Lista de exercícios 4
Árvore B

Lucas Emanuel Resck Domingues

Estruturas de Dados e Algoritmos
Professor: Jean Roberto Ponciano

24 de agosto de 2021

1. Explique como encontrar a maior chave armazenada em uma árvore B.

   Solução: Iniciamos a busca pela maior chave com um ponteiro para a raiz. Logo após, tomamos o ponteiro à direita da última chave do nó. Repetimos esse processo, até que estejamos em uma folha. Na folha, tomamos a última chave do nó. Veja o Algoritmo 1.

Algorithm 1 Busca da maior chave em uma árvore B.

```
procedure MAIORCHAVE(x)
    i ← n[x]                          ▷ n[x] é o número de chaves do nó x
    if leaf(x) then                   ▷ Caso x seja uma folha
        return key_{i-1}[x]           ▷ Retorna a última chave do nó
    else
        return MAIORCHAVE(c_i[x])     ▷ Maior chave na subárvore à direita da última chave
    end if
end procedure
```

   A corretude do algoritmo vem por indução. O que estamos fazendo é buscando a maior chave da subárvore que contém todas as chaves que são maiores do que as chaves da raiz; a chave que queremos precisa estar nessa subárvore. Após isso, repetimos o mesmo processo. Fazemos isso até que não tenhamos mais essa subárvore, e retornamos a maior chave do nó pai.

3. Escreva um algoritmo que, dada uma árvore B e uma chave, retorne o sucessor imediato da chave.

   Solução: De início, assumo que a chave não necessariamente está armazenada na árvore. Vamos nos beneficiar do algoritmo de busca estudado em aula: suponha que tenhamos uma função B-TREE-SEARCH que toma um ponteiro a um nó x e uma chave k e retorna o (um ponteiro ao) nó y em que a chave k está ou deveria estar, caso estivesse armazenada.

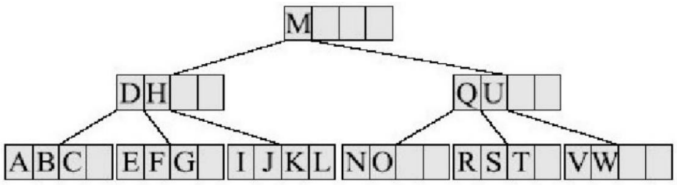
   - Caso k seja realmente esteja em y: o sucessor imediato de uma chave k é, dentre as chaves que são maiores, aquela menor. Isso significa que, se tomarmos o ponteiro à direita da chave k, o sucessor imediato é o menor nó dessa subárvore. Quando não existe essa subárvore (isto é, o ponteiro aponta para NULL), pegamos simplesmente a próxima chave dentro do nó.
   - Caso k não esteja em y: a subárvore que deveria conter k não existe, isto é, o ponteiro aponta para NULL, como visto no algoritmo de busca. Logo, a solução aqui é simplesmente encontrar qual é o sucessor dentro do nó y. Caso não exista um sucessor, retornamos algo que indique o erro, por exemplo, *None*.

   O Algoritmo 2 descreve esse procedimento. Note que usamos a função de busca estudada em aula e também uma função MENORCHAVE, análoga à função MAIORCHAVE do Exercício 1.

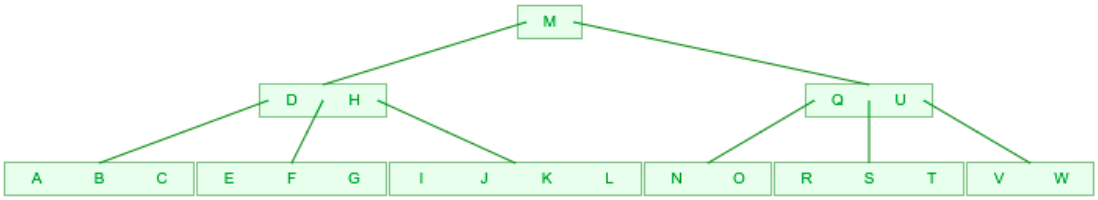
Algorithm 2 Sucessor imediato de uma chave em uma árvore B.

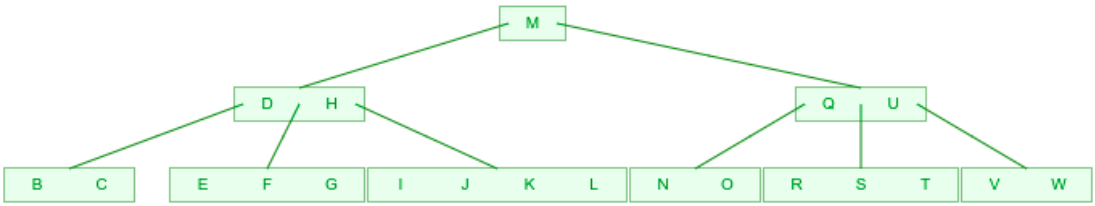
Require: Árvore B (ponteiro à raiz), chave k
  $y \leftarrow$ B-TREE-SEARCH(B, k) ▷ Encontra o nó em que k está, ou deveria estar
  for $0 \leq i < n[y]$ **do**
    if $\text{key}_i[y] == k$ **then** ▷ Se k está em y
      if $c_{i+1}[y] ==$ NULL **then** ▷ Se não existe subárvore para buscar o menor elemento
        if $i < n[y] - 1$ **then**
          return $\text{key}_{i+1}[y]$ ▷ Retorna a próxima chave
        else
          return *None* ▷ Não existe a chave sucessora
        end if
      else
        return MENORCHAVE$(c_{i+1}[y])$ ▷ Análoga a MAIORCHAVE (Exercício 1)
      end if
    end if
    if $\text{key}_i[y] > k$ **then** ▷ k não está em y, já retornamos a próxima chave
      return $\text{key}_i[y]$
    end if
  end for
  return *None* ▷ Não há uma chave sucessora

4. Mostre, a cada passo, as árvores que resultam depois da remoção das chaves A, B, M, Q e R da árvore B de ordem 5 a seguir.

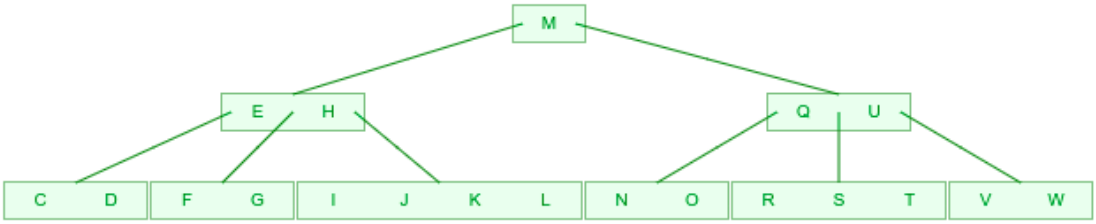


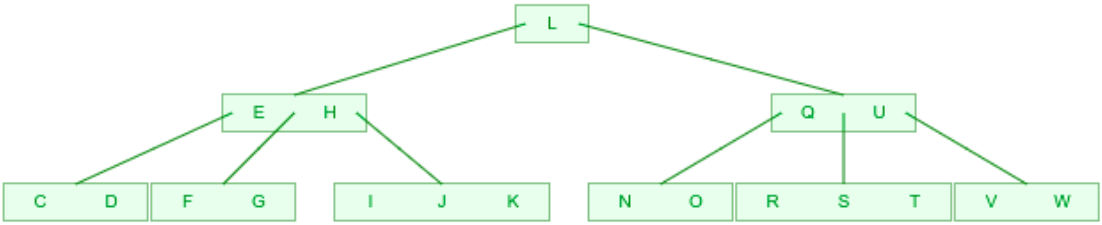
Solução: As Figuras a seguir mostram os passos da remoção das chaves A, B, M, Q e R da árvore B do enunciado. Foi utilizada a ferramenta B-Tree Visualization[1], da University of San Francisco, porém apenas a título de visualização; os resultados foram feitos à mão "no papel".

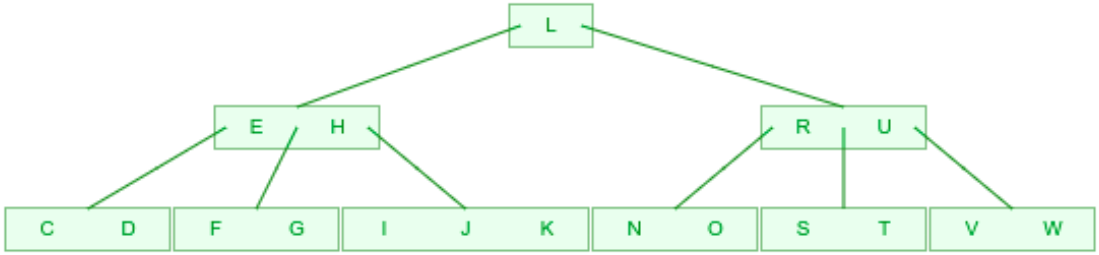


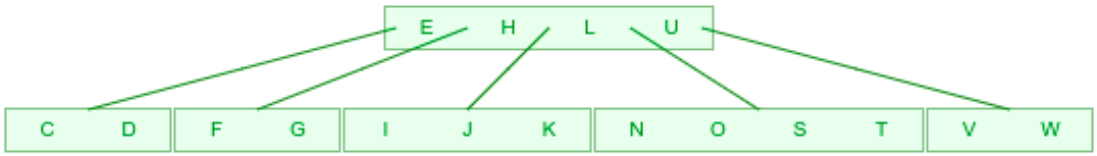


[1]Disponível em https://www.cs.usfca.edu/ galles/visualization/BTree.html.



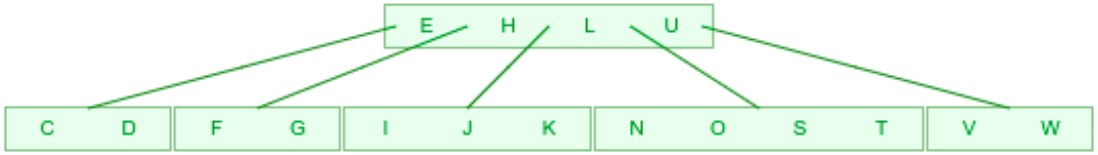


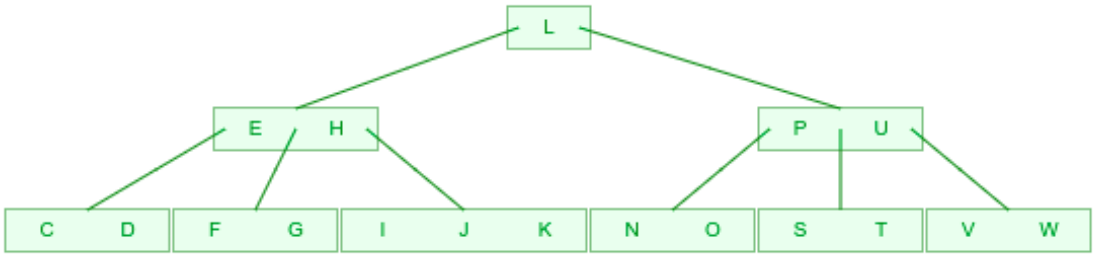


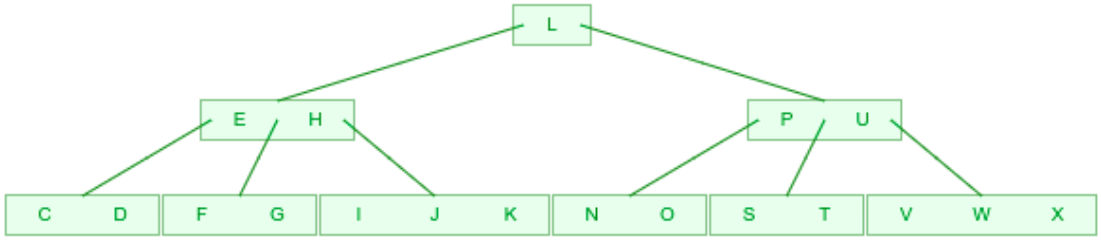


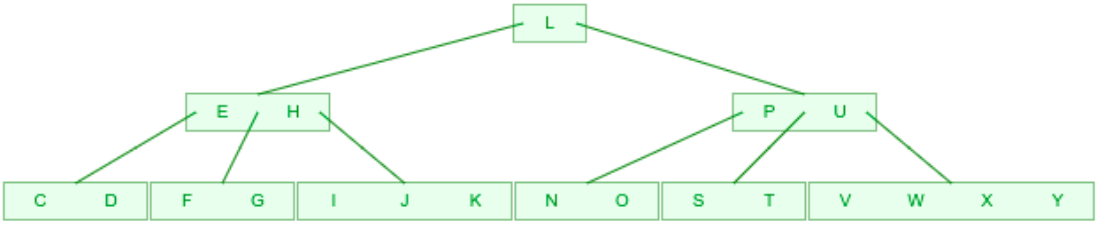
5. Considerando a árvore resultante do exercício anterior (após as remoções pedidas), insira P, X, Y, Z, Q. Mostre o passo a passo.

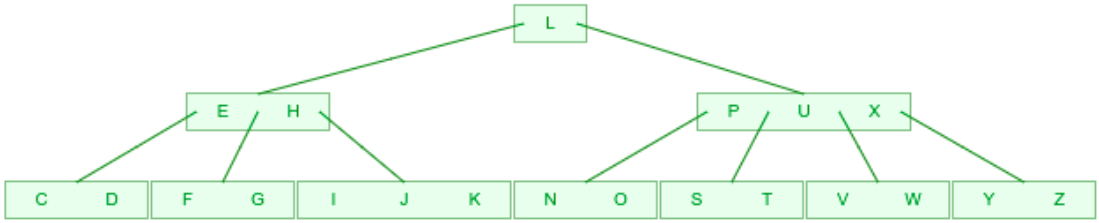
Solução:

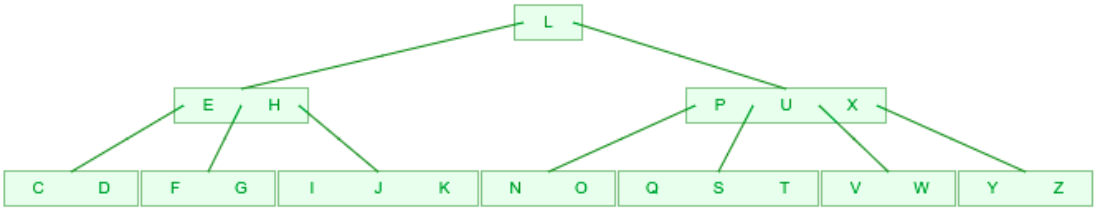












6. Para pesquisar: Que diferenças existem entre as árvores B, B+ e B*?

   Solução: As árvores B+ e B* são extensões da árvore B. Ou seja, elas incorporam a maioria das características da árvore B, com exceção de:

   - **Árvore B+:** Os registros são armazenados apenas nas folhas, isto é, os nós internos (que não são folhas) guardam apenas algumas chaves. Além disso, as folhas são encadeadas para acesso sequencial. Veja, por exemplo, a Figura 1. Nela, não estão representados os registros, claramente, mas, se estivessem, estariam apenas nas folhas.

Porém, podemos ver que algumas chaves presentes nas folhas são repetidas nos nós internos; isso acontece com a primeira chave de cada nó folha. O que é interessante sobre essa árvore é que, dado que possuímos acesso ao nó que contém as chaves 25 e 28, por exemplo, é trivial avançar para o nó que contém as próximas chaves 30 e 35. Outras diferenças da árvore B+ com relação à árvore B são os algoritmos de busca, inserção e remoção, que são adaptados para lidar com essas novas características.

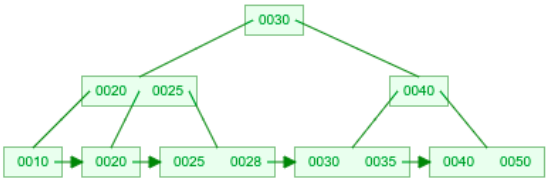


Figura 1: Exemplo de árvore B+. Criado com a ferramenta B+ Tree Visualization[3], da University of San Francisco.

- **Árvore B*:** O objetivo dessa árvore é manter os nós internos os mais cheios possível. Isso é feito pedindo que os nós contenham no mínimo 2/3 da sua capacidade, ao invés de 1/2, como na árvore B. Além disso, os algoritmos de inserção e remoção são modificados. Por exemplo, ao invés de simplesmente dividir um nó quando ele está cheio, busca-se redistribuir as chaves com os irmãos, evitando o custo de divisão de nós.

[3]Disponível em https://www.cs.usfca.edu/ galles/visualization/BPlusTree.html.