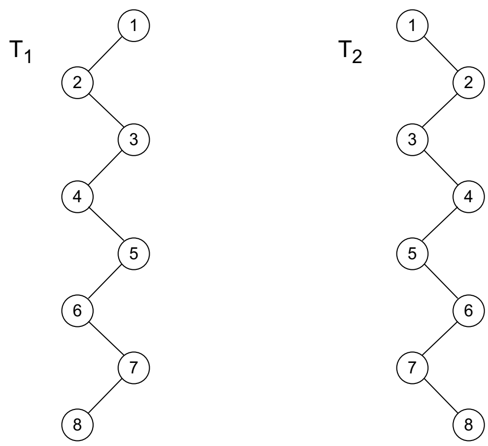
Estrutura de Dados - 2o. período de 2008

Gabarito da Segunda Avaliação à Distância

1. (1,5) Seja T uma árvore binária do tipo zigue-zague, com 8 nós, com rótulos respectivamente $1, \ldots, 8$, onde o nó $i > 1$ é filho do nó $i - 1$. Além disso, se i é filho esquerdo (direito) de $i - 1$ então $i + 1$ será filho direito (esquerdo) de i, para $i < 8$. Pede-se:

   (i) A árvore T é única ? Desenhar T e, caso não seja única, desenhar também todas as árvores binárias que satisfazem às condições acima.

   Resposta: Não. Temos duas árvores binárias que satisfazem às condições acima:



   (ii) Escrever os percursos pré-ordem, ordem simétrica, pós-ordem e percurso em nível, para cada possível árvore T encontrada no item (i).

   Resposta: Para T_1, temos os seguintes percursos:

   - pré-ordem: 1 2 3 4 5 6 7 8;
   - ordem simétrica: 2 4 6 8 7 5 3 1;
   - pós-ordem: 8 7 6 5 4 3 2 1;
   - em nível: 1 2 3 4 5 6 7 8.

   Para T_2, temos os seguintes percursos:

   - pré-ordem: 1 2 3 4 5 6 7 8;
   - ordem simétrica: 1 3 5 7 8 6 4 2;
   - pós-ordem: 8 7 6 5 4 3 2 1;
   - em nível: 1 2 3 4 5 6 7 8.

2. (2,0) Seja o conjunto S formado pelas 5 chaves s_i, $i = 1, \ldots, 5$. Desenhar a árvore de busca ótima para S, sabendo que a freqüência de acesso da chave i é igual a i. Por outro lado, são também realizadas tentativas de acesso com valores x de chaves inexistentes, onde $x \in (i, i+1)$, $1 \leq i \leq 4$. A freqüência desses acessos a chaves inválidas é igual a 1, para cada intervalo $(i, i+1)$. Determinar também o valor do custo da árvore ótima.

   Resposta: Para o conjunto de freqüências abaixo,

i	0	1	2	3	4	5
f_i	-	1	2	3	4	5
f'_i	1	1	1	1	1	1

   as matrizes do algoritmo de cálculo da árvore ótima são:

   Matriz dos valores $F[i, j]$:

1	3	6	10	15	21
-	1	4	8	13	19
-	-	1	5	10	16
-	-	-	1	6	12
-	-	-	-	1	7
-	-	-	-	-	1

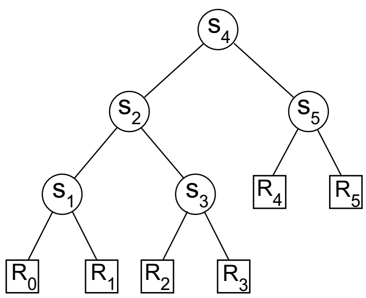
   Matriz dos custos $c[i, j]$:

0	3	9	18	30	46
-	0	4	12	23	38
-	-	0	5	15	28
-	-	-	0	6	18
-	-	-	-	0	7
-	-	-	-	-	0

   Matriz dos valores minimizantes k:

-	1	2	2	3	4
-	-	2	3	3	4
-	-	-	3	4	4
-	-	-	-	4	5
-	-	-	-	-	5
-	-	-	-	-	-

   Da última matriz acima, temos a seguinte árvore ótima, de custo 46:



3. (1,5) Explicar o conceito de balanceamento em árvores binárias. Em seguida, responder, com as devidas justificativas, às seguintes perguntas:

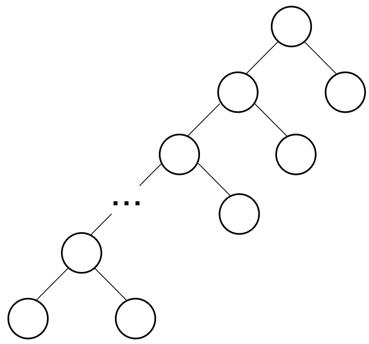
   Resposta: Uma árvore binária T é balanceada se cada subárvore de T com m nós possui altura igual a $O(\log m)$.

   (i) Uma árvore completa é sempre balanceada ?

   Resposta: Sim. Claramente, uma árvore cheia é sempre balanceada. Como uma árvore completa T é cheia até o penúltimo nível, temos que, para qualquer subárvore de T com m nós, sua altura será no máximo $O(\log m) + 1 = O(\log m)$.

   (ii) Uma árvore estritamente binária é sempre balanceada ?

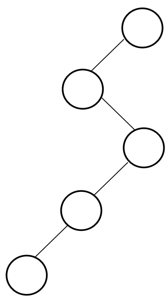
   Resposta: Não. Como exemplo, considere a árvore estritamente binária T, tal que, para cada nó interno de T, seu filho direito é uma folha.

   Sendo n o total de nós de T, sua altura é $\left\lceil \frac{n}{2} \right\rceil = O(n)$. Logo, a altura de T não é $O(\log n)$, e T não é balanceada.

   (iii) Uma árvore zigue-zague é sempre balanceada ?

   Resposta: Não. Para qualquer árvore zigue-zague T com n nós, temos que a altura de T é n. Logo, T não é balanceada.

   As respostas devem ser tais que se a resposta for SIM, deve ser apresentado um argumento que a justifique, enquanto que se a resposta for NÃO, deve ser descrito um exemplo onde a afirmativa correspondente falha.

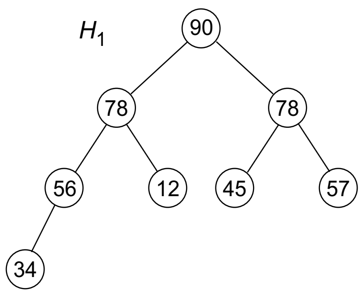
4. (1,5) Examinando as definições de árvore binária de busca e árvore B, descreva uma aplicação onde a primeira seria uma estrutura mais adequadada que a árvore B, e descreva outra aplicação onde a árvore B deveria ser a preferida.

   Resposta: Árvores binárias de busca são mais adequadas para armazenamento de tabelas pequenas, que podem ser inteiramente mantidas na principal. Já árvores B são mais adequadas para armazenamento de tabelas muito grandes, que não podem ser armazenadas na memória principal de uma só vez, tornando-se necessário armazenar parte da tabela em disco. Em razão disso, um tempo significativo é gasto para acessar um nó da tabela. Como as árvores B mantêm mais de uma chave em cada nó, as operações de busca, inserção e remoção são executadas rapidamente.

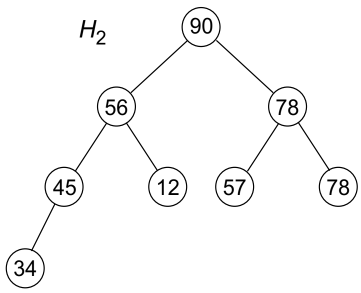
5. (1,5) Construir um heap H cujos nós possuem as prioridades $45, 34, 78, 90, 12, 57, 78, 56$. Em seguida desenhar o heap obtido a partir de H, efetuando-se, sucessivamente, as seguintes operações:

   Resposta:

   Utilizando a Solução 2 (slide 29.25), obtemos o seguinte heap H_1:

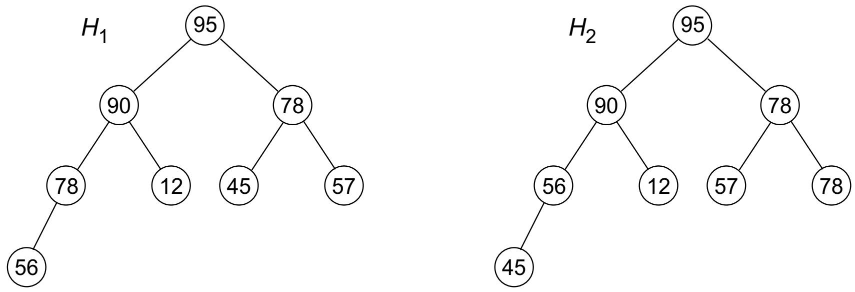


   Utilizando a Solução 3 (slide 29.31), obtemos o seguinte heap H_2:



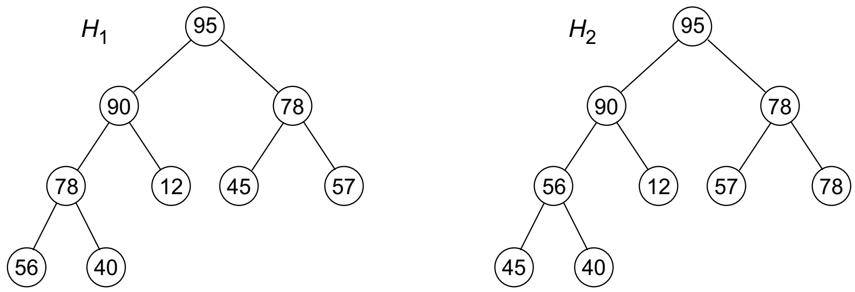
(i) Alterar a prioridade do nó 34 para 95.

Resposta: Dependendo do heap construído (H_1 ou H_2), temos as seguintes árvores, obtidas após a alteração da prioridade:



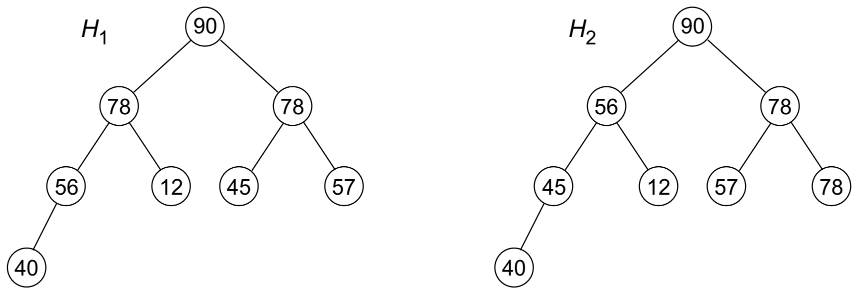
(ii) Incluir um novo nó de prioridade 40.

Resposta:



(iii) Remover o nó raiz.

Resposta:



6. (2,0) Para um certo conjunto de 8 símbolos, em relação aos quais se deseja construir uma árvore de Huffman H, escolher as frequências correspondentes a esses símbolos, de tal forma que H possua altura máxima. Justificar a escolha, em termos das operações efetuadas pelo algoritmo de Huffman. Desenhar a árvore obtida.

Resposta: Sejam $s_1, s_2, \cdots, s_8$ os símbolos e $f_1, f_2, \cdots, f_8$ suas respectivas freqüências. A partir do conjunto de freqüências $f_1 = 1,\ f_i = 2f_{i-1},\ 2 \leq i \leq 8$, por exemplo, obtemos uma árvore de Huffman H com altura máxima. A cada passo j da contrução de H, temos uma nova árvore, obtida da união da árvore formada pelos símbolos $\{s_1, \cdots, s_j\}$ com o símbolo s_{j+1} com. Assim, no primeiro passo do algoritmo, são unidos s_1 e s_2 em uma única árvore; no segundo passo, $\{s_1, s_2\}$ e s_3 são unidas, e assim por diante. Ao final, temos a seguinte árvore de Huffman: