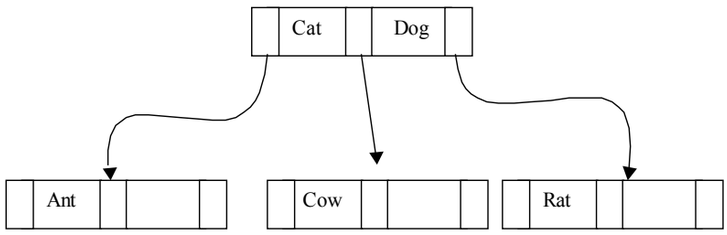
Árvores B

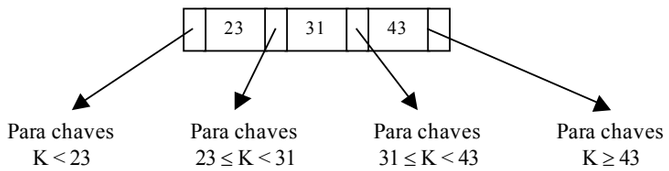
- Mantêm automaticamente tantos níveis de índices quantos sejam apropriados para o tamanho do arquivo que está sendo indexado.
- Gerenciam o espaço nos blocos que utilizam, de modo que cada bloco fique entre meio cheio e completamente cheio. Nenhum bloco de estouro é necessário para o índice.
- Organizam os blocos em uma árvore balanceada – os caminhos desde a raiz até uma folha têm o mesmo comprimento.
- Formadas por 3 partes: raiz, nós internos e folhas.
- Exemplo:



- Cada bloco terá espaço para n valores da chave (registros) de pesquisa e n+1 ponteiros.
- O número de registros que podem ser armazenados em um nó é dependente de sua capacidade.
- Cada nó de uma árvore B com ordem de capacidade *d* (*capacity order*) – ou simplesmente ordem - segue as seguintes propriedades:
  - $d \leq$ nº de chaves $\leq 2d$
    - Exceto a raiz que tem entre 1 a $2d$ chaves.
  - $d + 1 \leq$ nº de ponteiros $\leq 2d + 1$
    - Exceto a raiz que tem entre 2 a $2d + 1$ ponteiros.
  - Todos os nós folhas estão no mesmo nível.
- Cada nó, exceto a raiz, tem pelo menos 50% de utilização.
- Exemplo para nó com ordem 1. Terá uma ou duas chave e dois ou três ponteiros.

Ponteiro	Chave (Registro)	Ponteiro	Chave (Registro)	Ponteiro

- Os ponteiros de um nó apontam para nós filhos, no próximo nível da árvore.
- Se existem j chaves em um nó interno, existirão j+1 ponteiros. O primeiro ponteiro do nó indicará uma parte da árvore onde são encontradas chaves menores que k_1. O segundo ponteiro indica a parte da árvore com chaves pelo menos iguais a k_1, mas menores que k_2 e assim por diante. Finalmente, o (j+1)- ésimo ponteiro indica a parte da árvore com chaves maiores ou iguais a k_j



Inserção em árvore B

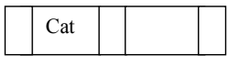
Algoritmo:

1. Navegar na árvore B procurando pela folha apropriada para inserir o novo registro. Em cada nó interno, realizar o seguinte teste se <, seguir à esquerda; se =, o registro já existe; se > e existe outro registro à direita, repetir a comparação anterior (< e = com o registro); senão seguir a

ramificação da direita (último registro à direita no nó).
/*Ao encontrar a folha*/
2. Se existir espaço disponível
   2.1. Inserir o registro (de forma ordenada).
3. Senão
   3.1. Enquanto existir overflow faça
      3.1.1. Se o nó com overflow é a raiz Então
         3.1.1.1.Criar novo nó raiz numa posição pai do nó raiz atual.
      3.1.2. Dividir o nó com overflow em dois nós.
      3.1.3. Subir o registro do meio para seu nó pai e posiciona-lo considerando o nó ordenado.
      3.1.4. Dividir os registros entre os dois novos nós. Os registros antes do registro do meio são posicionados no nó da esquerda e os demais no nó da direita.
      3.1.5. Ajustar os ponteiros do registro que subiu para o nó pai: o ponteiro à esquerda aponta para o nó filho com as chaves menores e o ponteiro da direita para o nó com as chaves maiores.

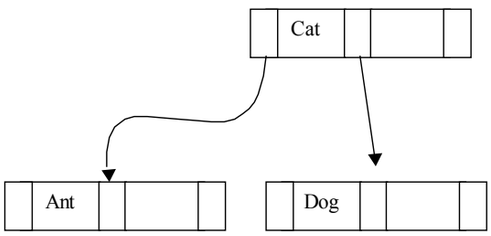
◆ Exemplo: Inserir os registros *cat (gato), ant (formiga), dog (cachorro), cow (vaca), rat (rato), pig (porco)* e *gnu (antílope)*  , em uma árvore com ordem *d* igual a 1.

- Inserir *cat*.

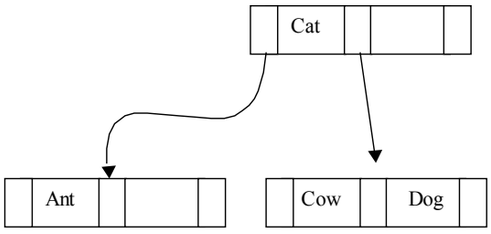


- Inserir *ant*.
  - Rearranjar os dois registros no nó para manter a ordenação.

- Inserir *dog*.
  - Ocorre overflow.
  - A chave do meio é *cat*, que sobe para o novo nó raiz. As demais chaves são posicionadas nos outros nós.
  - A altura da árvore é incrementada de 1.

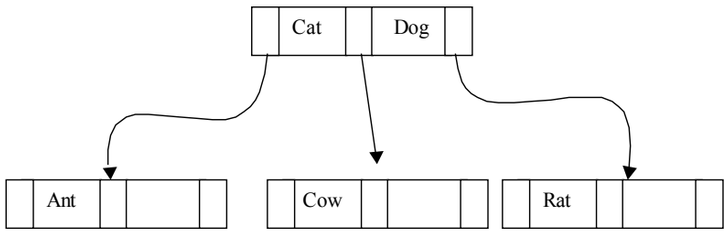


- Inserir *cow*.
  - Navegar na árvore até achar o nó folha (nó que contém *dog*).
  - Inserir o registro no nó, rearrumando-os.

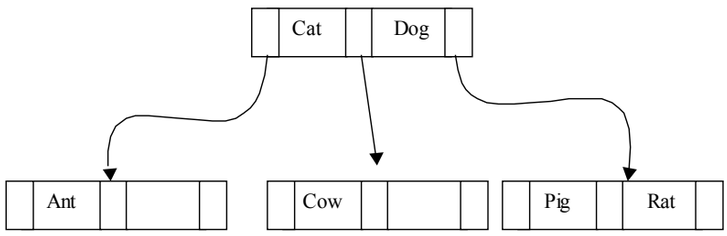


- Inserir *Rat*.
  - Navegar até o nó folha (aquele que contém *cow* e *dog*).
  - Como o nó está cheio é necessário dividi-lo.

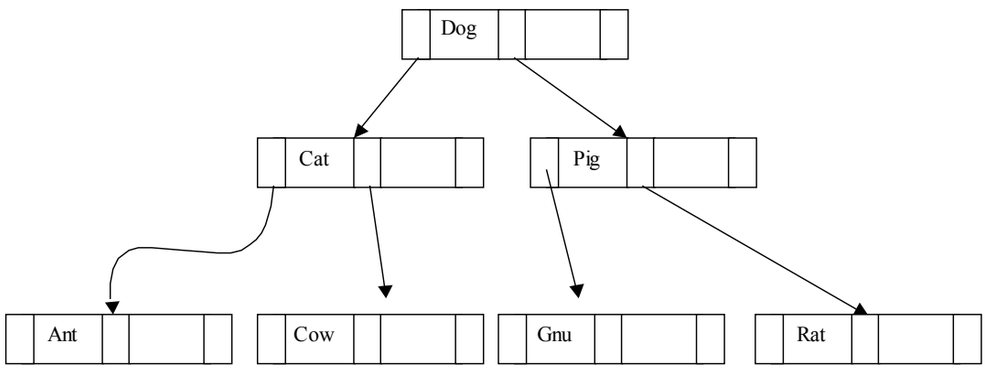
- O registro do meio tem a chave *dog*; assim ele é promovido para a raiz.
- O nó folha original é dividido em dois novos nós. Como há espaço na raiz, ela não é dividida.



➢ Inserir *Pig*.
- Navegar até o nó folha (aquele que contém *Rat*).
- Como há espaço disponível, inserir o registro.



➢ Inserir *Gnu*.
- Navegar até o nó folha (aquele que contém *Pig* e *Rat*).
- O nó está cheio, ele é então dividido.
- *Pig* é movido para o nó raiz.
- Nó raiz está cheio. Então ele é dividido em dois nós e um novo nó raiz é criado.



➢ São necessários 3 acessos para alcançar o registro *Gnu*.

Algumas características relevantes

◆ Fator de utilização de uma árvore B:

$$\frac{N^{o}\ de registros\ armazenados}{N^{o}\ de\ slots\ disponíveis}$$

➢ Para árvore do exemplo, o fator de utilização é igual a $\frac{7}{14}$, ou 50%.

◆ Uma árvore B é uma estrutura para armazenar grandes quantidades de informações- seus nós usualmente são armazenados em memória secundária.

◆ Uma busca em uma árvore B é definida como um acesso a um nó e não a um registro.

- Quando um nó é acessado, seu conteúdo é copiado para a memória. O tempo de pesquisa na memória principal é insignificante quando comparado ao tempo de acessar o nó na memória secundária.
- Como a árvore B é balanceada, no pior caso a busca tem número de acessos igual à altura da árvore, que especificado, para d > 1, como:

$Height \leq \log_{d+1} \frac{n+1}{2} + 1$, n é o número de registros armazenados na árvore B.

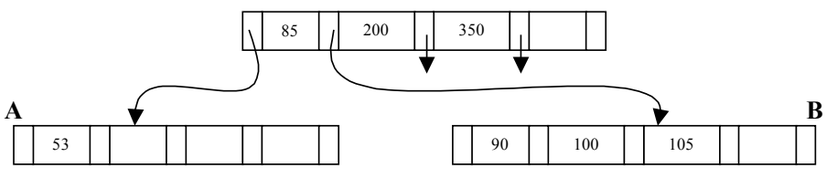
- Pior caso da busca tem complexidade $O(\log_d n)$

Exercício

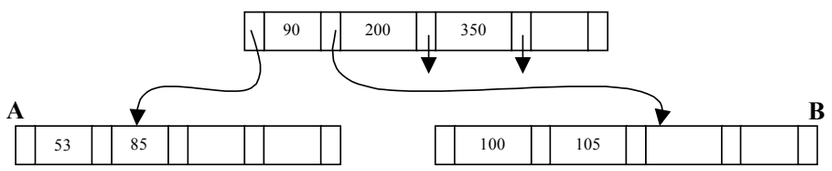
Em árvore B com ordem *d* igual a 2 inserir as seguintes chaves: 80, 50, 100, 90, 60, 65, 70, 75, 55, 64, 51, 76, 77, 78, 200, 300 e 150

Remoção

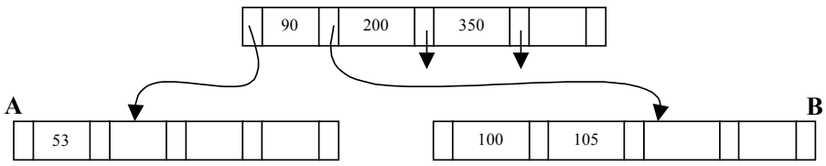
- É o oposto da inclusão.
- Após a remoção, as restrições da árvore B devem ser mantidas.
- Se o registro a ser removido está em um nó folha e a capacidade mínima é mantida, então nenhum nó interno é afetado.
- Se o registro a ser removido está em um nó não folha, é necessário mover um registro do nó folha para a posição do registro removido para manter uma chave a ser comparada durante a busca.
  - O registro sucessor com chave mais à esquerda substitui o registro sendo removido.
- Observe que as árvores estão sendo usadas como estrutura de armazenamento e estrutura de índice.
- Quando a restrição de capacidade mínima é violada quando um registro é removido de um nó folha ou movido para um nó interno devido a deleção neste nó, a restrição de capacidade é mantida da seguinte forma:
  - Redistribuindo os registros do nó folha e de um dos seus nós irmãos; ou,
  - Mesclar um nó folha com um nó irmão.
- Exemplos Árvore com ordem 2.
  - Exemplo 1: Existir irmão para emprestar registro.



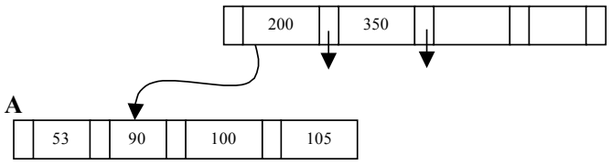
  - Nó A viola a restrição de capacidade mínima.
  - Como nó B tem mais registros do que a capacidade mínima, o nó A pode pedir emprestado registro ao nó B.
  - O nó de comparação do nó pai é substituído, com o objetivo de manter as características de comparações para a pesquisa.



  - Exemplo 2: Não existir nó para redistribuir registros



- Nó A viola a restrição de capacidade mínima e B não tem registro para emprestar.
- Nós A e B devem ser mesclados, juntamente com o registro de comparação no nó pai.



- A mesclagem de registros pode ser propagada até o nó raiz.
- Se o nó raiz tem apenas 1 registro e este é removido, o nó raiz não é mais necessário e o nível da árvore decresce de 1.