Estrutura de Dados - 2°Período de 2015

Segunda Avaliação a Distância

1. (1,0) Prove ou dê um contraexemplo: Uma árvore binária pode ser construída, de forma única, a partir dos seus percursos em *pós-ordem* e *em nível*.

   Resposta: A afirmação é falsa. Como contraexemplo, considere duas árvores binárias T_1 e T_2, cada uma contendo os nós A e B, tal que:

   - em T_1, B é filho esquerdo de A;
   - em T_2, B é filho direito de A.

   Para ambas as árvores temos que os percursos *em pós-ordem* e *em nível* são B A e A B, respectivamente.

2. (1,0) Desenhe uma árvore binária de busca *cheia* com altura 4, colocando dentro de cada nó o valor de sua chave. As chaves são $1, 2, \ldots, k$ (k é o número de nós da árvore, que é um valor que você deve deduzir). A seguir, escreva a sequência de chaves que corresponde ao percurso em *pré-ordem* desta árvore.

   Resposta: Seja T_h o número de elementos de uma árvore binária cheia de altura h. Pela definição de árvore cheia, temos que $T_h = 2 \times T_{h-1} + 1$, onde $T_1 = 1$. Isso se deve pelo fato de ambas as subárvores de T também serem árvores cheias, mas de altura uma unidade menor, além do nó raiz. Resolvendo esta equação de recorrência (análoga a do problema das torres de Hanói), temos que $T_h = 2^h - 1$. Logo teremos uma árvore com 15 elementos.

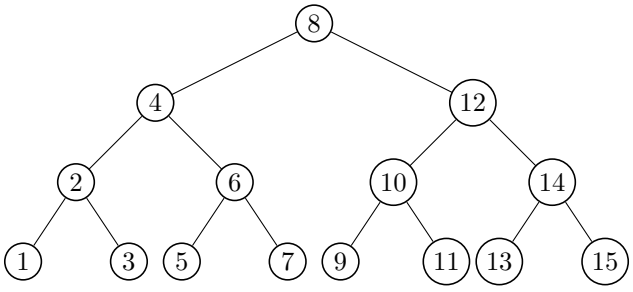


Figura 1: Árvore binária de busca cheia com altura 4.

   Percurso *em pré-ordem*: 8 4 2 1 3 6 5 7 12 10 9 11 14 13 15

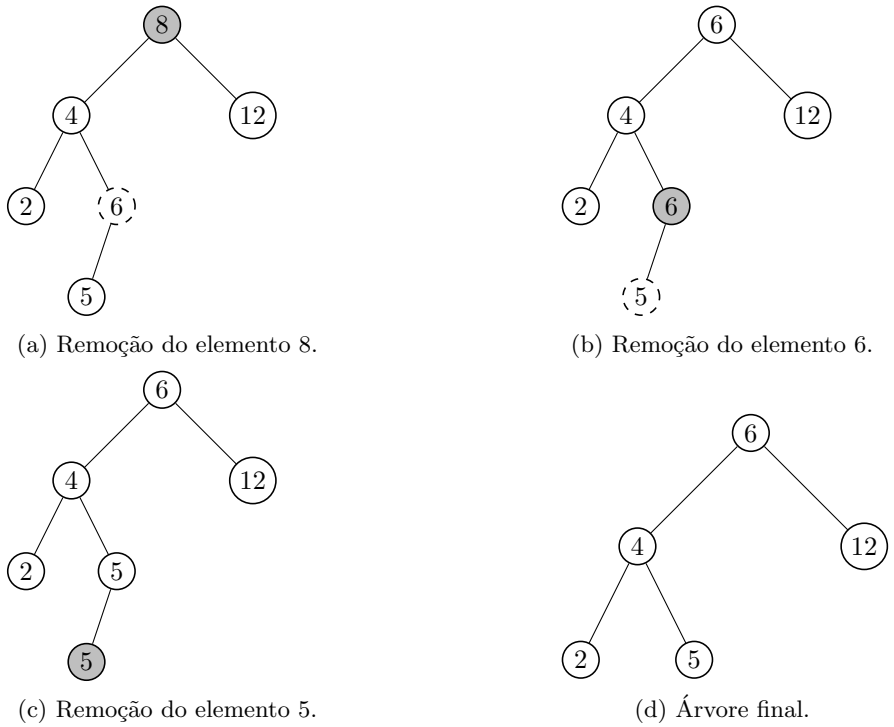
3. (1,0) Suponha que você deseja remover um nó qualquer de uma árvore binária de busca. Após removê-lo, como você deve reestruturar a árvore de modo que ela continue sendo uma árvore binária de busca? Dê um exemplo que mostre seu raciocínio.

   Resposta: Seja T uma árvore binária de busca com n elementos e subárvores esquerda e direita T^E e T^D, respectivamente. Além disso, seja $\ell_{\max}(T)$ e $r_{\min}(T)$ o maior elemento de T^E e o menor elemento de T^D, respectivamente.

Uma forma de remover um elemento x de T é substituí-lo por $\ell_{\max}(T_x)$, onde T_x é a subárvore de T enraizada em x. Se $T_x^E = \emptyset$, então substitui-se x por $r_{\min}(T_x)$. Se ambas T_x^E e T_x^D são vazias, então a remoção simples de x é suficiente. Este procedimento é aplicado recursivamente, onde a cada iteração removemos de T o elemento que substitui o elemento a ser removido no passo anterior.

A ideia deste algoritmo é substituir x por um elemento y que seja ao mesmo tempo maior que todos aqueles de $T_x^E \setminus \{y\}$ e menor que todos aqueles de $T_x^D \setminus \{y\}$. Essas propriedades são satisfeitas por $\ell_{\max}(T_x)$ e $r_{\min}(T_x)$. Pode-se determinar $\ell_{\max}(T_x)$ ao "descer" em T_x seguindo sempre a subárvore direita a partir do filho esquerdo de x, onde a busca continua enquanto houver filho direito. Analogamente, pode-se determinar $r_{\min}(T_x)$ ao "descer" em T_x seguindo sempre a subárvore esquerda a partir do filho direito de x, onde a busca continua enquanto houver filho esquerdo.

No exemplo abaixo, o nó cinza representa o valor x que deve ser removido em cada passo. Além disso, o nó pontilhado representa $\ell_{\max}(T_x)$ ou $r_{\min}(T_x)$.



(a) Remoção do elemento 8.

(b) Remoção do elemento 6.

(c) Remoção do elemento 5.

(d) Árvore final.

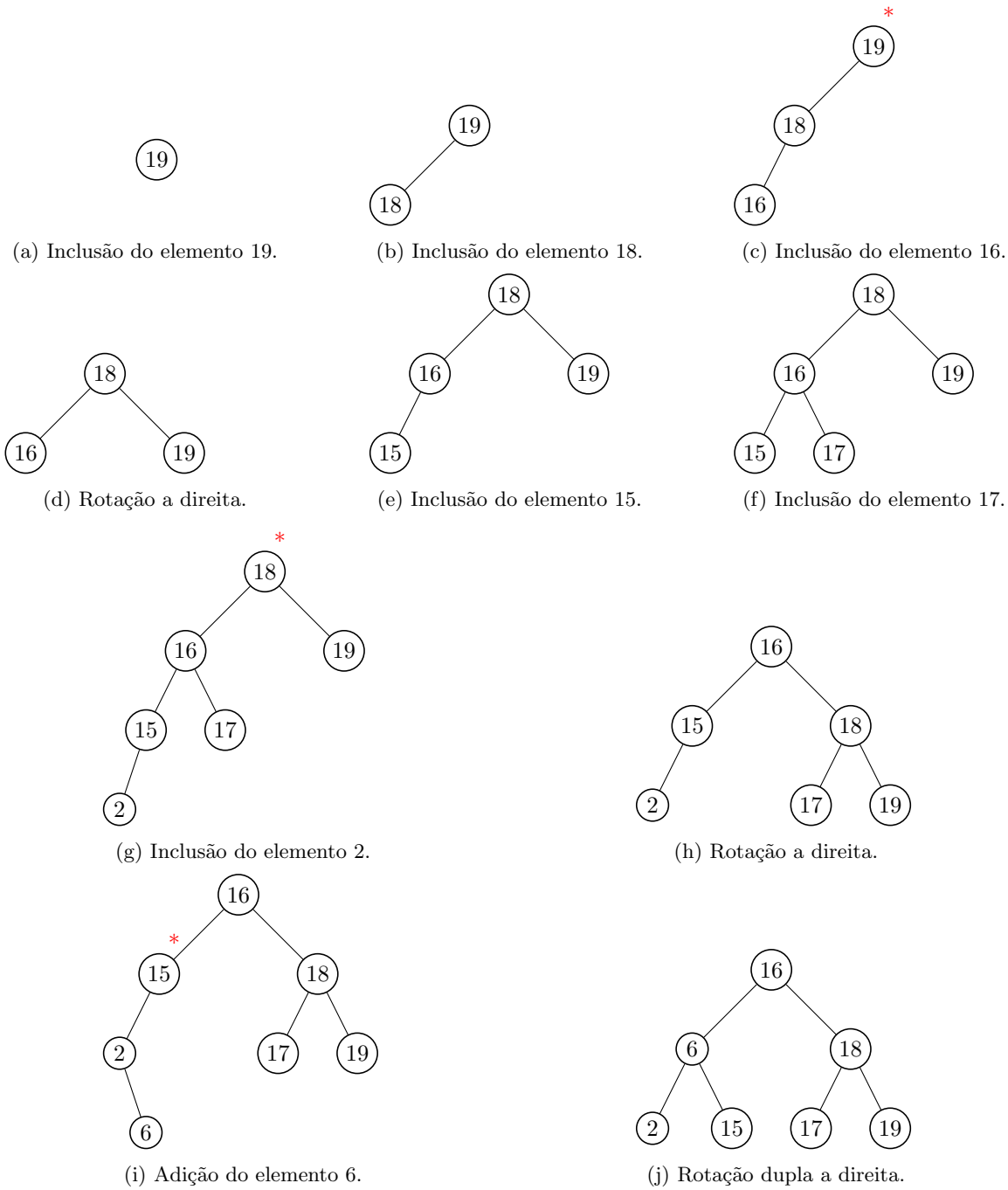
Figura 2: Exemplo de remoção do elemento 8.

4. (1,0) Desenhe a árvore AVL obtida pela sequência de inserções das chaves $19, 18, 16, 15, 17, 2, 6$, nessa ordem. Explique as operações realizadas, passo a passo.

   Resposta: A Figura 3 representa a sequência de inclusões e rotações necessárias para que a árvore obtida a cada etapa continue balanceada. O símbolo $(*)$ é usado sobre um nó para indicar que o mesmo tornou-se desbalanceado.

5. (1,0) Desenhe uma árvore B de ordem 3 que contenha as seguintes chaves: $10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65$.

   Resposta: A Figura 4 mostra algumas árvores intermediárias obtidas após a inclusão dos elementos em ordem crescente. Lembre-se de que numa árvore B de ordem d, cada página está ordenada, onde a página raiz deve conter entre 1 e $2d$ chaves e 2 a $2d+1$ filhos. Cada uma das demais páginas possui entre d e $2d$ elementos e $d+1$ a $2d+1$



(a) Inclusão do elemento 19.

(b) Inclusão do elemento 18.

(c) Inclusão do elemento 16.

(d) Rotação a direita.

(e) Inclusão do elemento 15.

(f) Inclusão do elemento 17.

(g) Inclusão do elemento 2.

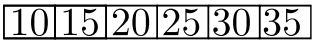
(h) Rotação a direita.

(i) Adição do elemento 6.

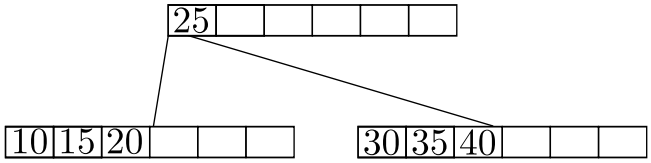
(j) Rotação dupla a direita.

Figura 3: Sequência de inclusões para construção de uma árvore AVL.

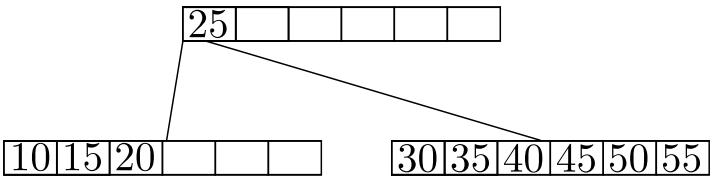
filhos. Além disso, um elemento na posição i de uma página é maior que todos os elementos da subárvore $i-1$ e de seu pai e é menor que todos os elementos da subárvore i. Finalmente, todas as folhas da árvore estão no mesmo nível.



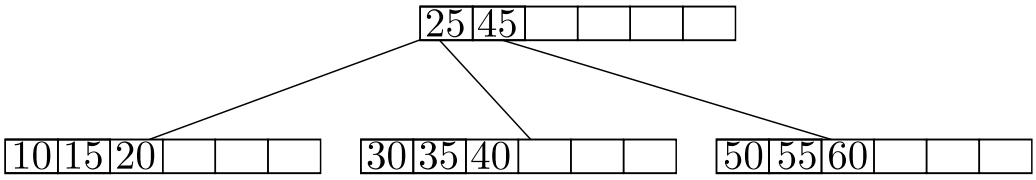
(a) Árvore obtida após a inclusão dos elementos 10, 15, 20, 25, 30 e 35.



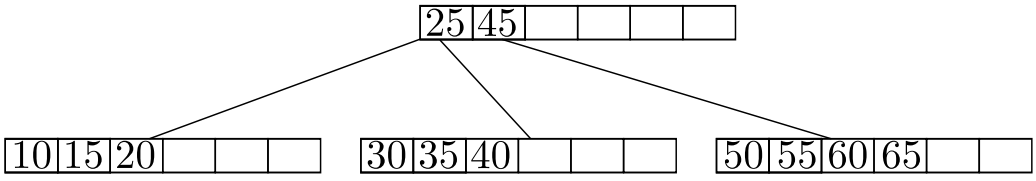
(b) Árvore obtida após a inclusão do elemento 40.



(c) Árvore obtida após a inclusão dos elementos 45, 50, 55.



(d) Árvore obtida após a inclusão do elemento 60.



(e) Árvore obtida após a inclusão do elemento 65.

Figura 4: Árvore B de ordem 3 contendo os elementos descritos na Questão 05.

6. (1,0) Desenhe uma árvore B de ordem $d = 2$ com três níveis e o maior número possível de chaves. (Os valores das chaves ficam à sua escolha.). Responda (V ou F): Se uma árvore B tem o maior número possível de chaves, então ela tem o maior número possível de páginas.

Resposta: A resposta é verdadeira, pois nesta árvore todas as páginas estão completas e qualquer inclusão promoveria a quebra de uma página, gerando um efeito em cadeia até a raiz, aumentando a altura da árvore em uma unidade. A Figura 5 representa uma tal árvore, onde as páginas folha estão dispostas não no mesmo nível devido ao tamanho da figura. Como sabemos, o número de páginas será igual a 31 e o número de elementos será 124.

7. (1,0) Construa um heap com as seguintes prioridades: $18, 25, 41, 60, 14, 10, 52, 50, 48$. A seguir, redesenhe o heap após a remoção do nó com prioridade 60.

   Resposta: A Figura 6 mostra uma representação em árvore de um possível heap obtido com os elementos dados, além das operações realizadas para a remoção do elemento 60. A cor laranja representa o elemento a ser removido, enquanto a cor azul representa que um elemento alcançou sua posição final no heap, ou seja, seus dois filhos são menores que ele.

8. (1,0) Determine o heap obtido pela aplicação do algoritmo de construção à seguinte lista de prioridades: $18, 25, 41, 34, 14, 10, 52, 50, 48$. Explique passo a passo.

   Resposta: A Figura 7 mostra a sequência de operações realizadas para construir o heap.

9. (1,0) Escreva um algoritmo que, dadas duas cadeias de caracteres X e Y, verifica se a cadeia X é prefixo ou sufixo ou ambas as coisas da cadeia Y. Exemplo: se $X = aba$ e $Y = abacataba$, então X é prefixo e sufixo de Y simultaneamente; ao passo que se $X = taba$, então neste caso X é apenas sufixo de Y.

   Resposta: O Algoritmo 1 mostra um exemplo de verificação se uma dada cadeia X é sufixo, prefixo, ambas ou nenhum em relação a outra cadeia Y.

Algoritmo 1: Algoritmo que verifica se uma dada cadeia X é sufixo, prefixo ou ambas de uma cadeia Y.

Entrada: Vetores de caracteres X e Y de tamanhos x e y, respectivamente.
Saída: Responde se X é sufixo, prefixo, ambos ou nenhum deles de Y.

```
1  p ← 1;
2  s ← 1;
3  i ← 1;
4  enquanto (p ≠ 0 ou s ≠ 0) e i ≤ x faça
5      se X[i] ≠ Y[i] então
6          p ← 0;
7      se X[i] ≠ Y[y − x + i] então
8          s ← 0;
9      i ← i + 1;
10 selecione (p, s) faça
11     caso (1, 1)
           Resultado: Ambos
12     caso (1, 0)
           Resultado: Prefixo
13     caso (0, 1)
           Resultado: Sufixo
14     Nenhum
```

10. (1,0) Descreva as árvores de Huffman para frequências de entrada que satisfazem a seguinte propriedade: $f_1 + f_2 < f_3, f_1 + f_2 + f_3 < f_4$, e assim por diante. De modo geral, temos:

$$f_1 + f_2 + \cdots + f_{j-1} < f_j, \text{ para todo } j = 3, 4, \ldots, n.$$

    Resposta: Como $f_1 + f_2 + \cdots + f_{j-1} < f_j$, temos que f_1 e f_2 são os elementos mínimos dentre todas as frequências, caso contrário $f_1 + f_2 \geq f_3$. Portanto, ambos f_1 e f_2 devem ser os primeiros elementos escolhidos para compor

uma nova subárvore, tal que sua raiz é um novo nó r_1 com valor $f_1 + f_2$ e cujos filhos são f_1 e f_2, de acordo com o algoritmo de Huffman. Pelo mesmo raciocínio, temos que $f_1 + f_2$ e f_3 são os novos menores elementos dentre os nós raízes resultantes. Assim é gerada uma nova subárvore com raiz de valor $f_1 + f_2 + f_3$ e filhos r_1 e f_3. A construção segue até que restem apenas duas subárvores cujas raízes valem $\sum_{i=1}^{n-1} f_i$ e f_n, gerando a raiz da árvore resultante com valor $\sum_{i=1}^{n} f_n$. A Figura 8 representa uma estrutura de tais árvores de Huffman (note que cada nó interno pode ter seus filhos permutados).

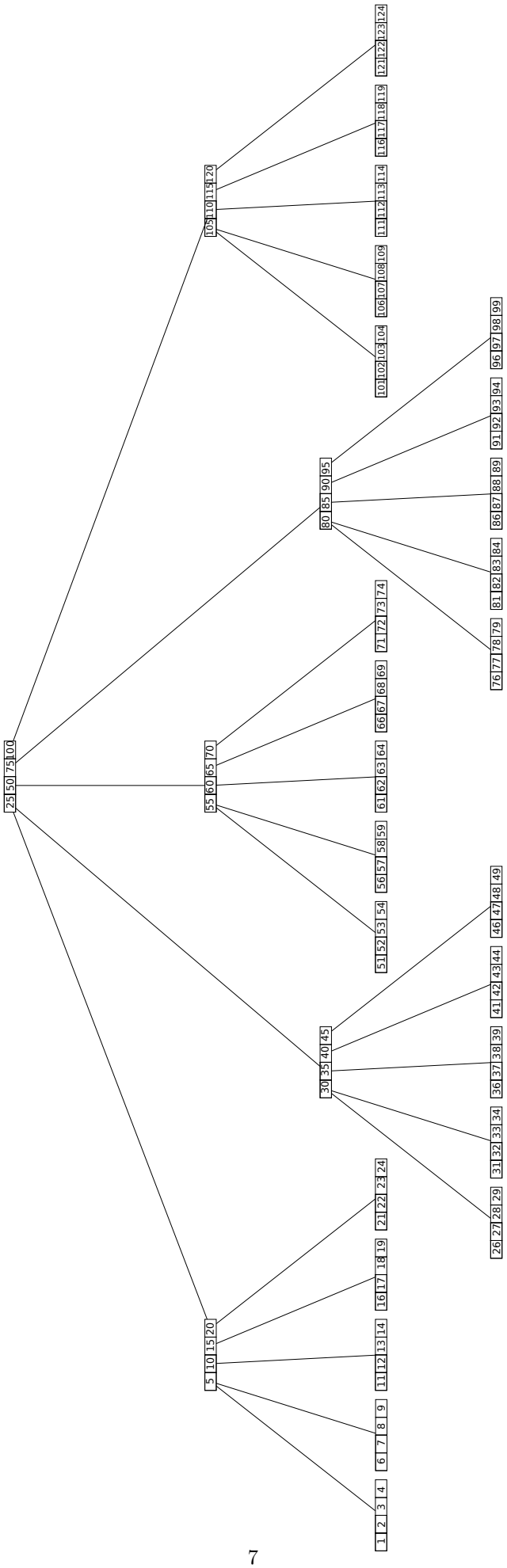
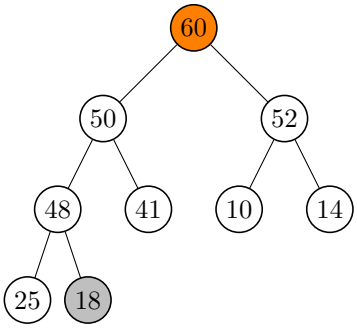
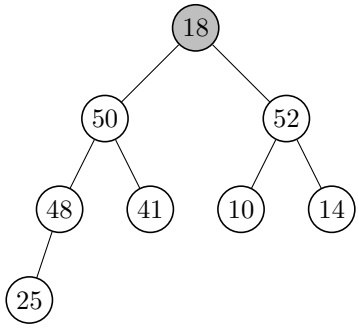


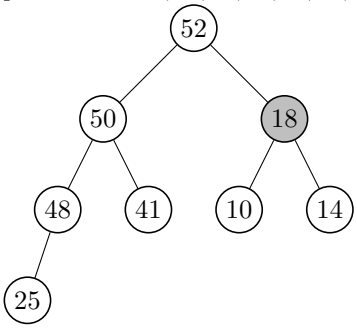
Figura 5: Representação de uma árvore B de ordem 2 e altura 3 com o número máximo de elementos.



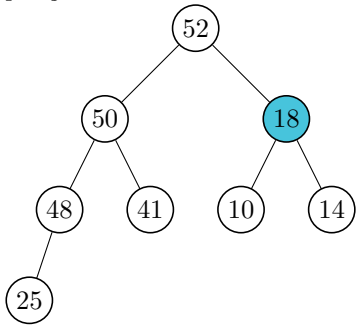
(a) Representação do Heap obtido a partir das prioridades: $18, 25, 41, 60, 14, 10, 52, 50, 48$.



(b) Copiamos o elemento da posição n para a posição 1.

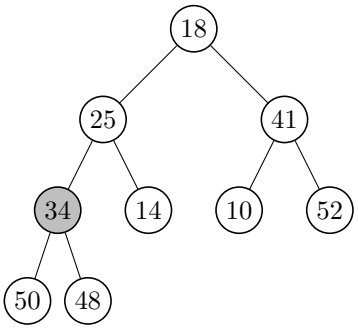


(c) Aplicamos o procedimento de descida do elemento 18.

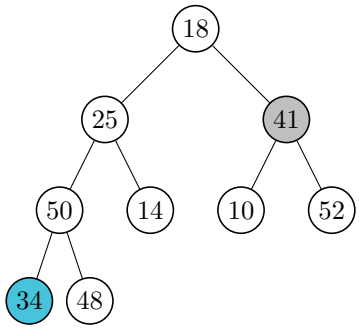


(d) Heap final.

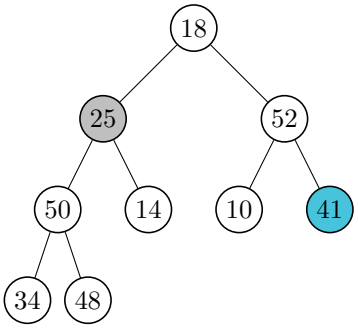
Figura 6: Operação de remoção do elemento 60.



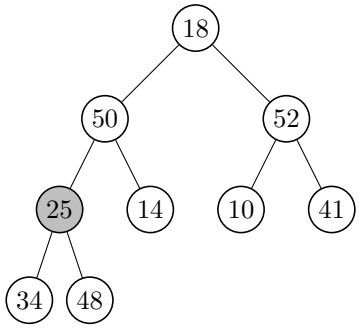
(a) Aplicamos o procedimento de descida ao elemento 34.



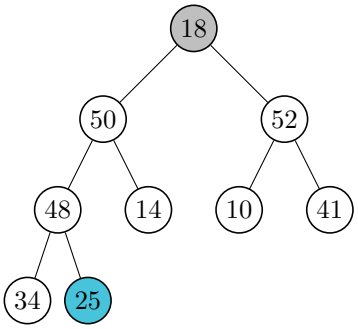
(b) Efetuamos a descida do elemento 41.



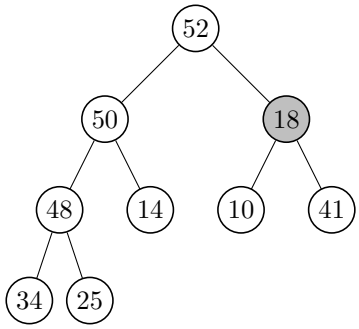
(c) Aplicamos o procedimento de descida do elemento 25.



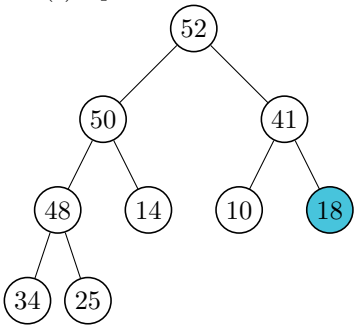
(d) Continuação da descida do elemento 25.



(e) Aplicamos a descida do elemento 18.



(f) Continuação da descida do elemento 18.



(g) Representação do heap final.

Figura 7: Operação de remoção do elemento 60.

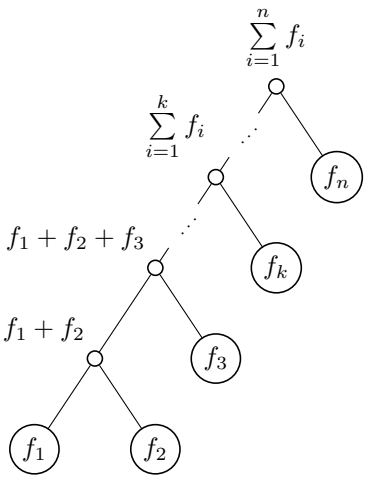


Figura 8: Representação do Heap obtido a partir das prioridades: $18, 25, 41, 60, 14, 10, 52, 50, 48$.