Fundação CECIERJ - Vice Presidência de Educação Superior a Distância
Curso de Tecnologia em Sistemas de Computação – UFF
Disciplina: ESTRUTURA DE DADOS
AD2 - 1° semestre de 2016

AVALIAÇÃO À DISTÂNCIA 2

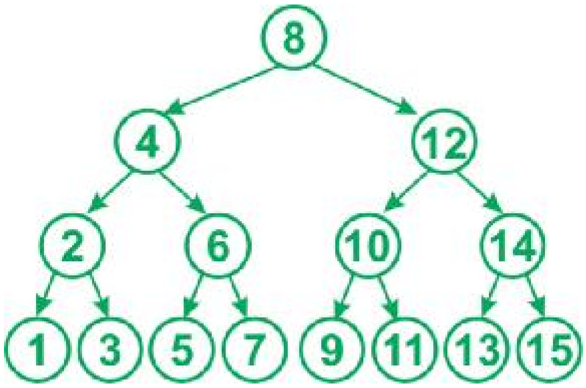
GABARITO ALTERNATIVO (explicado e corrigido)

QUESTÕES:

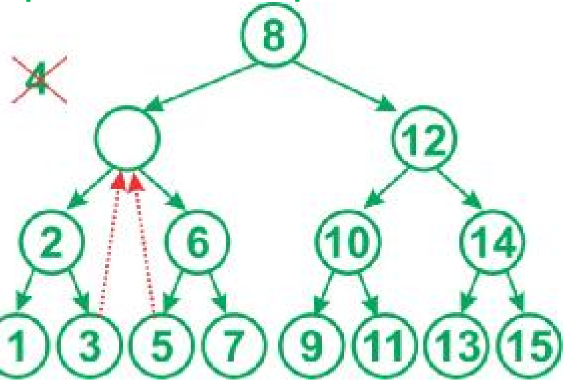
1. (1,0) Suponha que você deseja remover um nó de uma árvore binária de busca. Após removê-lo, como você deve reestruturar a árvore de modo que ela continue sendo uma árvore binária de busca? Dê um exemplo que mostre seu raciocínio. (Sugestão: desenhe uma árvore, remova um nó e reestruture-a.)

Resposta: Existem duas opções: Deve ser colocado no lugar do nó removido: 1) o nó mais a direita da sub-árvore esquerda ou 2) o nó mais a esquerda da sub-árvore direita.

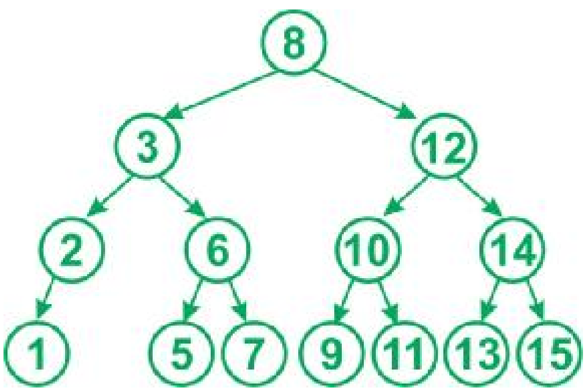
Exemplo: Seja a árvore binária abaixo:

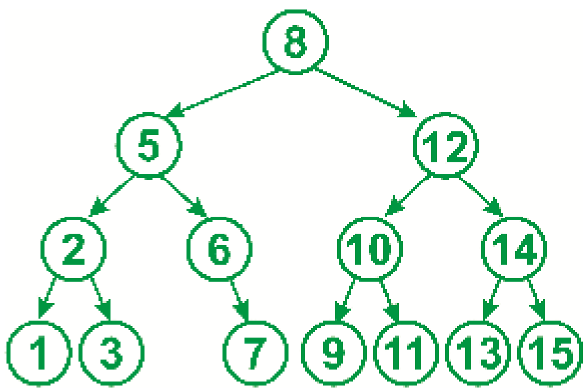


Supondo que desejamos remover o nó 4. Ele poderá ser substituído pelo nó mais à direita da sub-árvore esquerda (nó 3) ou pelo nó mais à esquerda da sub-árvore direita (nó 5):

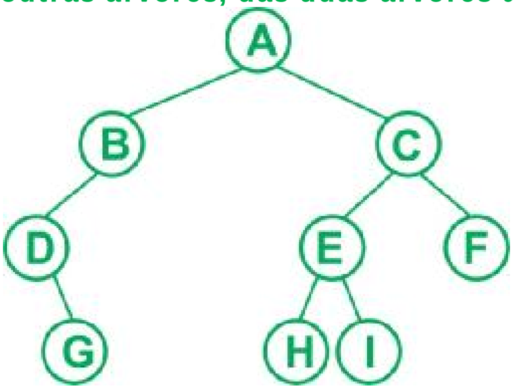


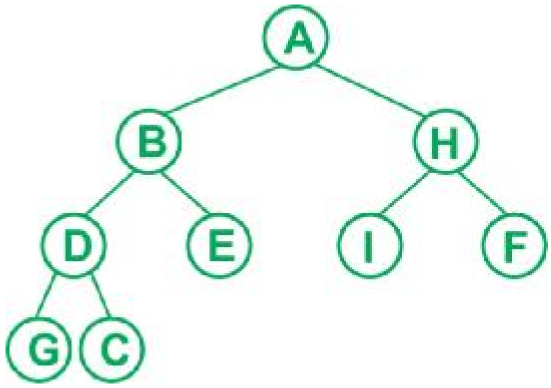
Assim, a árvore resultante será uma das duas abaixo:





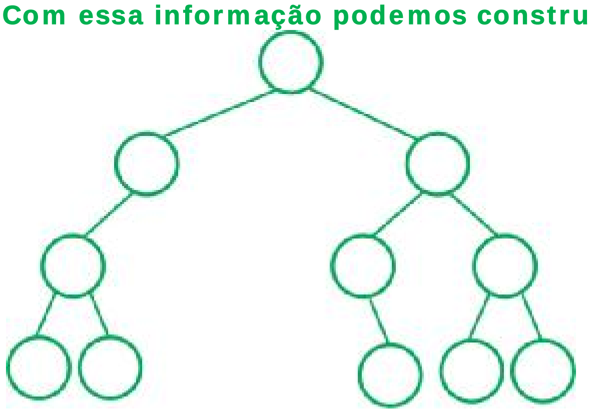
2. (1,5) Prove ou dê um contra-exemplo: Uma árvore binária pode ser construída, de forma única, a partir das seguintes informações:
(i) percurso em nível
Resposta:
Falso. Podemos ter duas árvores binárias diferentes, porém, com o mesmo percurso de nível. Por exemplo, seja o percurso pré-ordem ABDGCEHIF. Podemos obter este percurso, entre outras árvores, das duas árvores abaixo:

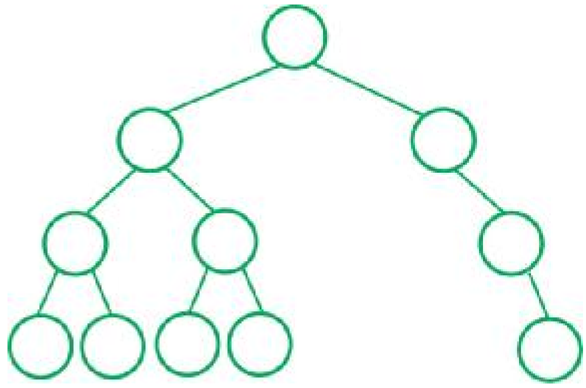




Logo, um percurso em nível não representa uma única árvore.

(ii) número de nós em cada nível.
Resposta:
Falso. Imaginemos uma árvore binária, onde temos: no nível 1, 1 nó (raiz); no nível 2, 2 nós; no nível 3, 3 nós e no nível 4, 5 nós.
Com essa informação podemos construir, por exemplo, as duas árvores distintas abaixo:

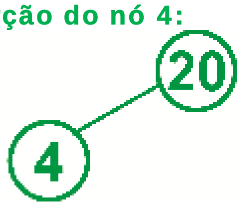




Logo, a informação do número de nós por nível não descreve, de forma única, uma árvore binária.

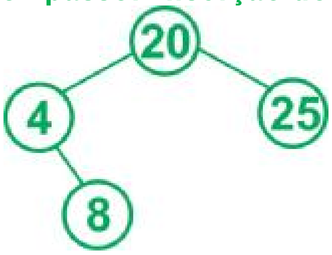
3. (1,5) A partir de uma árvore inicialmente vazia, desenhe o passo a passo e a árvore AVL resultante da inserção de nós com chaves 20, 4, 25, 8, 7, 2, 10, 23 (nesta ordem).
Resposta:
1° passo: Árvore vazia
2° passo: Inserção do nó 20:

3° passo: Inserção do nó 4:

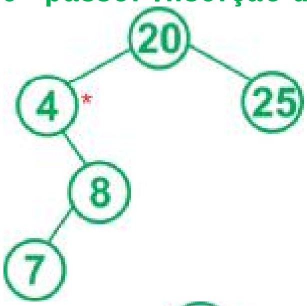


4° passo: Inserção do nó 25:

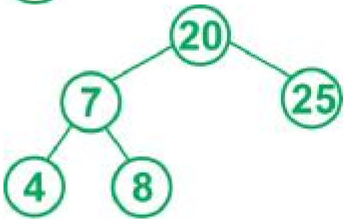
5º passo: Inserção do nó 8:



6º passo: Inserção do nó 7:

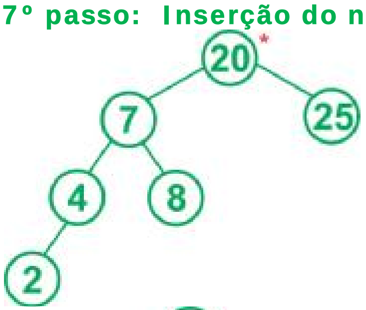


* - Nó desregulado.
q = 7; p = 4; u = 8; v = 7;
$h_D(p) > h_E(p)$ e $h_E(u) > h_D(u)$
Aplicar rotação dupla esquerda (RDE).

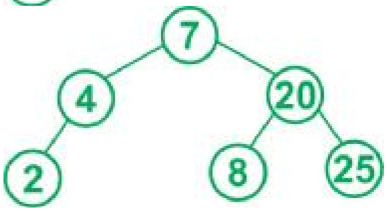


Árvore balanceada.

7º passo: Inserção do nó 2:

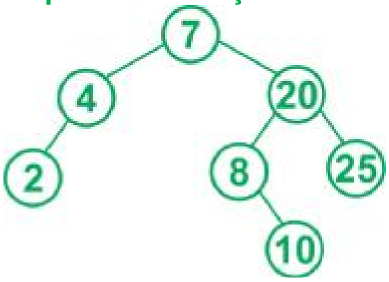


* - Nó desregulado.
q = 2; p = 20; u = 7; v = 4;
$h_E(p) > h_D(p)$ e $h_E(u) > h_D(u)$
Aplicar rotação direita (RD).

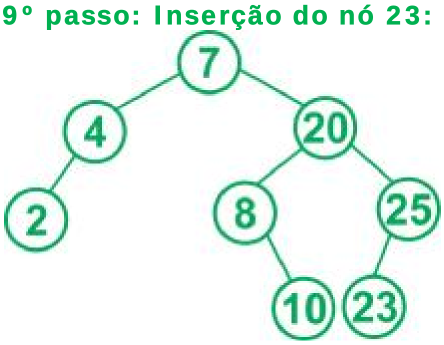


Árvore balanceada.

8º passo: Inserção do nó 10:



9º passo: Inserção do nó 23:

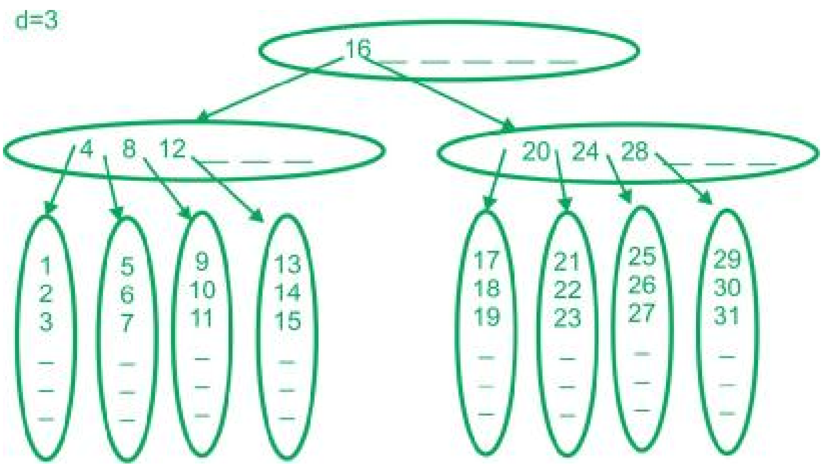


Árvore AVL final, resultante da inserção dos nós 20, 4, 25, 8, 7, 2, 10 e 23 (nesta ordem).

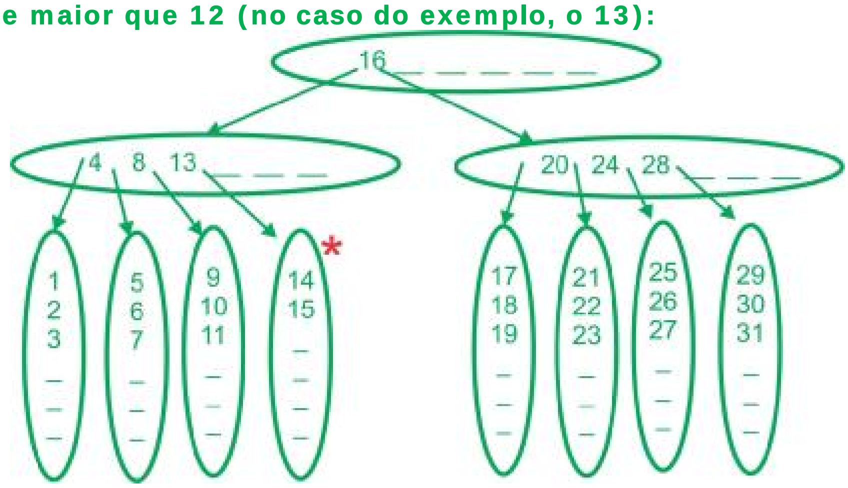
4. (1,5) Desenhe uma árvore B de ordem d = 3 e altura 3 contendo o menor número possível de chaves (Os valores ficam à sua escolha). A seguir, efetue a remoção de uma chave e desenhe a árvore B resultante da remoção.

Resposta:

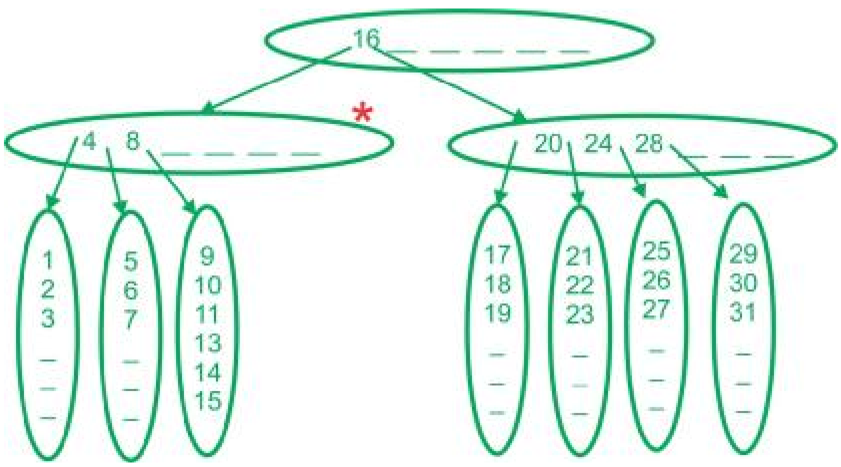
Árvore B de ordem d=3 e h=3 com o menor número possível de chaves:



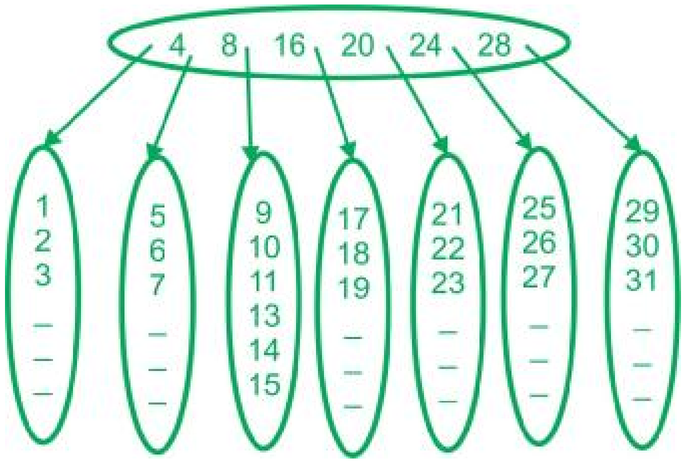
Efetuar a remoção de um nó. Por exemplo, nó 12. Ao remover o nó 12, ele é substituído pelo valor imediatamente maior que 12 (no caso do exemplo, o 13):



A página marcada com * ficou com menos de d chaves (d=3). Como a página tem k=2 chaves e sua irmã ao lado tem m=3 chaves, e k+m < 2d, aplicamos a concatenação das duas páginas.



Após a concatenação, agora página marcada com *, no nível 2, ficou com menos de d chaves (d=3). Como a página tem k=2 chaves e sua irmã ao lado tem m=3 chaves, e k+m < 2d, aplicamos a concatenação das duas páginas. Assim, a concatenação resulta na árvore a seguir:



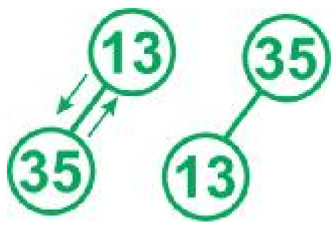
Que também é uma árvore B, de ordem d=3, porém de altura 2, visto que a de altura 3 já tinha o mínimo de chaves para a ordem 3 e altura 3, a remoção de uma chave necessariamente implicaria na redução de sua altura.

5. (1,5) Execute o método de ordenação por heap ("heapsort"), aplicando-o às seguintes prioridades (nesta ordem): 13, 35, 41, 04, 22, 10, 42, 14. Desenhe as configurações sucessivas da àrvore durante o processo de execução do algoritmo.
Resposta:

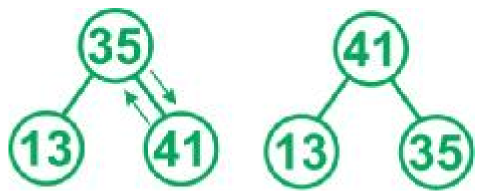
Como a lista de prioridades não é uma heap, o primeiro passo é construir uma heap com as prioridades dadas:

1º - Inserção do 13:

2º - Inserção do 35



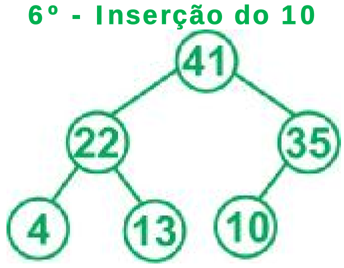
3º - Inserção do 41



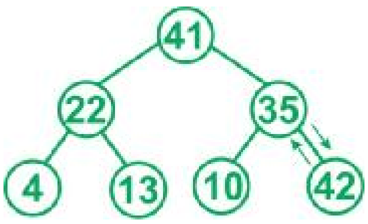
4º - Inserção do 04

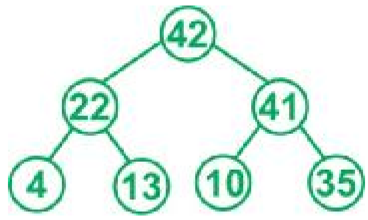
5º - Inserção do 22

6º - Inserção do 10

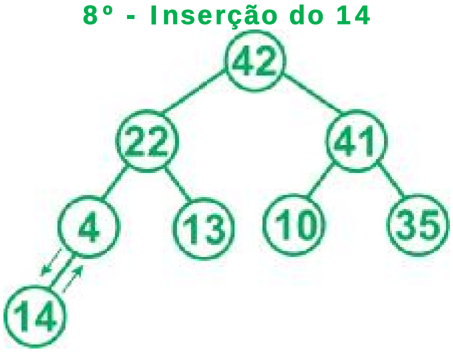


7º - Inserção do 42

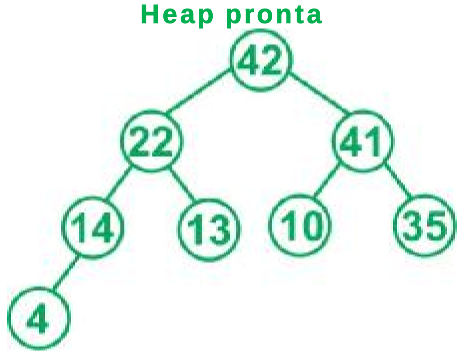




8º - Inserção do 14



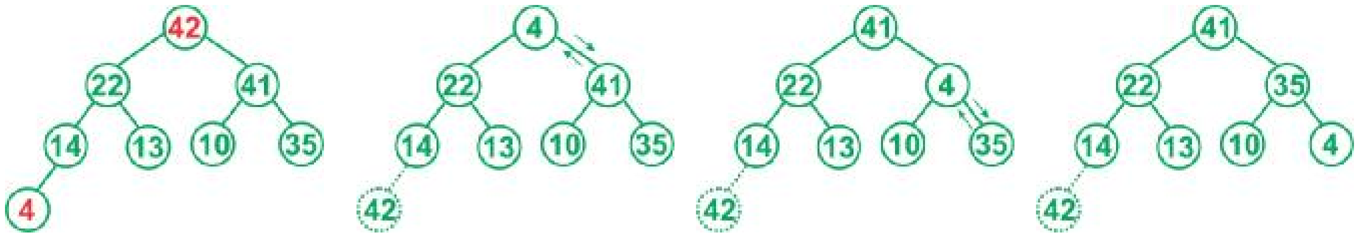
Heap pronta



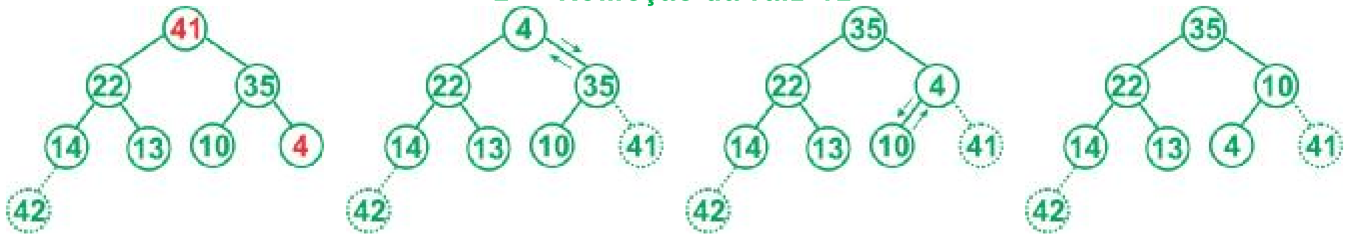
Partindo da heap pronta, executamos o algoritmo de ordenação por heap, que consiste em execuções sucessivas do algoritmo de remoção. Neste algoritmo, remove-se a raiz, colocando em seu lugar a folha mais à direita do último nível da árvore. Feito isso, sobre essa nova raiz deve-se executar a descida desse nó, de forma a manter a nova árvore na estrutura de heap.

Ordenação por heap:

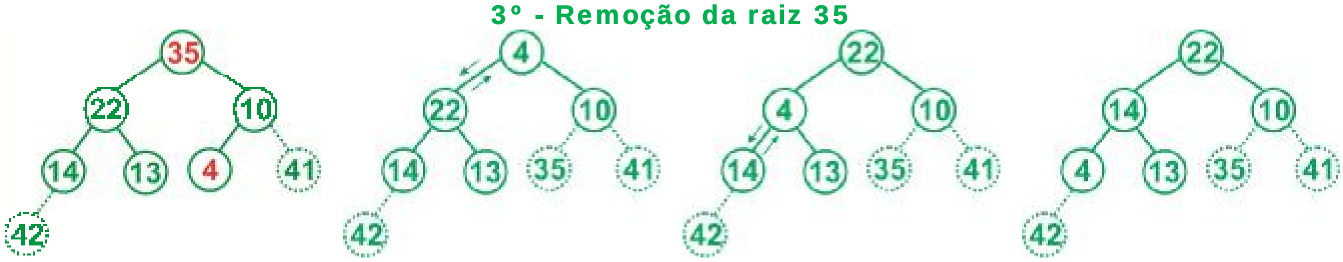
1º - Remoção da raiz 42



2º - Remoção da raiz 41



3º - Remoção da raiz 35



4º - Remoção da raiz 22

5º - Remoção da raiz 14

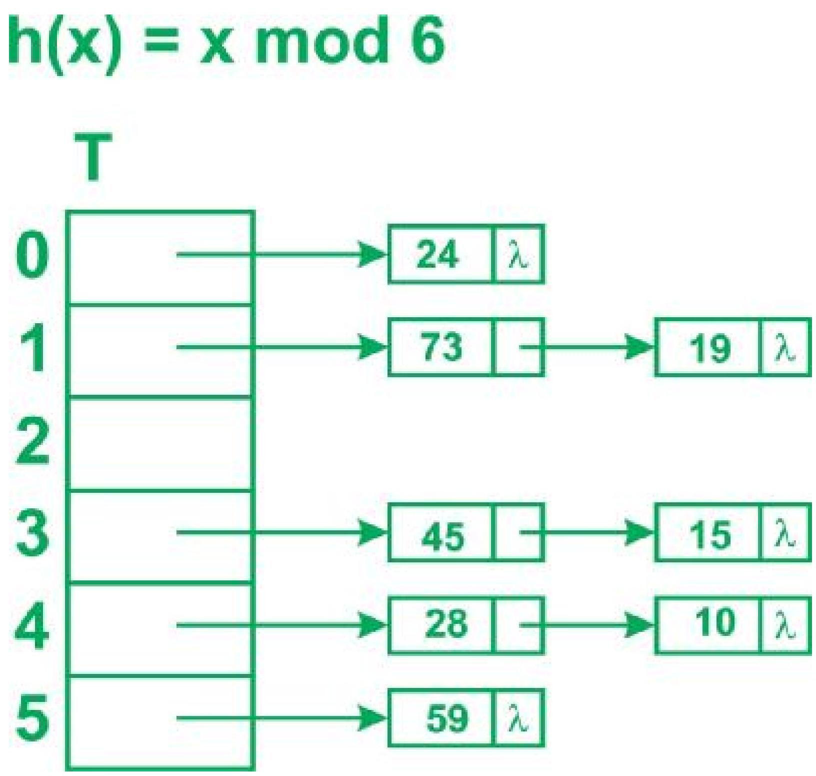
6º - Remoção da raiz 13

7º - Remoção da raiz 10

8º - Remoção da raiz 4 - heapsort finalizado.

6. (1,5) Seja T uma tabela de dispersão com 6 posições implementada por encadeamento exterior. A função de dispersão é h(x) = x mod 6. Desenhe a tabela após a inclusão das chaves 45, 73, 59, 28, 15, 24, 10, 19, nesta ordem.

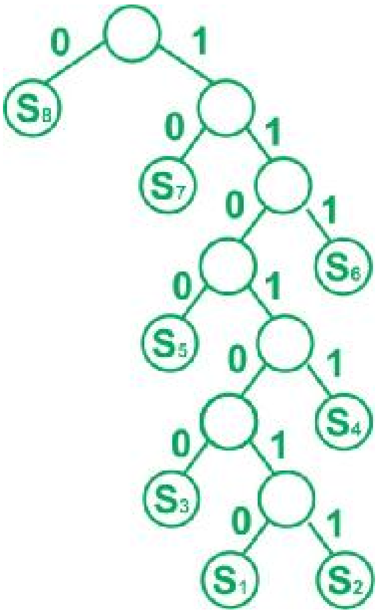
Resposta:



7. (1,5) Desenhe uma árvore de Huffman relativa às frequências 1, 1, 2, 3, 5, 8, 13, 21. A àrvore que você desenhou é a única possível?

Resposta: Sejam os códigos S_1, S_2, S_3, S_4, S_5, S_6, S_7 e S_8, correspondente às frequências 1, 1, 2, 3, 5, 8, 13, 21, respectivamente.

Assim, executando o algoritmo de Huffman, poderemos chegar a (uma das prováveis) árvore de Huffman, relativa a essas frequências e a esses símbolos:



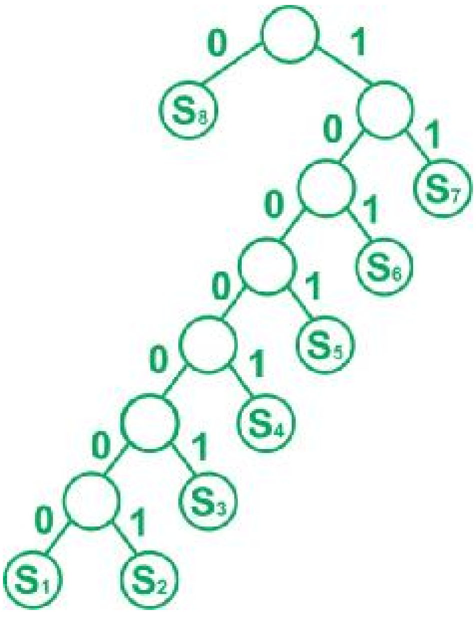
Onde os códigos relativos a cada símbolo são:
S8 = 0
S7 = 10
S6 = 111
S5 = 1100
S4 = 11011
S3 = 110100
S2 = 1101011
S1 = 1101010

O cálculo do custo C(T) dará:

$$C(T) = \sum_{i=1}^{8} f_i l_i$$

$$C(T) = 1x7 + 1x7 + 2x6 + 3x5 + 5x4 + 8x3 + 13x2 + 21x1 = 132$$

A árvore desenhada não é a única para a frequência dada, pois para cada iteração do algoritmo de Huffman, ao somar as duas árvores de menor frequência (chamemos de T' e T''), poderemos tomar a liberdade de colocar a Árvore T' do lado direito e T'' do lado esquerdo da árvore resultante da soma (ou vice-versa). Esse fato, associado ao número de iterações do algoritmo de Huffman, poderia gerar árvores diferentes da apresentada acima, mas que também fosse uma árvore de Huffman que representa a frequência dada. Para ilustrar, mostramos abaixo uma outra árvore, completamente diferente, que representa a mesma frequência (mas com códigos finais que podem ser diferentes para cada símbolo).



Onde os códigos relativos a cada símbolo são:
S8 = 0
S7 = 11
S6 = 101
S5 = 1001
S4 = 10001
S3 = 100001
S2 = 1000001
S1 = 1000000

O cálculo do custo C(T) será o mesmo, uma vez que cada código tem o mesmo comprimento do mesmo código na árvore anterior:

$$C(T) = \sum_{i=1}^{8} f_i l_i$$

$$C(T) = 1x7 + 1x7 + 2x6 + 3x5 + 5x4 + 8x3 + 13x2 + 21x1 = 132$$