Universidade Federal do Ceará
Centro de Ciências
Mestrado e Doutorado em Ciências da Computação – MDCC
Estruturas de Dados Avançadas

# Lista de Exercícios: Árvores B/B+

Data da Entrega: 25/05/2017
Exercício individual
Entrega da solução dos exercícios pelo SIGAA

1. Seja uma árvore B de ordem d = 3 e altura h = 4. Qual o número máximo de chaves na árvore? E o número mínimo? Justifique.

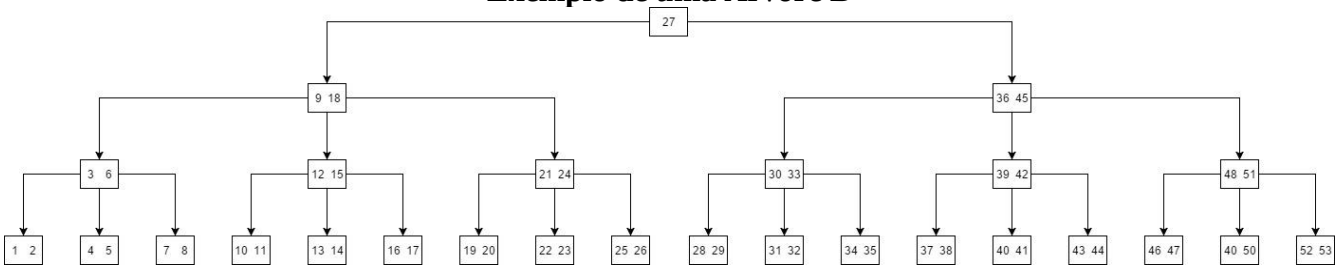
**Máximo:**
**Sendo altura igual a 4 então a árvore tem 5 níveis. Sabendo que cada nó (página) na árvore tem no máximo d-1 chave, ou seja, 2 chaves e no máximo d filhos, ou seja, 3 filhos temos que o primeiro nível, a raiz, terá 2 chaves. No segundo nível, teremos 2*d = 2*3 = 6 chaves. No terceiro nível temos 2*d*d = 18, no quarto nível temos 2*d*d*d = 54, e no último nível teremos 2*d*d*d*d = 162. Logo o número máximo de chaves dessa árvore será 2 + 6 + 18 +54 + 162 = 242.**

**Mínimo:**
**Sendo altura igual a 4 então a árvore tem 5 níveis. No primeiro nível, a raiz, teremos no mínimo 1 chave. Como a raiz tem que ter no mínimo 2 filhos e cada nó interno e folha tem no mínimo [d/2] -1 chaves, então no segundo nível teremos no mínimo 2 * ([d/2] -1) = 2*([3/2]-1) = 2*(2-1) = 2 chaves. Sabendo que cada nó interno tem no mínimo [d/2] filhos, então no terceiro nível teremos 2 * [d/2] * ([d/2] -1) = 2 * 2 * (2-1) = 4 chaves, no quarto nível 2 * [d/2] * [d/2] * ([d/2] -1) = 2 * 2 * 2 *(2-1) = 8 chaves e no quinto nível teremos 2 * [d/2] * [d/2] * [d/2] * ([d/2] -1) = 2 * 2 * 2 * 2 * (2-1) = 16 chaves. Logo teremos 1 + 2 + 4 + 8 + 16 = 31 chaves no mínimo.**

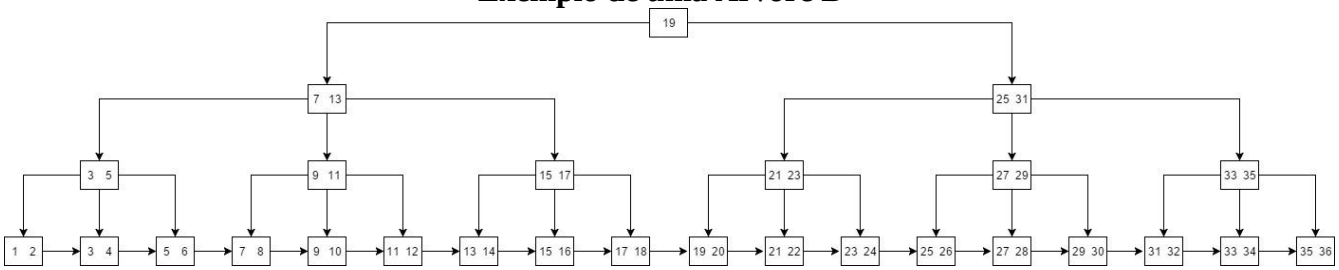
2. Descreva a diferença entre as árvores B e B+. Exemplifique cada uma das estruturas com ordem 2 e altura 3. Por que a árvore B+ é normalmente preferida como estrutura de acesso a arquivos de dados? **(Vou considerar ordem 5, fazendo de acordo com a métrica estudada em sala.)**

**As principais diferenças entre as árvores B e B+ é que na B+ todas as chaves são mantidas em folhas, sendo que algumas chaves são repetidas em nós não folhas formando um índice. E as folhas são ligadas, oferecendo um caminho sequencial para percorrer as chaves. Com essas diferenças, as árvores B+ aumenta a eficiência da localização do próximo registro na árvore de O(lgn) para O(1) e continua mantendo a eficiência da busca e da inserção das árvores B. Por isso elas são preferidas como estrutura de acesso a arquivos de dados.**

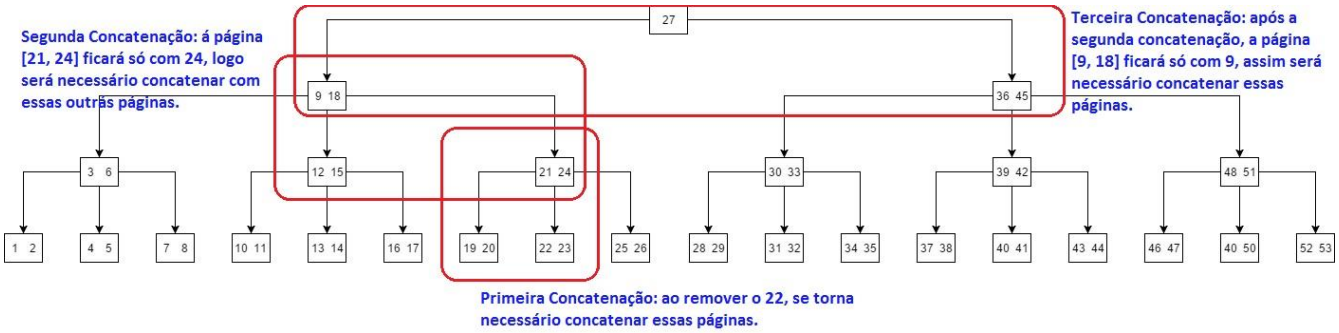
**Exemplo de uma Árvore B**



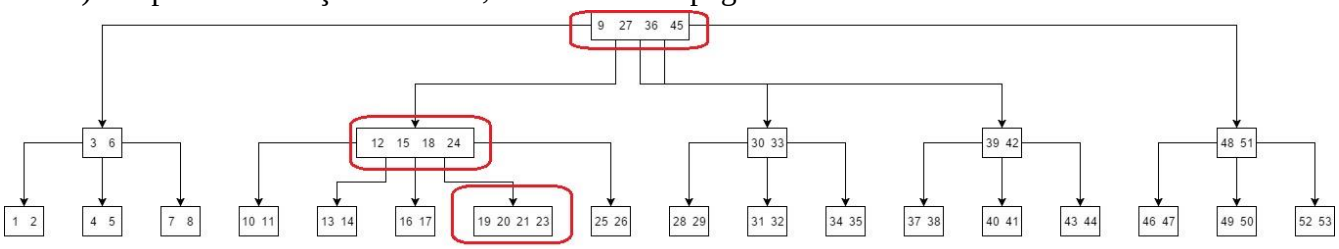
**Exemplo de uma Árvore B+**




3. Dê um exemplo de árvore B de ordem d=2 e altura h=3, de forma que a remoção de uma chave implique no maior número possível de páginas escritas. Mostre o seu exemplo: **(Vou considerar ordem 5, fazendo de acordo com a métrica estudada em sala.)**
   a) Antes da remoção da chave, assinalando as páginas a serem alteradas.
   **A chave escolhida para ser removida é a 22.**




   b) Depois da remoção da chave, assinalando as páginas escritas.




4. Responda o que se segue:
   a) A que condições deve satisfazer uma árvore B de ordem d para que a inserção de qualquer chave ocasione o aumento da altura da árvore?
   **Todas as páginas, exceto as folhas, devem ter d filhos e todas as páginas devem conter d-1 chaves.**

   b) A que condições deve satisfazer uma árvore B de ordem d para que a remoção de qualquer chave ocasione a redução da altura da árvore?
   **A raiz tem que ter apenas 1 chave e 2 filhos. Todas as páginas internas devem ter [d/2]-1 chaves e [d/2] filhos. E todas as páginas folhas devem ter [d/2] -1 chaves.**

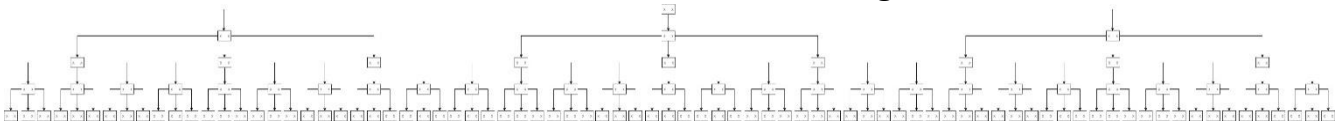
   c) Por que a redistribuição deve ser tentada antes da concatenação durante a remoção de uma chave situada em um nó com ocupação mínima de uma árvore B?
   **Porque na redistribuição não há propagação, já que o número de chaves no pai do nó não muda. Logo a redistribuição é menos custosa que a concatenação.**

d) Se uma chave não está situada em uma folha de uma árvore B, o que garante que sua sucessora imediata, se existir, estará obrigatoriamente localizada em uma folha?
**Se a chave não está situada em um nó folha, então o nó onde ela se encontra ramificará a árvore em vários ramos de acordo com o número de chaves do nó. Cada chave deste nó interno forma uma subárvore, onde a chave de maior valor na sua ramificação à esquerda, consequentemente seu predecessor, certamente estará em uma folha e a chave de menor valor na sua ramificação à direita, que será o seu sucessor, também deve estar em uma folha.**

e) Qual é o pior caso do algoritmo de inserção de uma chave em uma árvore B de ordem d? Como deve ser a árvore?
**O pior caso é quando é necessário aumentar a altura da árvore. Pois, primeiro ele localiza o elemento na folha em que deve ser inserido a chave e a insere. Se esta folha estiver cheia, será necessário realizar uma subdivisão de nós, em que passa o elemento mediano da folha para o pai e subdivide a folha em duas novas folhas com [d/2] -1 chaves para uma e d – [d/2] chaves para a outra. Contudo, se o pai estiver cheio, repete a subdivisão para esse nó pai, e continua assim até chegar na raiz. Se a raiz estiver cheia então será necessário aumentar a altura da árvore.**

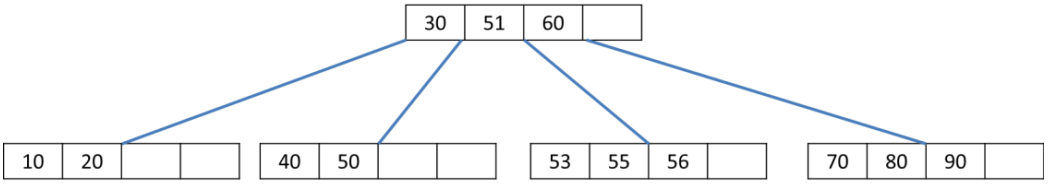
5. Responda as perguntas abaixo sobre árvores B:
   a) As operações de busca e inserção em uma árvore B tem a mesma complexidade?
   **Sim tanto para busca quanto para inserção uma árvore B requer O($\log_{\lceil d/2 \rceil}(n)$), sendo d a ordem da árvore. Como as operações de Split na inserção ocorre em tempo constante, então não interfere na complexidade da mesma.**

   b) Mostre uma árvore B de ordem 3 e altura 4 onde a inclusão de um novo registro provoque a escrita do maior número de páginas.
   **Para a ordem 3, a árvore que mais provocaria a escrita de maior número de páginas na inserção seria uma árvore onde todos (exceto as folhas) os nós tivessem 3 filhos e 2 chaves. Como a árvore com altura 4 estruturada na imagem abaixo.**

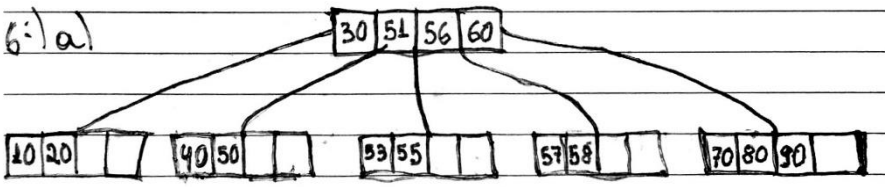


   c) Se uma chave não está numa folha em uma árvore B, seu antecessor e sucessor estão em folhas?
   **Sim, pois, o antecessor dessa chave será a chave de maior valor da ramificação a imediatamente à esquerda dessa chave no nó interno. E o sucessor será a chave de menor valor da ramificação imediatamente a direita dessa chave no nó interno.**

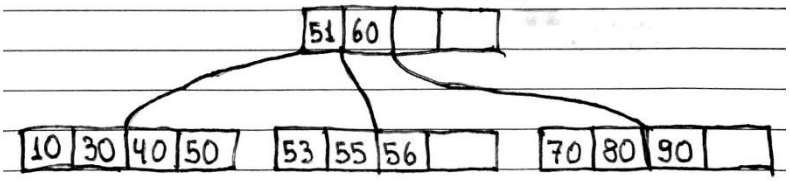
6. Considere a árvore B mostrada na figura abaixo.
   a) Mostre como ela ficaria se fossem inseridas as chaves 57 e 58.
   b) Mostre como ela (original, sem o 57 e 58) ficaria se for retirada a chave 20.



a)

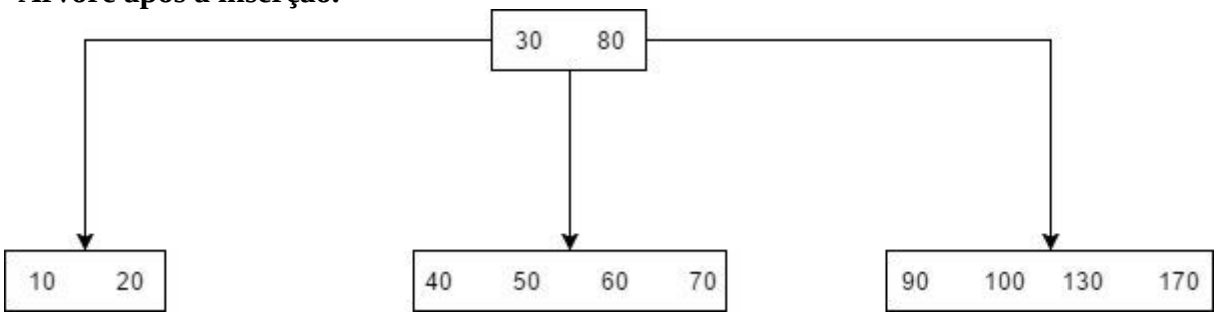



b)

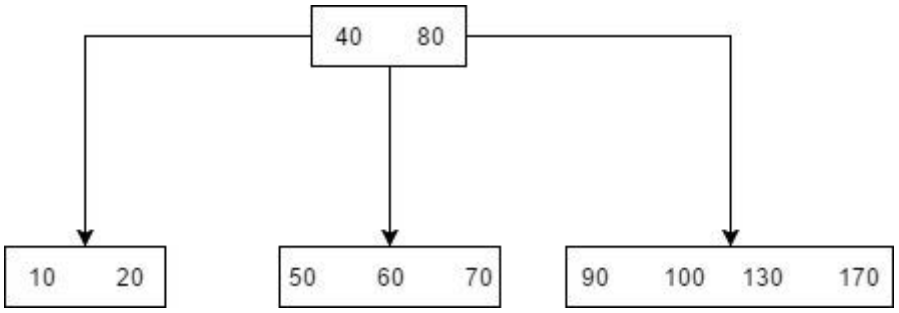



7. Inserir em uma árvore B de ordem 2 vazia, as chaves {10, 90, 170, 50, 80, 130, 100, 20, 30, 40, 70, 60} na ordem. Remova sucessivamente da árvore B a menor chave do nó raiz até que a árvore fique vazia. Quando houver possibilidade de optar entre a chave sucessora ou predecessora imediata, opte sempre pela predecessora. Indique as ocorrências de redistribuição ou concatenação. **(Vou considerar ordem 5, fazendo de acordo com a métrica estudada em sala.)**

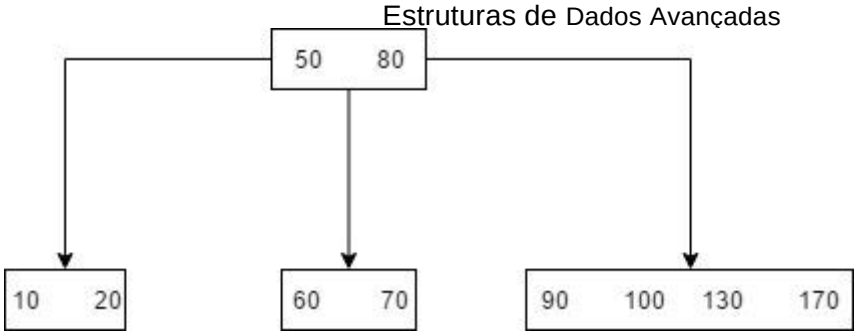
**Árvore após a inserção.**



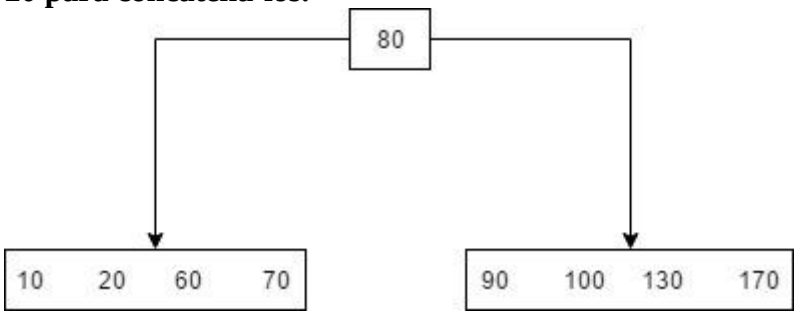

**Remove 30: Troco o 30 com o 20 (seu predecessor) e o removo. Após, faço a redistribuição do 20 - 40.**



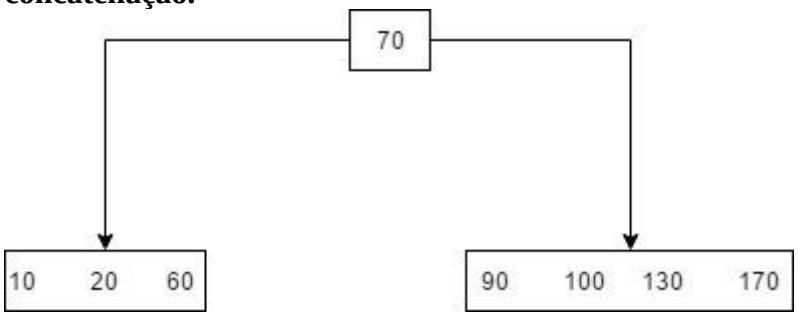

**Remove 40: Troco 40 com o 20 e o removo. Faço a redistribuição do 20 – 50.**



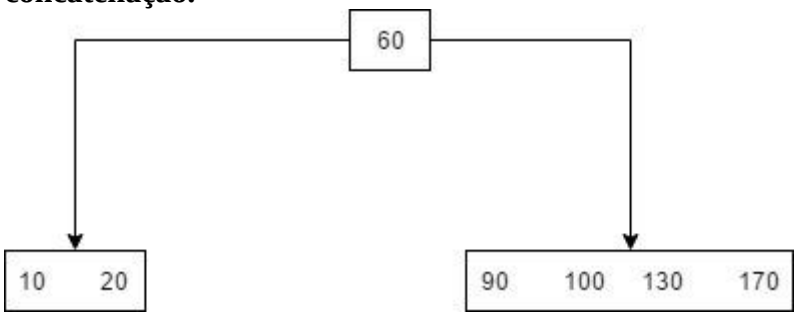

**Remove 50: Troco 50 com o 20 e o removo. Faço a concatenação o nó [10] e [60, 70], baixando o 20 para concatena-los.**



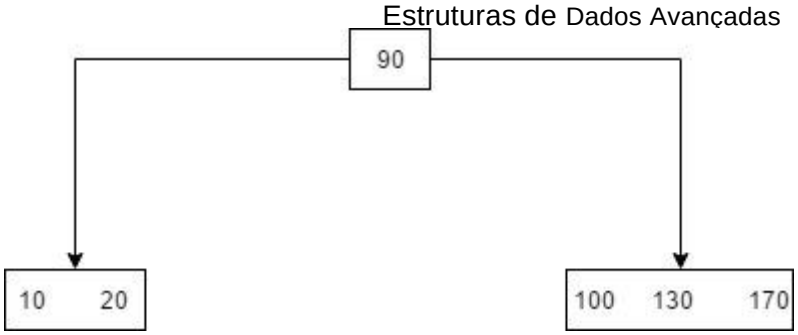

**Removo 80: Troco o 80 com o 70 e o removo. Não há necessidade de redistribuição ou concatenação.**



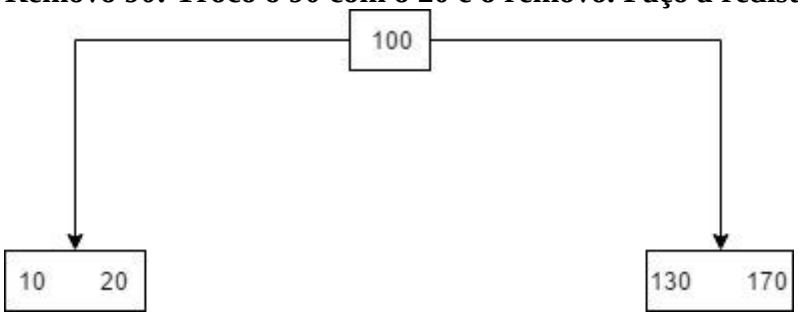

**Removo 70: Troco o 70 com o 60 e o removo. Não há necessidade de redistribuição ou concatenação.**




**Removo 60: Troco o 60 com o 20 e o removo.  Faço a redistribuição do 20-90.**



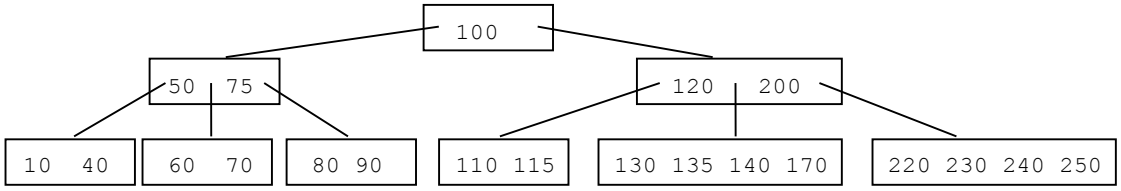

**Removo 90: Troco o 90 com o 20 e o removo. Faço a redistribuição do 20-100.**




**Removo 100: Troco 100 com o 20 e o removo. Faço a concatenação dos nós restantes.**

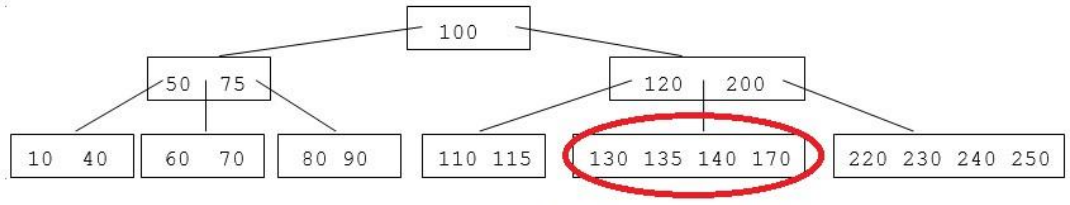

**De agora em diante é só remover cada uma das chaves.**

8. Considere a árvore B de ordem 2 abaixo:

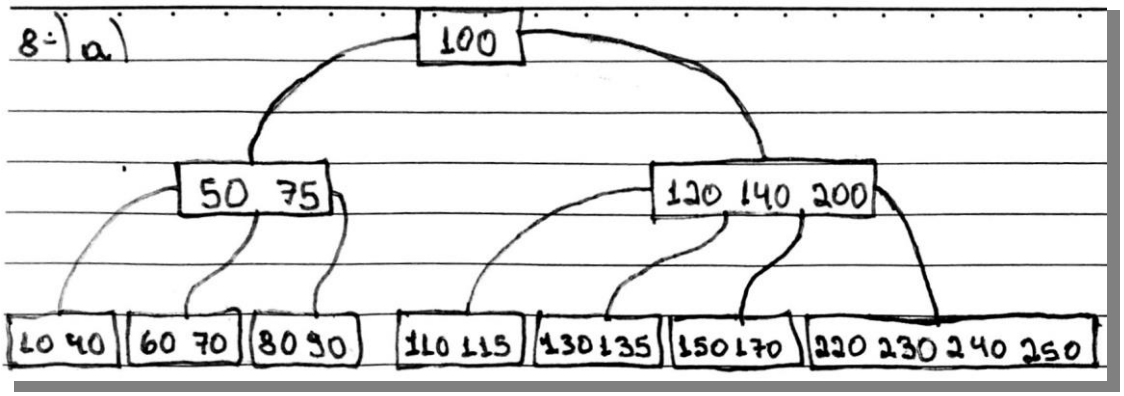



Realize as seguintes operações, utilizando sempre a árvore resultante da operação anterior. Redesenhe a árvore a cada passo, indicando os nós que sofrem modificações, bem como a ocorrência de CISÃO (*Split*), REDISTRIBUIÇÃO ou CONCATENAÇÃO (*Merge*):

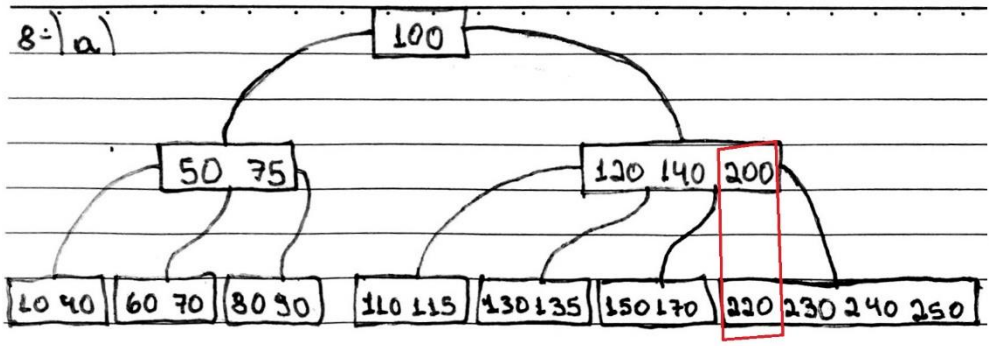
a) inserção de 150



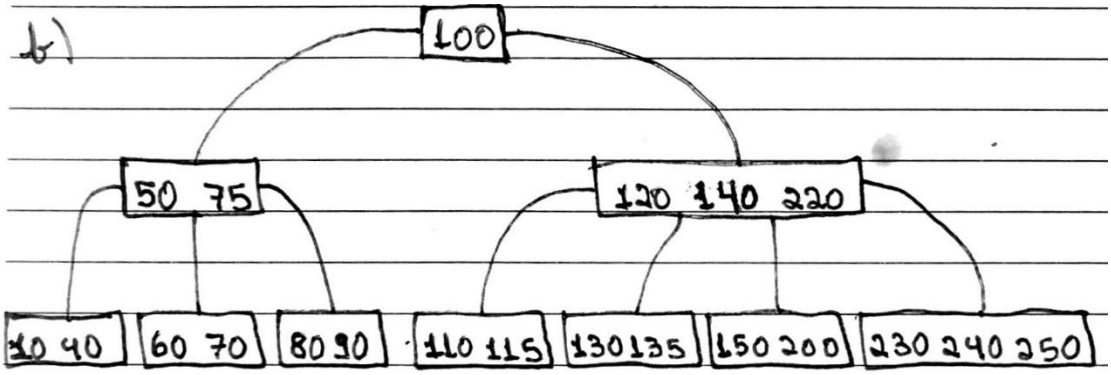

Overflow com 150: Split



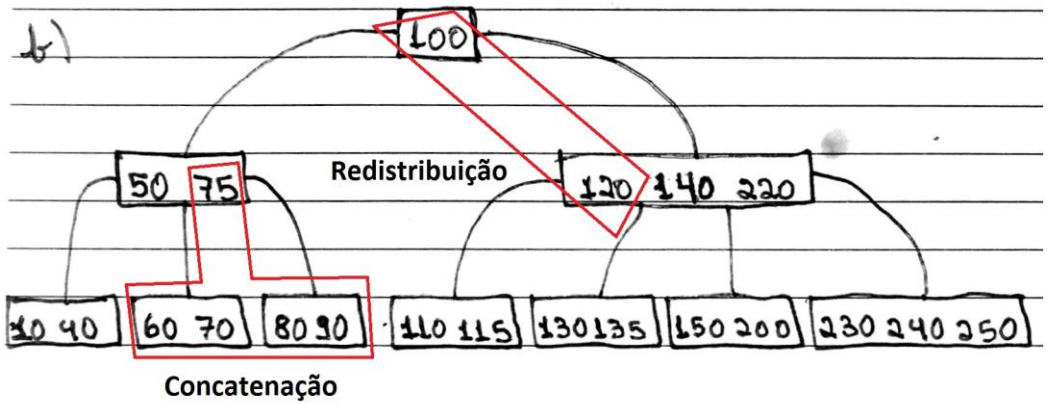

b) remoção de 170

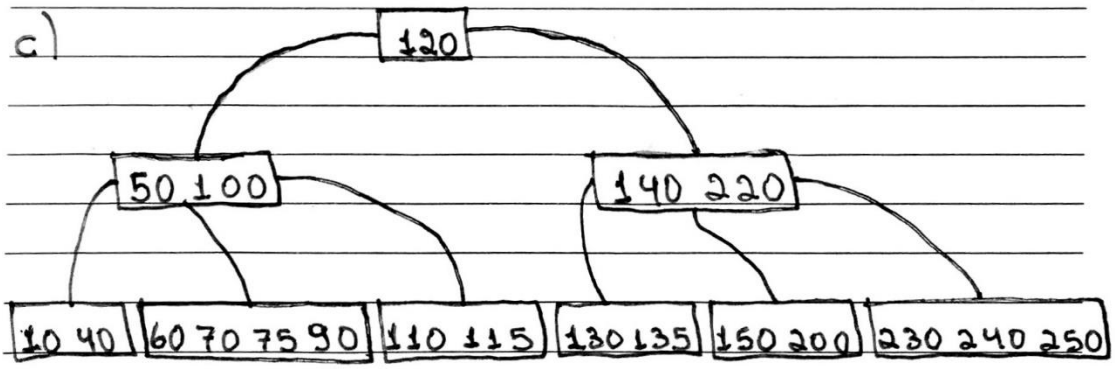



Redistribuição

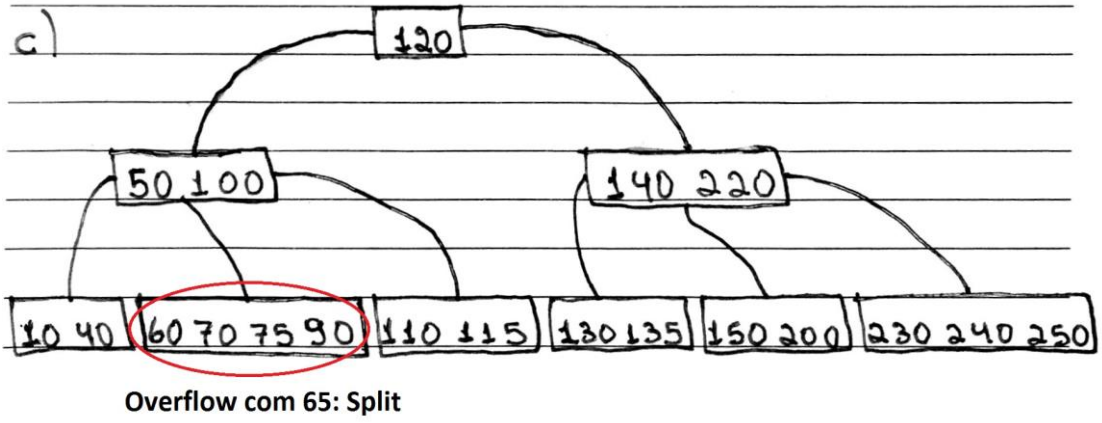


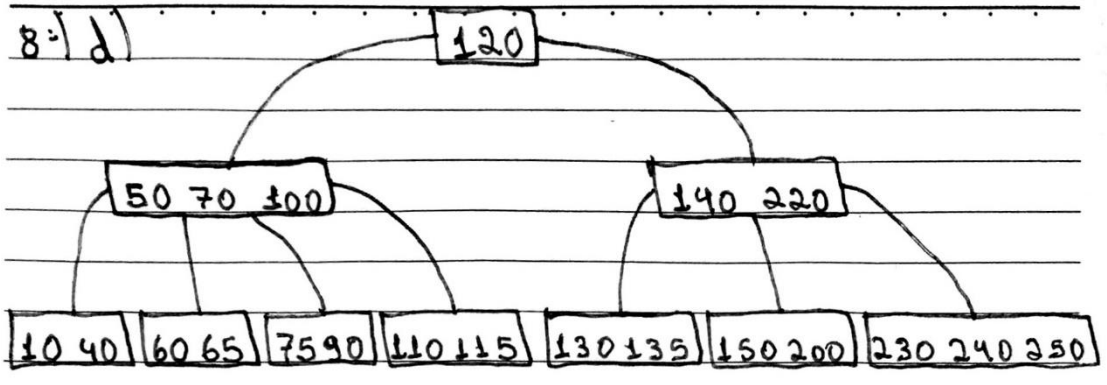
c) remoção de 80



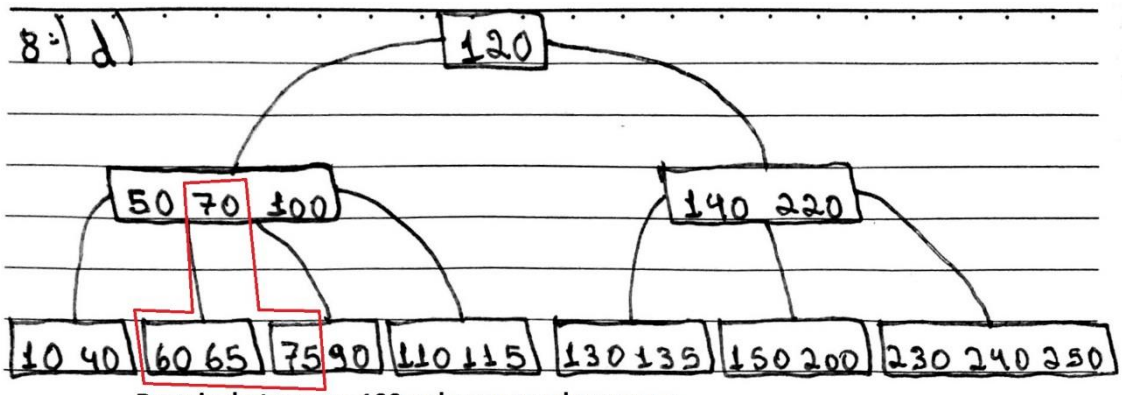




d) inserção de 65

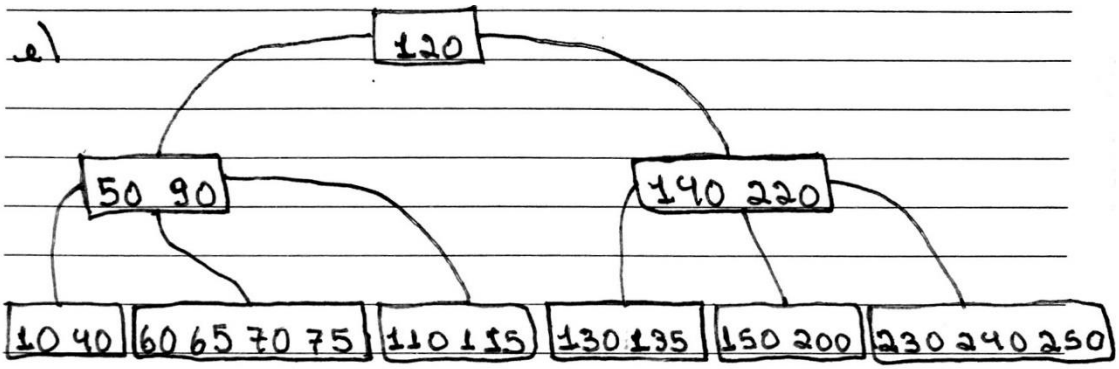


e) remoção de 100




**Depois de trocar o 100 pelo seu predecessor e eliminá-lo eu concateno.**



9. A estrutura em C do nó de uma árvore B de ordem 2 é a seguinte:

```
#define MAX 4
#define MIN 2
typedef struct no t_no;
struct no
{
 int     chave[MAX+1];
 t_no    *ramo[MAX+1];
int      ndesc;

};
```

Escreva em Scala um programa que, recebendo a entrada para a árvore (`arv`) e os limites inferior (`lim_inf`) e superior (`lim_sup`), percorre a árvore B (`arv`) em ordem simétrica e imprime todas as chaves x tais que: `lim_inf < x < lim_sup` (assuma que `lim_inf <= lim_sup`).

```
def imprimir(arv: ArvoresB, lim_inf: Int, lim_sup: Int) = {
   imprimirRecursivo(arv.raiz, lim_inf, lim_sup)
}

def imprimirRecursivo(no: No, lim_inf: Int, lim_sup: Int){
   if(no != null){
         for(i <- 0 to no.ndesc -1){
               imprimirRecursivo(no.ramo(i), lim_inf, lim_sup)
               if(no.chave(i) > lim_inf && no.chave(i) < lim_sup){
                     println(no.chave(i))
               }
         }
         imprimirRecursivo(no.ramo(no.ndesc), lim_inf, lim_sup)
   }
}
```