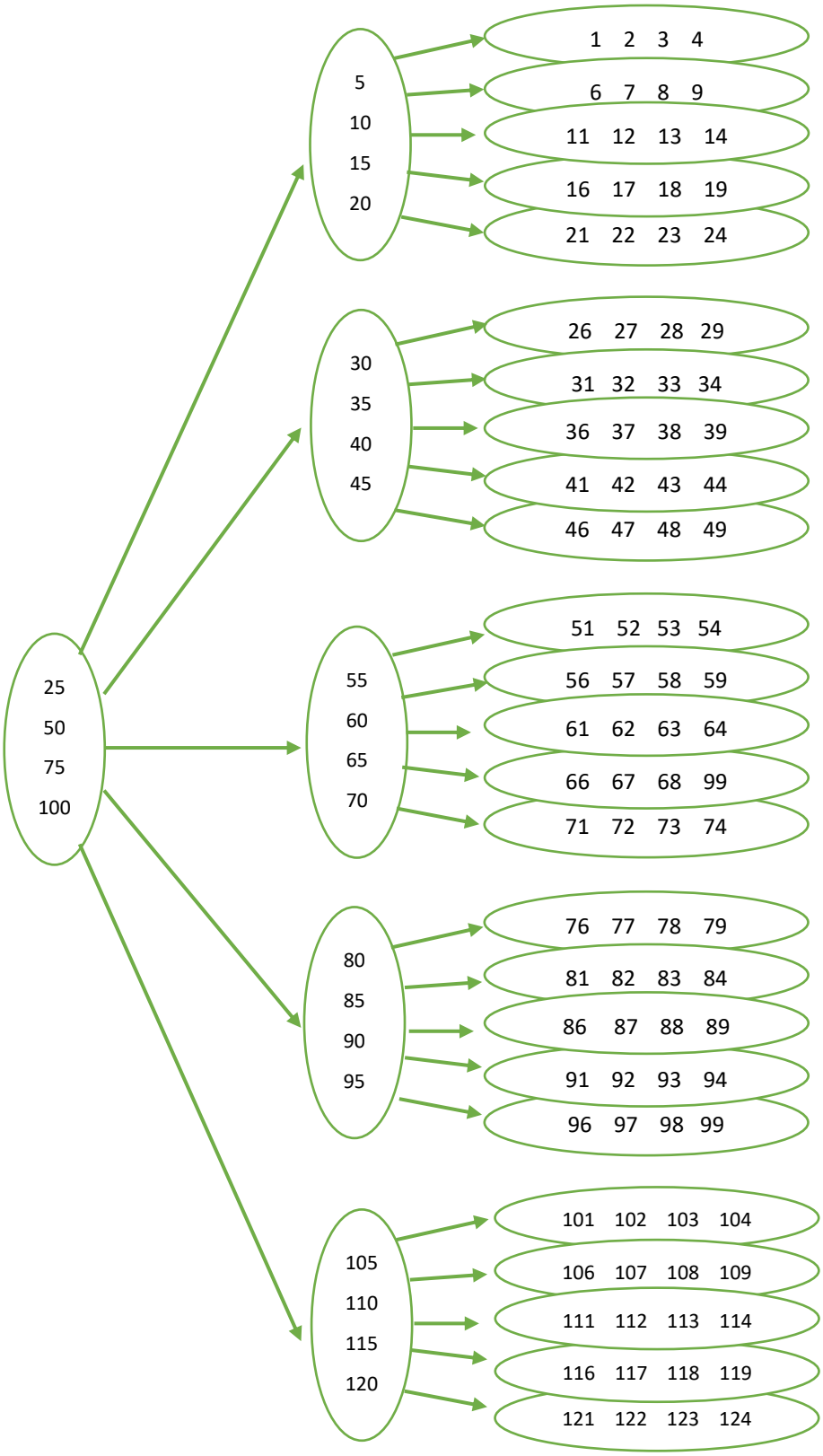
Gabarito Avaliação a Distância – Estrutura de Dados – 1º Semestre de 2019

Todas as questões valem 1.5

1. Responda os seguintes itens:

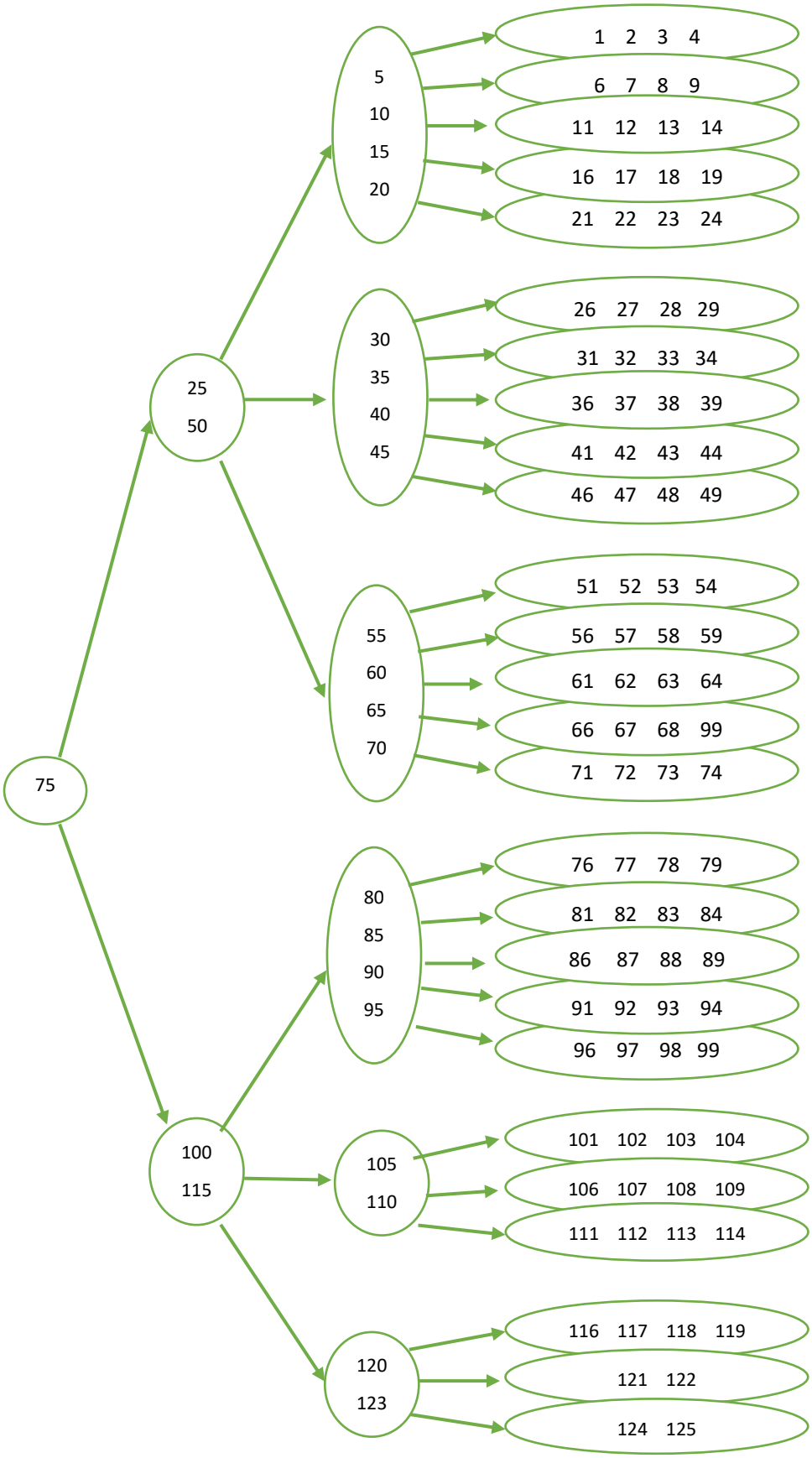
(a) Construa uma árvore B de ordem d=2 e altura h=3 que possua número máximo de chaves.

R: Segue abaixo uma proposta de árvore B de que atende aos requisitos do enunciado.



(b) Realize a inserção de uma nova chave (de valor qualquer) na árvore B do item anterior, e represente a árvore RESULTANTE após a inserção.

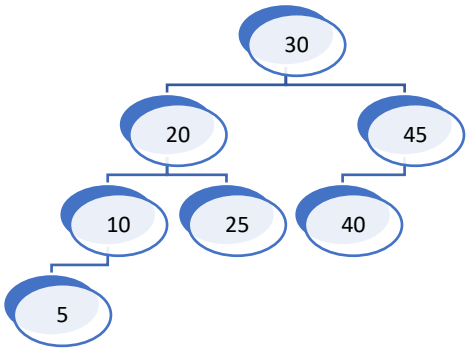
R: Segue abaixo a árvore resultante após a inserção da chave com valor 125.



2. Responda os seguintes itens:

(a) Construa uma árvore AVL de altura h=4 que possua um número mínimo de nós. Coloque o valor da chave dentro de cada nó.

R: Veja, a seguir, uma das árvores possíveis que atendem ao requisito do enunciado.

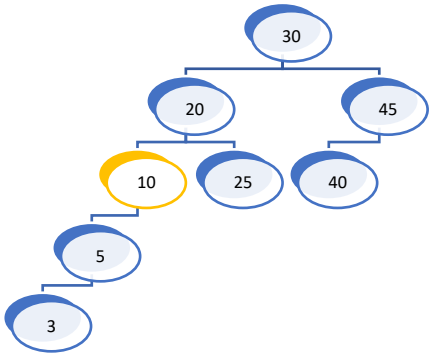


(b) Dê exemplo de uma inserção na árvore AVL do item anterior que exija uma rotação para tornar a árvore novamente regulada. Explique qual a rotação realizada, e por quê.

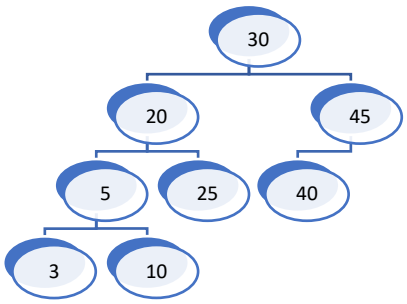
R: A árvore abaixo considera a inserção do nó com valor 3. Note que a inserção deste nó causa um desbalanceamento na árvore e o primeiro nó desregulado, subindo a partir do nó 3, é o nó 10. Neste caso, acontece então uma rotação a direita pois:

- O nó 10 está desregulado
- A altura da subárvore da esquerda do nó 10 é maior que a altura da subávore da direita.
- A altura da subárvore da esquerda do nó 5 (primeiro filho de 10 no caminho até o nó inserido) é maior que a altura da subávore da direita.

Árvore desregulada após inserção do Nó 3



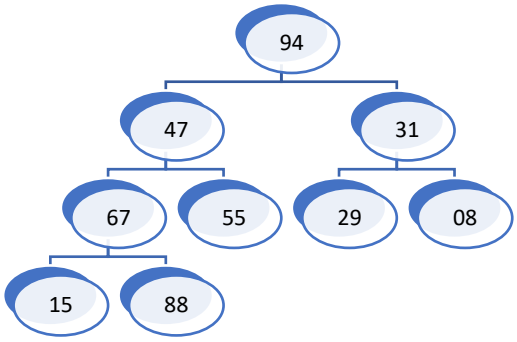
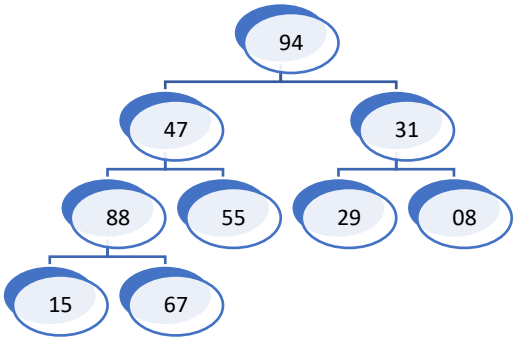
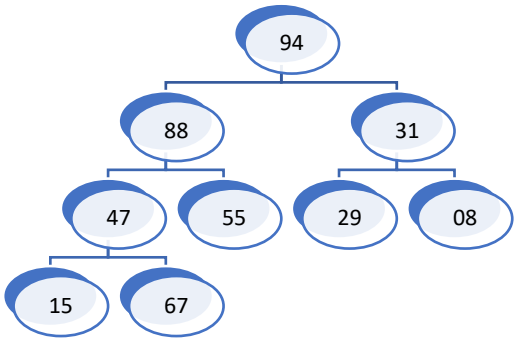
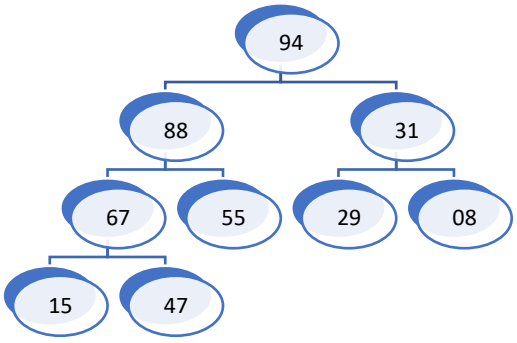
Árvore regulada após a inserção do Nó 3.



3. Responda os seguintes itens:

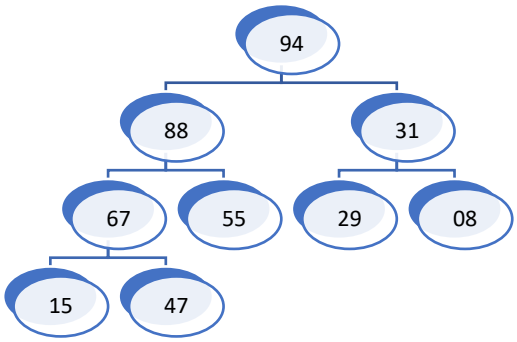
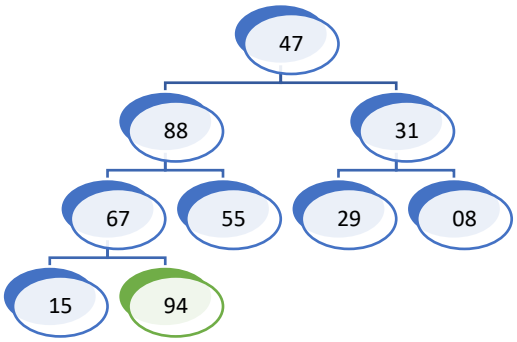
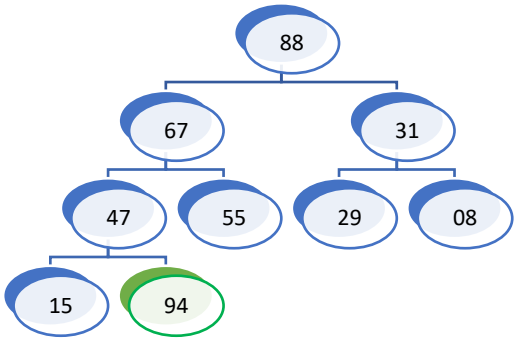
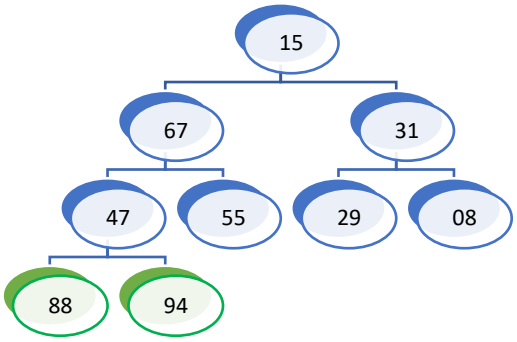
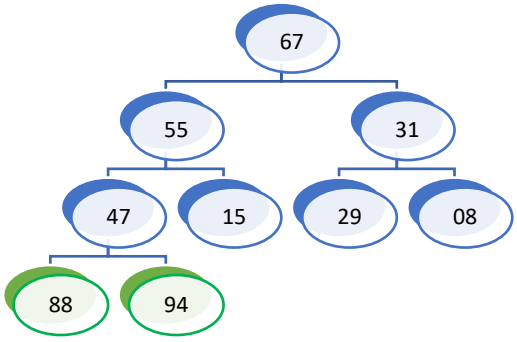
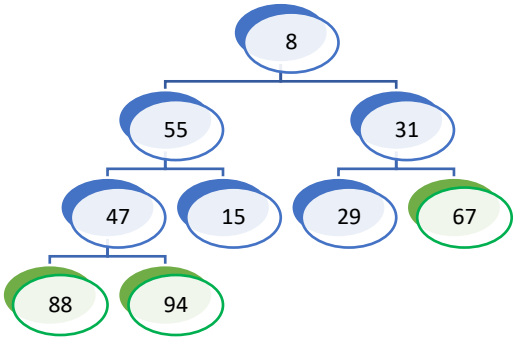
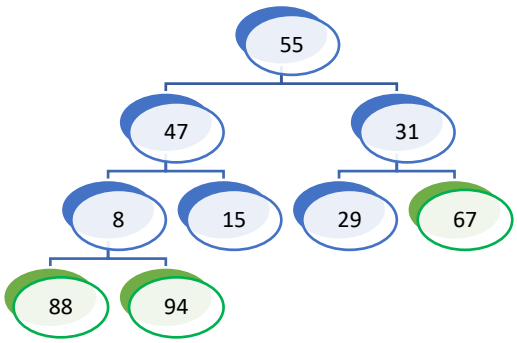
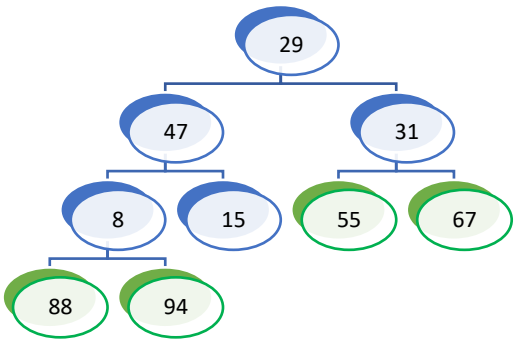
(a) Determine o heap obtido pela aplicação do algoritmo de construção (de tempo linear) às seguintes prioridades: 94, 47, 31, 67, 55, 29, 08, 15, 88. (Represente todas as trocas de elementos durante a execução do algoritmo.)

R: Segue abaixo a construção do heap em tempo linear para a lista de prioridades apresentada no exercício.

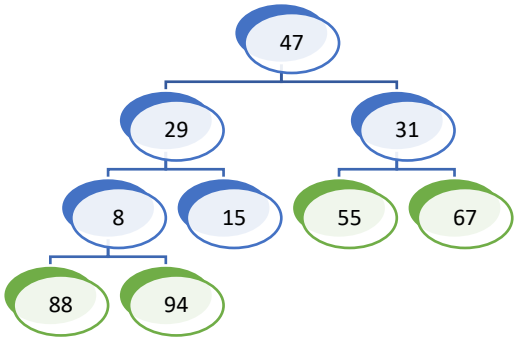
<table>
<tr>
<td>Condição Inicial
 
</td>
<td>Desce 67
 
</td>
</tr>
<tr>
<td>Desce 47 (troca 1)
 
</td>
<td>Desce 47 (troca 2)
 
</td>
</tr>
</table>

(b) A partir do heap construído no exercício anterior, execute o método de ordenação "heapsort", desenhando as configurações sucessivas da árvore durante o processo de ordenação.

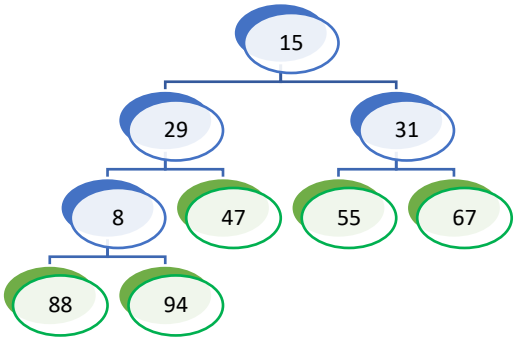
R: Segue abaixo as sucessivas trocas nos elementos da árvore durante todo o processo de ordenação do heapsort

<table>
<tr>
<td>Condição inicial:
 
</td>
<td>Troca 94 e 47
 
</td>
</tr>
<tr>
<td>Desce 47
 
</td>
<td>Troca 88 e 15
 
</td>
</tr>
<tr>
<td>Desce 15
 
</td>
<td>Troca 67 e 8
 
</td>
</tr>
<tr>
<td>Desce 8
 
</td>
<td>Troca 55 e 29
 
</td>
</tr>
</table>

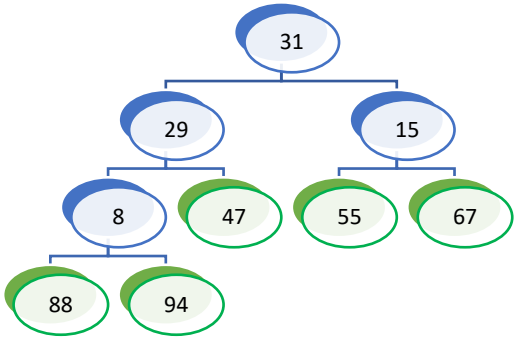
Desce 29



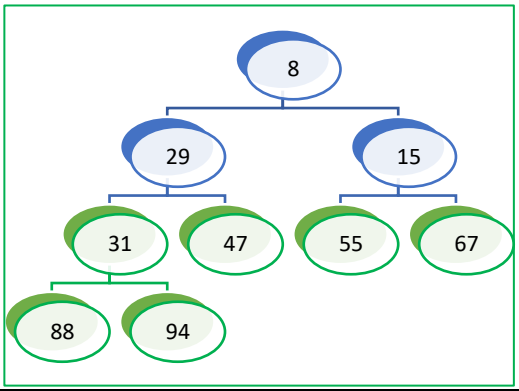
Troca 47 e 15



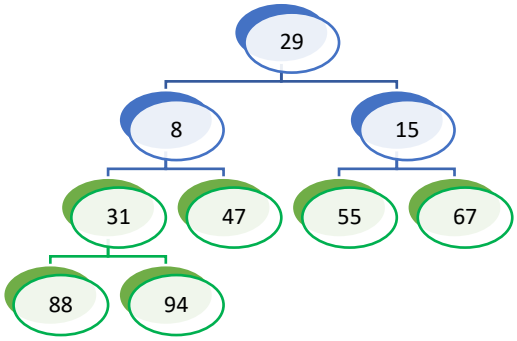
Desce 15



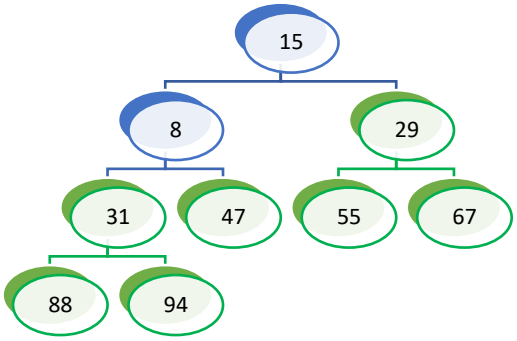
Troca 31 e 8



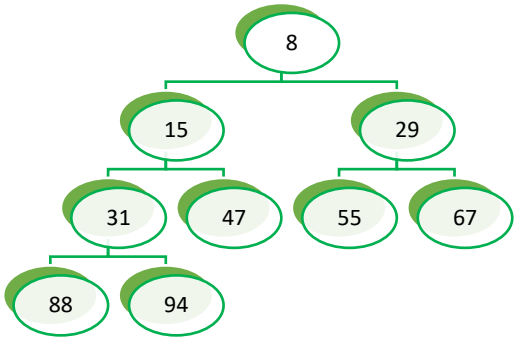
Desce 8



Troca 29 e 15



Troca 15 e 8 (condição final)



4. Responda as questões abaixo:

Suponha um conjunto S com 5 chaves s_1 , s_2 , s_3 , s_4 , s_5, que serão inseridas nesta ordem em uma tabela de dispersão T de tamanho 5, segundo uma função de dispersão h, onde o tratamento de colisões se realiza pelo método do encadeamento exterior. Determinar valores que as chaves devem possuir, e qual deve ser a função de dispersão h, para que T obedeça, respectivamente, às seguintes condições:

(a) O número de colisões é mínimo.

R: O número mínimo de colisões é zero. Neste sentido um conjunto de chaves e função de dispersão capazes de gerar zero colisões é apresentado a seguir.

Função de dispersão: h(x) = x mod 5
Chaves: 1, 2, 3, 4 e 5

Tabela	
Índice	Valor
0	5
1	1
2	2
3	3
4	4

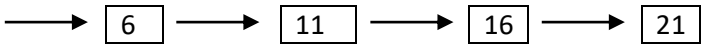
(b) O número de colisões é máximo.

R: Considerando um conjunto com N chaves, o número de colisões máximas será dado por N-1, onde N, neste caso, é igual 5. Portanto, o máximo de colisões neste exemplo é 5. A mesma função de dispersão do exercício anterior pode ser utilizada, porém, considerando um outro conjunto de chaves,

Função de dispersão: h(x) = x mod 5
Chaves: 1, 6, 11, 16 e 21

Tabela	
Índice	Valor
0	
1	1
2	
3	
4	

Encadeamento exterior



(c) O número de colisões é dois.

R: Já para se ter um número de duas colisões podemos usar o mesmo conjunto de chaves do exercício (a) porém, com outra função de dispersão. Veja:

Função de dispersão: h(x) = x mod 4
Chaves: 1, 2, 3, 4 e 5

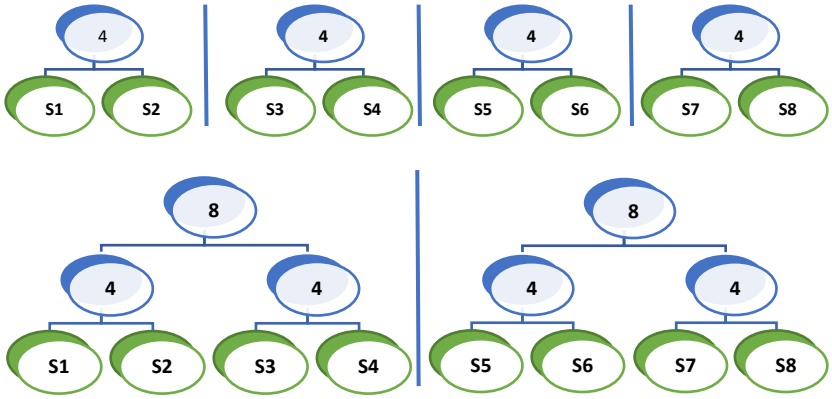
Tabela		***Encadeamento exterior***
Índice	Valor	
0	4	
1	1	→ 5
2	2	
3	3	
4		

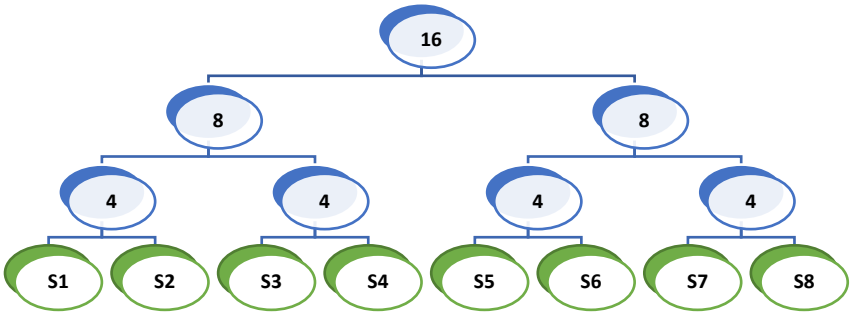
5. Responda as questões abaixo:

(a) Que tipo de árvore binária é uma árvore de Huffman relativa a n frequências iguais, quando n é uma potência de dois? Desenhe um exemplo.

R: Neste caso esta árvore de Huffman também será uma árvore cheia com altura k +1, onde $n=2^k$. Isso acontece, pois, o algoritmo irá unir as árvores, duas a duas, até que todas as árvores façam parte de uma árvore cheia. Como *n* é uma potência de 2, o algoritmo irá unir sempre árvores cheias, gerando como resultado uma outra árvore, também cheia. Acompanhe o exemplo(os nós sombreados em verde indicam os símbolos).

Símbolo	***Frequência***
S1	2
S2	2
S3	2
S4	2
S5	2
S6	2
S7	2
S8	2

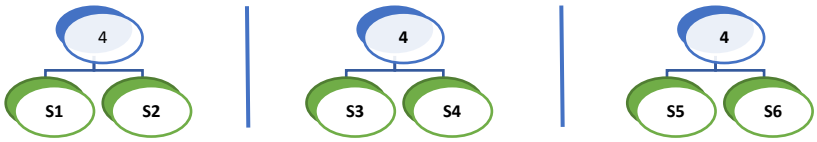


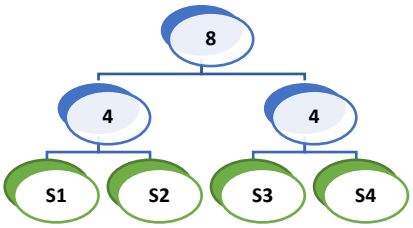


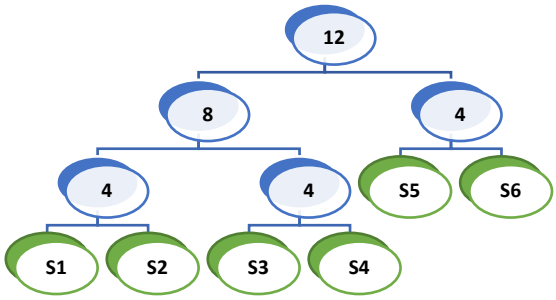
(b) Refaça a questão acima, considerando agora que n não é necessariamente uma potência de dois.

R: Neste caso a árvore não será cheia, mas será completa. Segue exemplo (os nós sombreados em verde indicam os símbolos)

Símbolo	*Frequência*
S1	2
S2	2
S3	2
S4	2
S5	2
S6	4







6. Descreva um algoritmo que, tendo com entrada uma árvore binária de busca e uma chave qualquer, determina a altura do nó que contém esta chave (caso ela pertença à árvore) ou imprime "chave não encontrada"

R: O algoritmo a seguir a apresenta um dos caminhos possíveis para este problema. A solução apresentada é dividida em duas partes. A primeira é a típica busca em uma árvore binária, que tem por objetivo encontrar um nó numa dada árvore através da *função busca(no, val)*. Uma vez encontrado o nó, a sua altura é calculada pela função *altura(no).* Segue.

```
1   função busca (no, val):
2       se (no == vazio):
3           retorne vazio
4
5           se (no->chave == val):
6           return no
7
8       se (no->chave > val):
9           return busca (no->esq, val)
10      senão:
11          return busca (no->dir, val)
12
13  função altura (no):
14     se (no == vazio)
15        retorne -1;
16     senão {
17        he = altura (no->esq)
18        hd = altura (no->dir)
19
20        se (he < hd): retorne hd + 1
21        senão: retorne he + 1
22
23
24  CHAMADA EXTERNA:
25  no = busca(arvore, 5)
26
27  se (no == vazio):
28          print("Chave não encontrada")
29  senão:
30          h = altura(no)
31          print("A altura do nó é", h)
```