UNIVERSIDADE PRESBITERIANA MACKENZIE
- Faculdade de Computação e Informática –

Disciplina: Estrutura de dados
Profa. Valéria Farinazzo

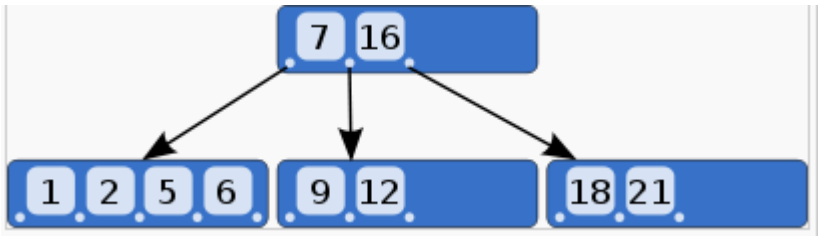
Árvore B

Definição

As árvores B são árvores balanceadas projetadas para trabalhar com dispositivos de armazenamento secundário como discos magnéticos.

Elas visam otimizar as operações de entrada e saída nos dispositivos. O tempo de acesso às informações em um disco é prejudicado principalmente pelo tempo de posicionamento do braço de leitura. Uma vez que o braço esteja posicionado no local correto, a leitura pode ser feita de forma bastante rápida. Desta forma, devemos minimizar o número de acessos ao disco.

Diferente das árvores binárias, cada nó em uma árvore B pode ter muitos filhos, isto é, o grau de um nó pode ser muito grande. A figura a seguir apresenta um exemplo de árvore B.



Definição: Uma árvore B possui as seguintes propriedades:

1. Todo o nó X possui os seguintes campos:
    a. n, o número de chaves armazenadas em X;
    b. as n chaves k1, k2...kn são armazenadas em ordem crescente;
    c. folha, que indica se X é uma folha ou um nó interno.
2. Se X é um nó interno então ele possui n+1 ponteiros f1, f2...fn+1 para seus filhos (podendo alguns serem nulos)
3. Se ki é alguma chave na sub-árvore apontada por fi, então
4. Todas as folhas da árvore estão na mesma altura (que é a altura da árvore).
5. Existe um número máximo e mínimo de filhos em um nó. Este número pode ser descrito em termos de um inteiro fixo t maior ou igual a 2 chamado grau mínimo.
    a. Todo o nó diferente da raiz deve possuir pelo menos t-1 chaves. Todo o nó interno diferente da raiz deve possuir pelo menos t filhos. Se a árvore não é vazia, então a raiz possui pelo menos uma chave.
    b. Todo o nó pode conter no máximo 2t - 1 chaves. Logo um nó interno pode ter no máximo 2t filhos. Dizemos que um nó é cheio se ele contém 2t - 1 chaves.

Estrutura do Nó

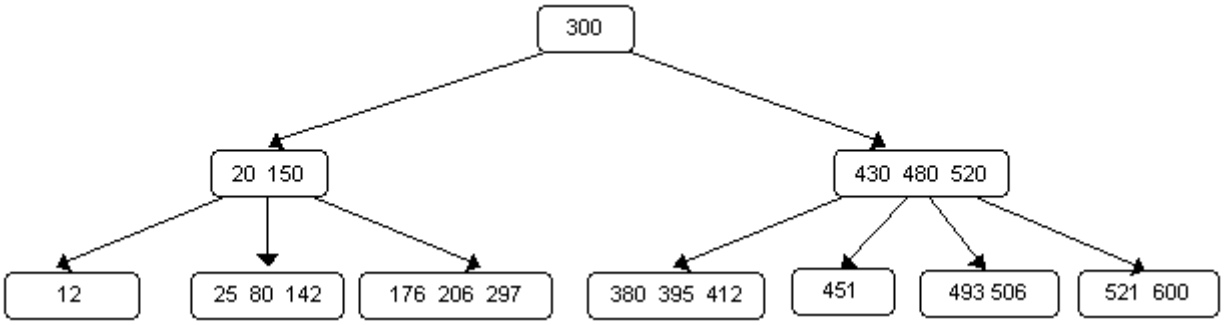
```
const t = 2;
typedef struct no_arvoreB arvoreB;

struct no_arvoreB {
  int num_chaves;
  char chaves[2*t-1];
  arvoreB *filhos[2*t];
  bool folha;
};
```

De acordo com a definição acima a árvore B mais simples ocorre quando t=2. Neste caso todo o nó diferente da raiz possui 2, 3 ou 4 filhos. Esta árvore é também conhecida por **árvore 2-3-4**.

Árvore com t = 2

De acordo com a definição acima a árvore B mais simples ocorre quando t=2. Neste caso todo o nó diferente da raiz possui 2, 3 ou 4 filhos. Esta árvore é também conhecida por árvore 2-3-4.



Dizemos que uma árvore B é de **ordem n** quando n representa o número máximo de filhos de um nó, exceto a raiz. Por exemplo, uma árvore de ordem 4 terá máximo 4 filhos. Porém a palavra "ordem" é usada de formas diferentes pelos autores, podendo indicar o número máximo de chaves em cada nó, ou até mesmo a ocupação mínima em cada nó.

Altura da árvore

O número de acessos requeridos pelas operações em uma árvore B é proporcional à altura da árvore.

Teorema: Uma árvore B com n chaves, altura h e grau mínimo t >= 2 satisfaz a relação:

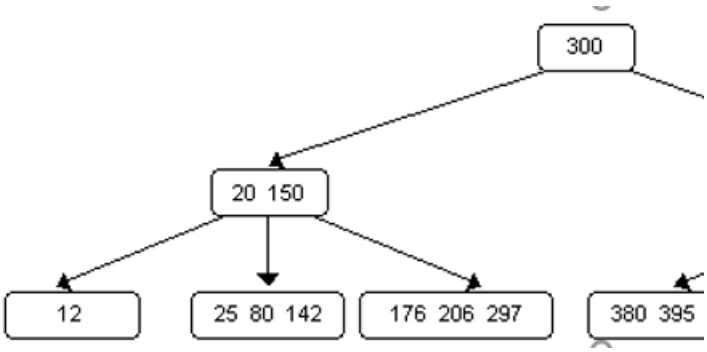
$$h \leq \log_t \frac{n+1}{2}$$

Busca em uma árvore B

A busca em uma árvore B é uma função parecida com a de busca em uma árvore de busca binária, exceto o fato de que se deve decidir entre vários caminhos. Como as chaves estão ordenadas, basta realizar uma busca binária nos elementos de cada nó. Isso levará tempo **O(lg(t))**. Se a chave não for encontrada no nó em questão, continua-se a busca nos filhos deste nó, realizando-se novamente a busca binária. Caso o nó não esteja contido na árvore a busca terminará ao encontrar um ponteiro igual a **NULL**, ou de forma equivalente, verificando-se que o nó é uma folha. A busca completa pode ser realizada em tempo **O(lg(t)logt(n))**.

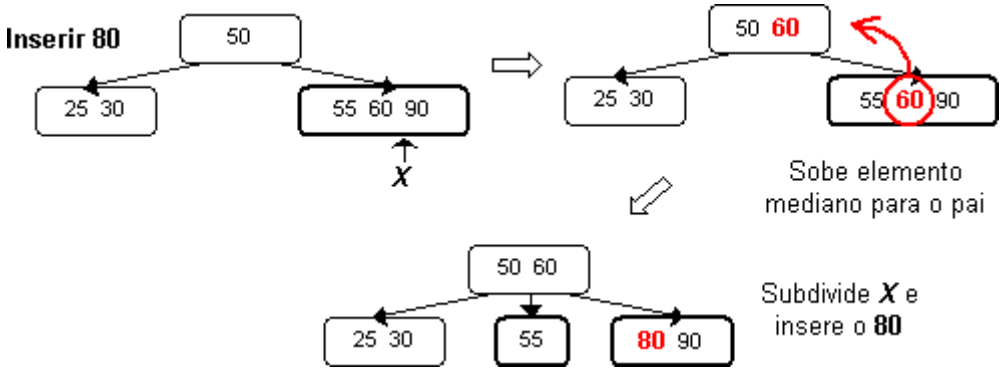
A seguir, é apresentado um algoritmo para a busca em árvore B.

```
BuscaB(x, pt, f, g):
  p = ptraiz; pt = NULL; f = 0;
  while (p != NULL) {
    i = 1; g = 1; pt = p;
    while (i <= m) {
      if (x > p->s[i]) {
        i = i + 1; g = i;
      } else if (x == p->s[i]) {
        p = NULL; f = 1;
      } else {
        p = p->pont[i - 1];
        i = m + 2;
      }
    }
    if (i == m + 1)
      p = p->pont[m];
  }
```



Inserção em uma árvore B

Para inserir um novo elemento em uma árvore B, basta localizar o nó folha X onde o novo elemento deva ser inserido. Se o nó X estiver cheio, será necessário realizar uma subdivisão de nós que consiste em passar o elemento mediano de X para seu pai e subdividir X em dois novos nós com t - 1 elementos e depois inserir a nova chave.



Passos para a inserção da chave 80 em uma árvore B $t = 2$

Se o pai de X também estiver cheio, repete-se recursivamente a subdivisão acima para o pai de X. No pior caso terá que aumentar a altura da árvore B para poder inserir o novo elemento.

Note que diferentemente das árvores binárias, as árvores B crescem para cima.

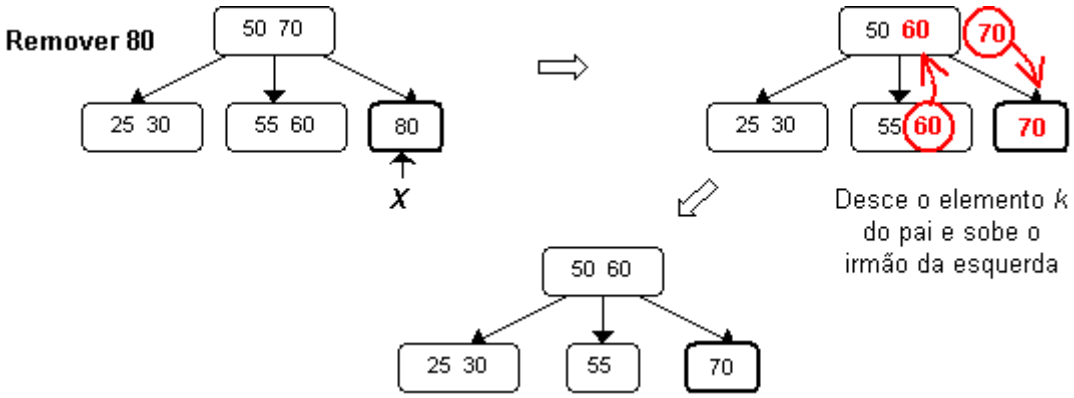
Remoção em uma árvore B

A remoção de um elemento de uma árvore B pode ser dividida em dois casos:

1. O elemento que será removido está em uma folha
2. O elemento que será removido está em um nó interno.

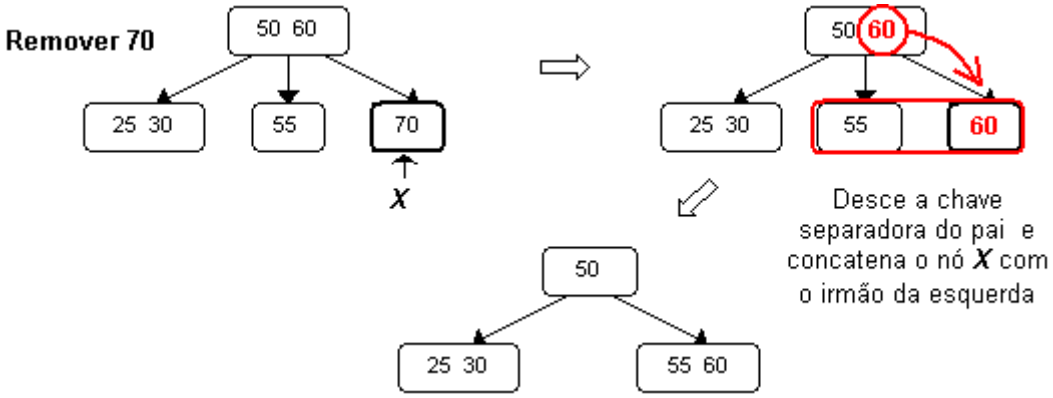
Se o elemento estiver sendo removido de um nó não folha, seu sucessor, que deve estar em uma folha, será movido para a posição eliminada e o processo de eliminação procede como se o elemento sucessor fosse removido do nó folha.

Quando um elemento for removido de uma folha ***X*** e o número de elementos no nó folha diminui para menos que ***t - 1***, deve-se reorganizar a árvore B. A solução mais simples é analisarmos os irmãos da direita ou esquerda de ***X***. Se um dos irmãos (da direita ou esquerda) de ***X*** possui mais de ***t - 1*** elementos, a chave ***k*** do pai que separa os irmãos pode ser incluída no nó ***X*** e a última ou primeira chave do irmão (última se o irmão for da esquerda e primeira se o irmão for da direita) pode ser inserida no pai no lugar de ***k***.



Passos para a remoção da chave 80 em uma árvore B com *número mínimo de chaves = 1 e número máximo de chaves = 2*

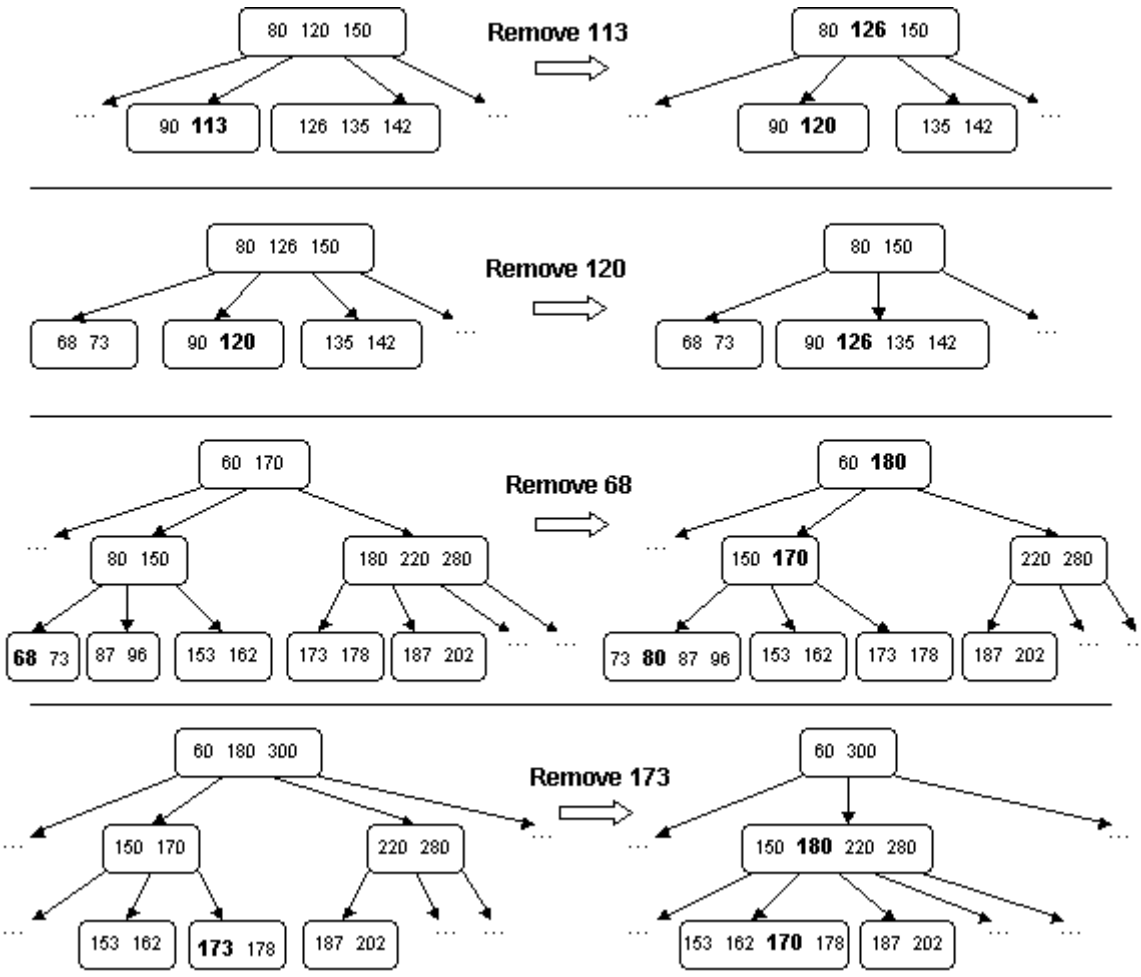
Se os dois irmãos de ***X*** contiverem exatamente ***t - 1*** elementos (ocupação mínima), nenhum elemento poderá ser emprestado. Neste caso, o nó ***X*** e um de seus irmãos são concatenados em um único nó que também contém a chave separadora do pai.



Passos para a remoção da chave 70 em uma árvore B com *número mínimo de chaves = 1 e número máximo de chaves = 2*

Se o pai também contiver apenas ***t - 1*** elementos, deve-se considerar os irmãos do pai como no caso anterior e proceder recursivamente. No pior caso, quando todos os ancestrais de um nó e seus irmãos contiverem exatamente ***t - 1*** elementos, uma chave será tomada da raiz e no caso da raiz possuir apenas um elemento a árvore B sofrerá uma redução de altura.

A figura abaixo ilustra a remoção de elementos em uma árvore B com *t=3*.



Exercícios

1. Como uma folha de uma árvore B difere de um nó interno?

2. Quais são as partes necessárias a uma folha?

3) Mostre as árvores-B de ordem 4 (máximo de 4 filhos) resultantes da entrada das letras abaixo na ordem apresentada.

a) C G J X

b) C G J X N S U O A E B H I

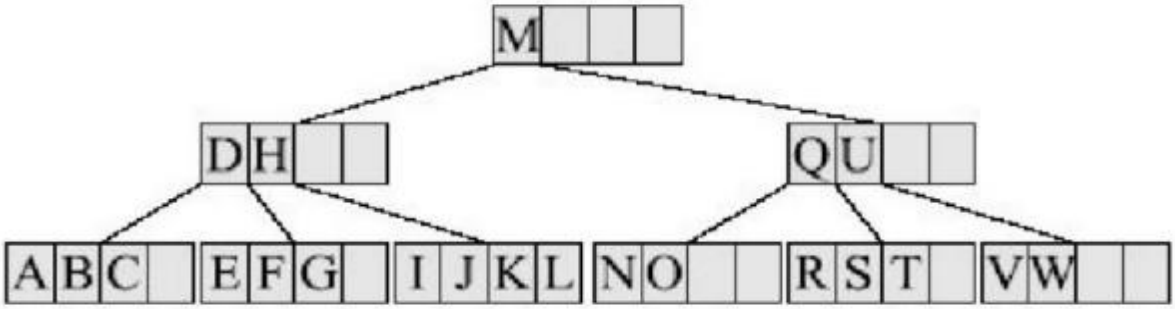
c) C G J X N S U O A E B H I F

d) C G J X N S U O A E B H I F K L Q R T V U W Z

4. Mostre os resultados de inserir as chaves a seguir em uma árvore B com mínimo de 1 chaves e máximo de 2 chaves, inicialmente vazia: 14, 39, 1, 6, 41, 32, 8, 38, 43, 3, 36. Após essas inserções, mostre os resultados de remover as chaves ímpares nessa estrutura.

5. Explique como encontrar a menor chave armazenada em uma árvore B.

6. Mostre a cada passo, as árvores que resultam depois de remoção das chaves A, B, Q e R da árvore-B de **ordem** 5 (máximo de 5 filhos) na figura a seguir:


M
D H
Q U
A B C
E F G
I J K L
N O
R S T
V W