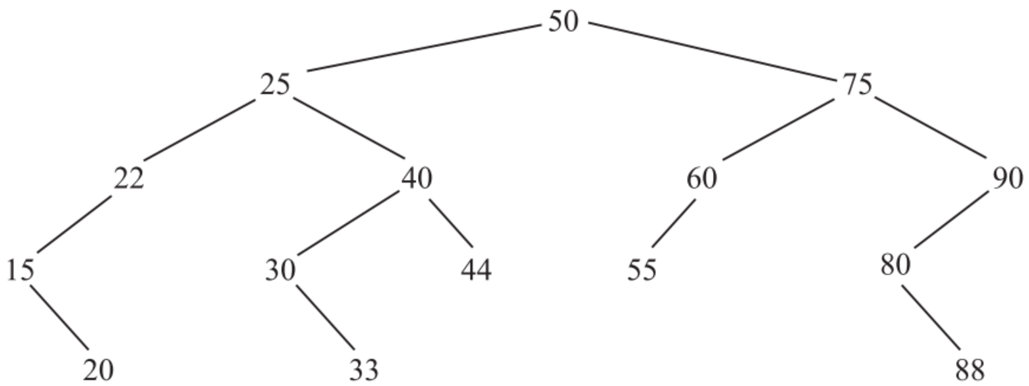
(*b*) Se ITEM $> N$, proceda para o filho à direita de $N$.

***Passo 2.*** Repita o Passo 1 até ocorrer o seguinte:

(*a*) Encontramos um nó $N$ tal que ITEM $= N$. Neste caso, a busca é bem-sucedida.

(*b*) Encontramos uma subárvore vazia, o que indica que a busca é malsucedida. Insira ITEM no lugar da subárvore vazia.

***Passo 3.*** Saída.



17) Seja T a árvore binária de busca do exercício anterior. Suponha que os nós 22, 25 e 75 são removidos, um após o outro, a T. Determine a árvore final T.

**Algoritmo 10.2:** Uma árvore binária de busca $T$ e um ITEM de informação são dados. $P(N)$ denota o pai de um nó $N$, e $S(N)$ corresponde ao sucessor inordem de $N$. O algoritmo deleta ITEM de $T$.

***Passo 1.*** Use Algoritmo 10.1 para encontrar a localização do nó que contém ITEM e rastreie a localização do nó pai $P(N)$. (Se ITEM não estiver em $T$, então pare (STOP) e vá para a saída.)

***Passo 2.*** Determine o número de filhos de $N$. Há três casos:

(*a*) ***N* não tem filhos.** $N$ é deletado de $T$, simplesmente substituindo a localização de $N$ no nó pai $P(N)$ pelo apontador NULL.

(*b*) ***N* tem exatamente um filho *M*.** $N$ é deletado de $T$, substituindo a localização de $N$ no nó pai $P(N)$ pela localização de $M$. (Isso troca $N$ por $M$.)

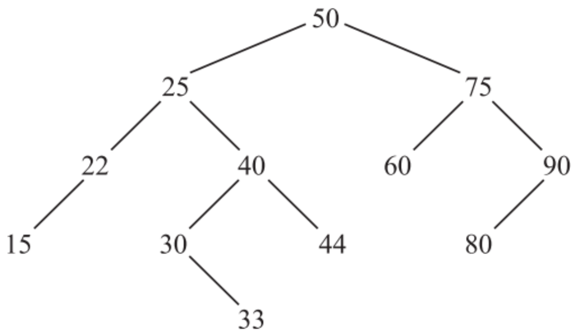
(*c*) ***N* tem dois filhos.**

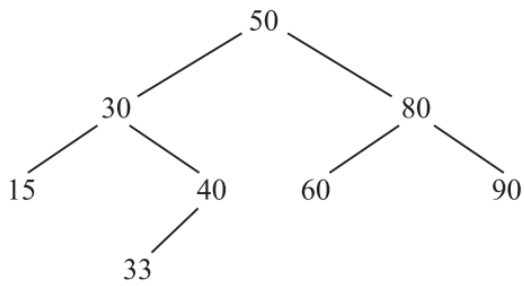
(i) Encontre o sucessor inordem $S(N)$ de $N$. (Então $S(N)$ não tem filho à esquerda.)

(ii) Delete $S(N)$ de $T$, usando (*a*) ou (*b*).

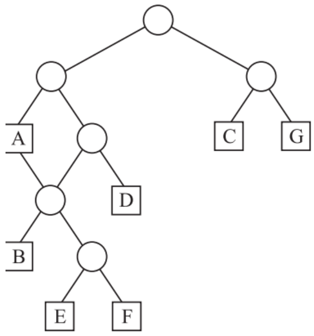
(iii) Substitua $N$ por $S(N)$ em $T$.

***Passo 3.*** Saída.



18) Encontre o comprimento de caminho ponderado P da árvore na Figura se os itens de dados A, B,..., G são assinalados aos seguintes pesos: (A, 13), (B, 2), (C, 19), (D, 23), (E, 29), (F, 5), (G, 9)




P = 329.

P=23(2)+2(4)+19(2)+23(3)+29(5)+5(5)+9(2)