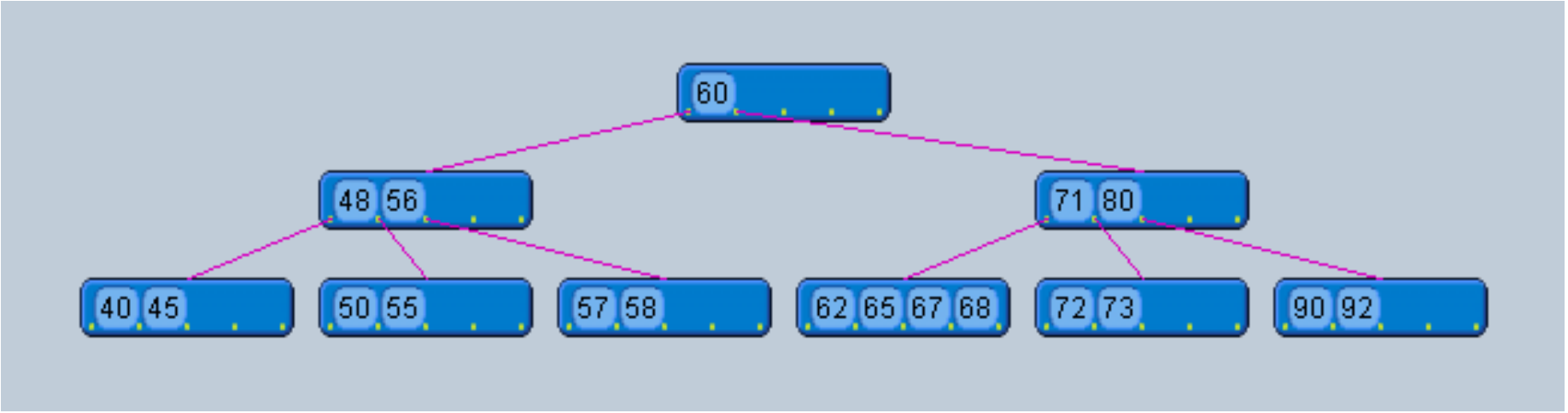
B-Tree

Exemplo de uma B-tree de ordem 5

- m = 5 (ordem)
- n = 4 (chaves)
- T = 3 (grau mínimo)
- Nenhum nodo pode ter menos o que 3 filhos ou 2 chaves.



Altura da B-tree

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

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

Altura da B-tree

- Se uma árvore B possui altura ***h***, o número de nós é minimizado quando a raiz contém uma chave e todos os outros nós contém ***t - 1*** chaves.
- Neste caso, existem 2 nós no nível 1, $2t$ nós no nível 2, $2t^2$ nós no nível 3, e na altura ***h*** terá $2t^{h-1}$ nós.
- Logo, o número de chaves satisfaz:

$$n \geq 1 + (t-1)\sum_{i=1}^{h} 2t^{i-1} = 1 + 2(t-1)\frac{t^h - 1}{t-1} = 2t^h - 1$$

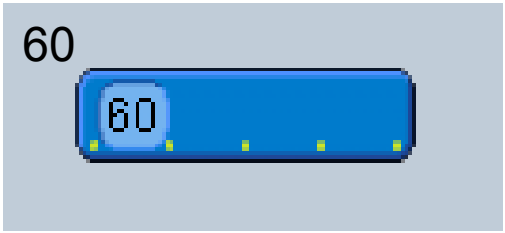
Busca em B-tree

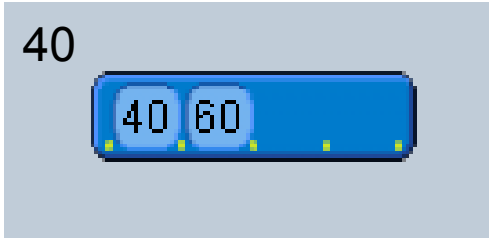
- Processo similar à busca em uma árvore binária.
- Comparação entre os valores da chave de busca e das chaves armazenadas na B-tree.
- Como as chaves estão ordenadas, pode-se realizar uma busca binária nos elementos de cada nó.
  - Custo a busca binária: ***O(lg(t))***.
- Se a chave não for encontrada no nó em questão, continua-se a busca nos filhos deste nó, realizando-se novamente a busca binária.
- Custo da busca completa
  - $O(lg(t).log_t(n))$

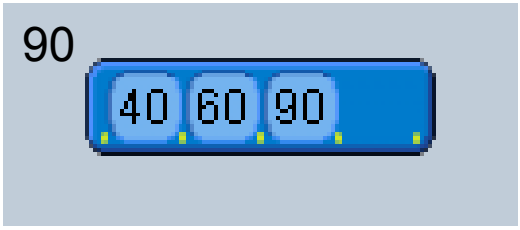
Inserção em B-tree

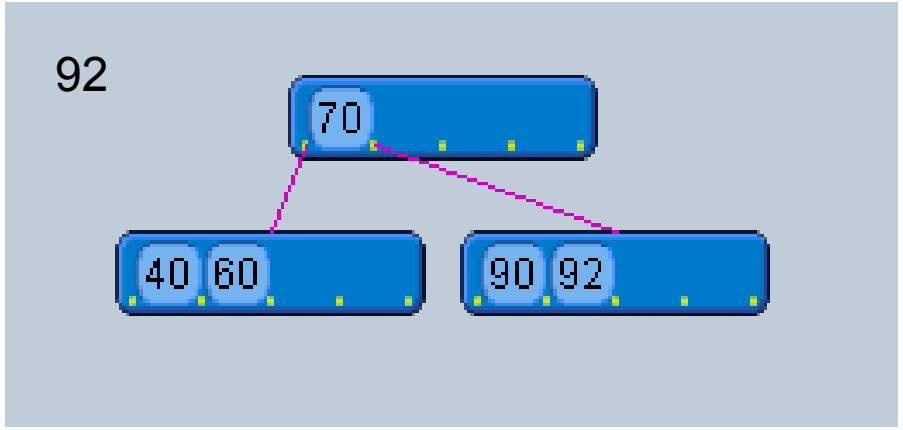
- Localizar o nó folha ***X*** onde o novo elemento deve ser inserido.
- Se o nó ***X*** estiver cheio, realizar uma subdivisão de nós
  - passar o elemento mediano de ***X*** para seu pai
  - subdividir ***X*** em dois novos nós com ***t - 1*** elementos
  - inserir a nova chave
- Se o pai de ***X*** 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

Exemplos de inserção em uma B-tree



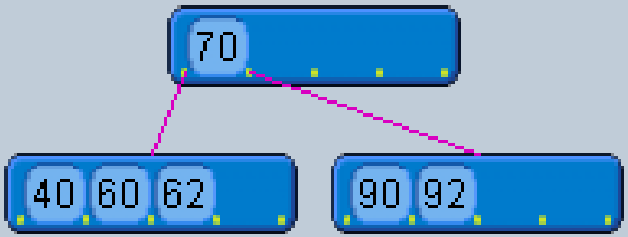




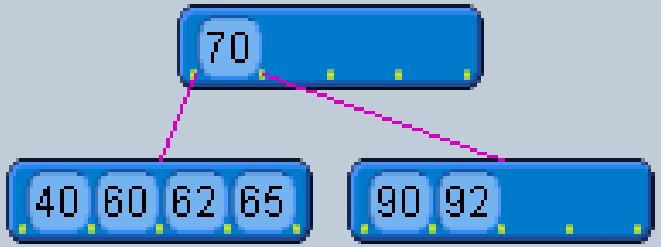


Exemplos de inserção em uma B-tree

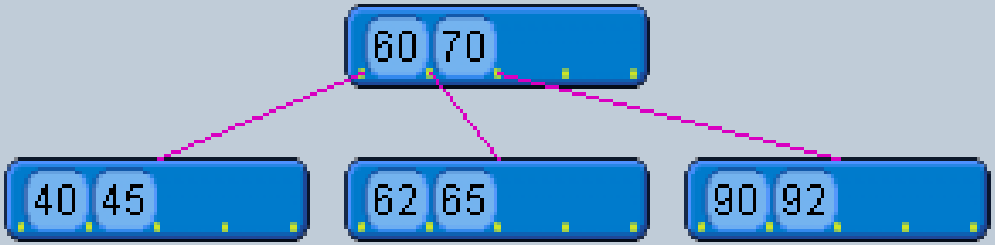
62



65

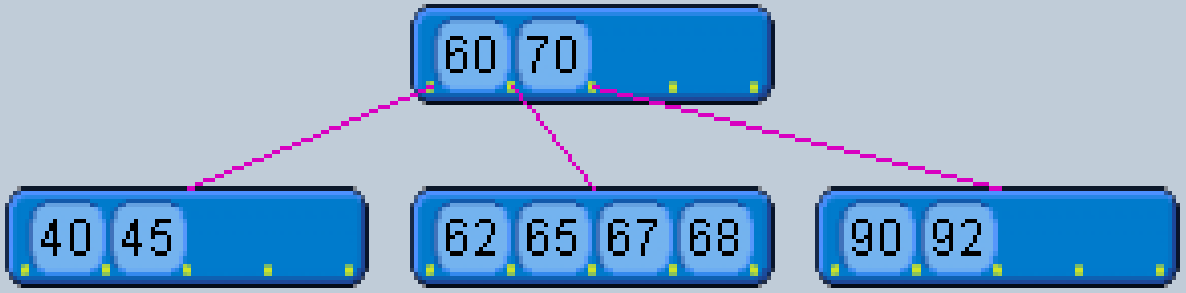


45

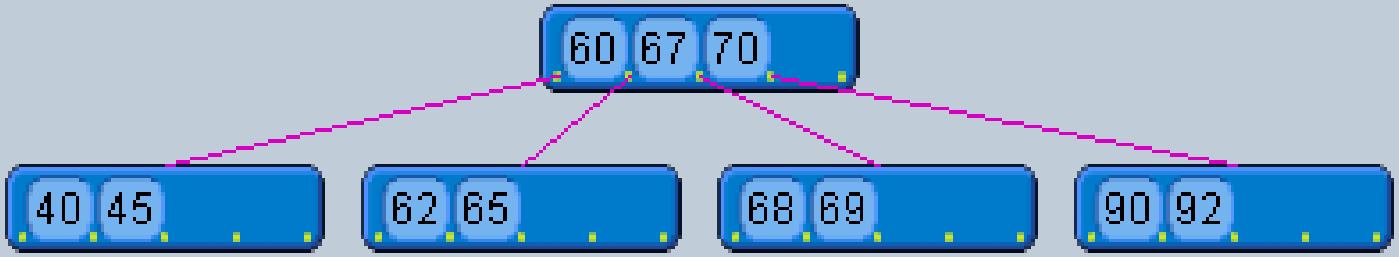


Exemplos de inserção em uma B-tree

67, 68

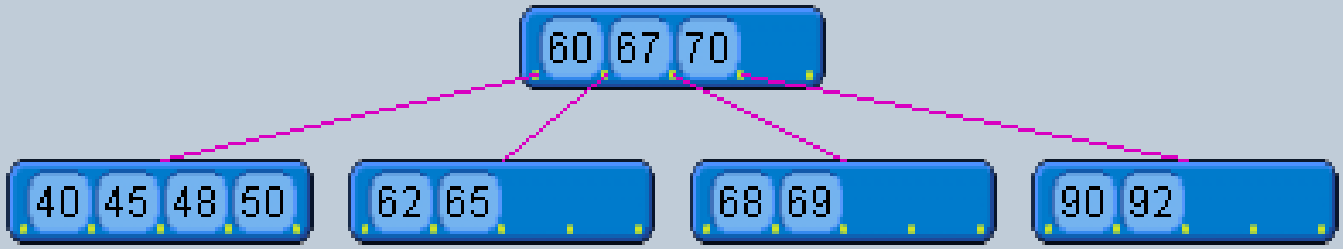


69



Exemplos de inserção em uma B-tree

48, 50

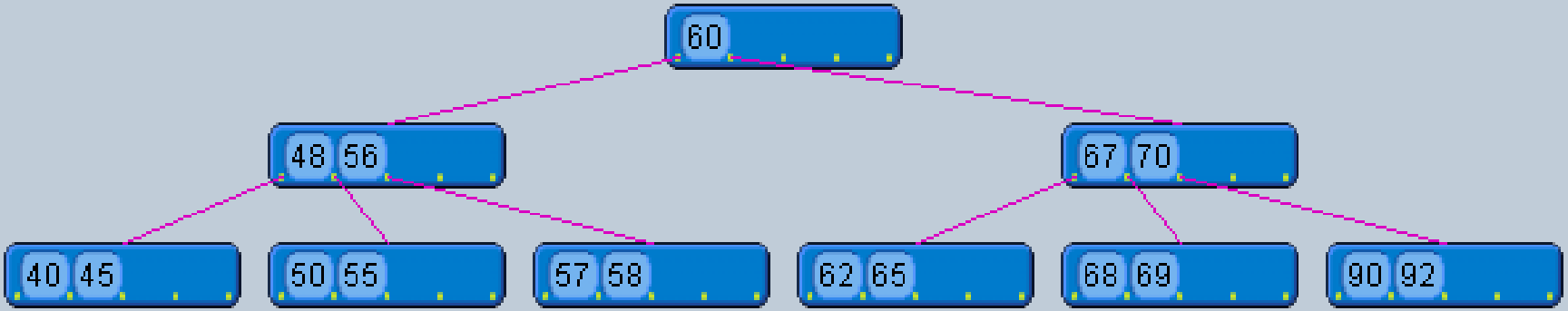


55

Exemplos de inserção em uma B-tree

56, 57

58



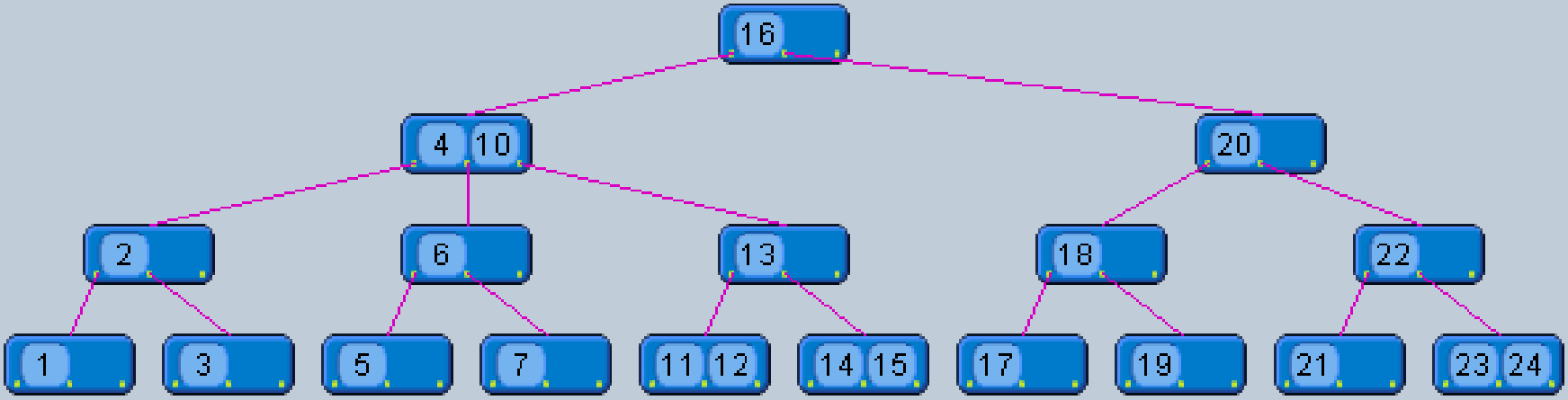
Remoção em B-Tree

- Dois casos:
  - O elemento que será removido está em um nó interno (não folha).
  - O elemento que será removido está em uma folha.

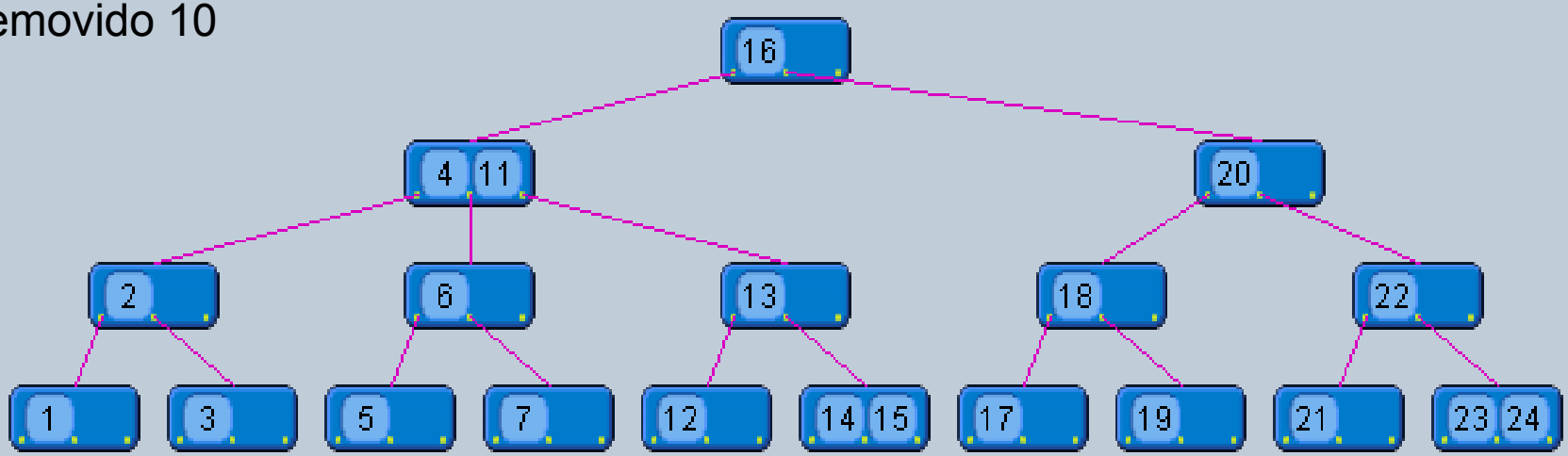
Elemento removido de um nó não folha

- Seu sucessor, que sempre estará em uma folha, será movido para a posição eliminada e o processo de eliminação procede com a eliminação de um valor de nó folha (o valor sucessor ao que está sendo removido, nesse caso).

Remover o valor 10



Removido 10



Elemento removido de um nó folha

- Quando um elemento for removido de uma folha ***X*** e o número de elementos permanece maior ou igual a ***t - 1***, apenas remove-se o elemento.
- 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***.

Remover o valor 69

Removido 69

- 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.

Remover o valor 70

Removido 70

- 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.

Remover o valor 55

Removido 55

Referências

- Thomas H. Cormen, Charles E. Leiserson, Ronald Rivest, Clifford Stein, et al., *Introduction to Algorithms*, 2nd Edition, MIT Press, 2001.
- Bruno R. Preiss. Data Structures and Algorithms with Object-Oriented Design Patterns in C++. Disponível em: http://www.brpreiss.com/books/opus4/html/page340.html#SECTION00117000000000000000
- http://www.bluerwhite.org/btree/
- animação em Java: http://slady.net/java/bt/