Árvore B#

- Devido à restrição de capacidade mínima, uma árvore B apresenta 50% como menor utilização do espaço para todos os nós, exceto a raiz. A utilização máxima é de 100%, mas isto geralmente não ocorre com dados randômicos.
- Como melhorar a utilização do espaço de armazenamento utilizando uma árvore B?
  - Adiar a divisão de nós (*splitting*) durante a inserção pois isto reduz a utilização do espaço de armazenamento.
  - A divisão também aumenta a altura da árvore, o que afeta negativamente a performance da busca.
- As árvores B* e B# adiam a divisão de nós, consequentemente a árvore tem altura menor do que a árvore B correspondente.
- A principal diferença entre a árvore B# e a árvore B* é que o nó raiz da árvore B* é maior do que os outros nós da árvore e pode conter até $4d+1$ registros, onde d é *ordem* (de capacidade) da árvore.
- Todos os nós da árvore B# têm o mesmo tamanho, sendo essencialmente uma árvore B com um algoritmo de inserção modificado.
- Características principais de uma árvore B#
  - Adiar a divisão (*splitting*) pela redistribuição de registros do nó que sofre *overflow* entre os nós irmãos adjacentes; e
  - Quando a divisão é necessária, converter dois nós em 3 nós.
- A árvore B# possui melhor utilização do espaço e busca mais rápida do que a árvore B.

Algoritmo de Inserção em árvore B#

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 ENTÃO
   2.1. Inserir o registro (de forma ordenada).
3. SENÃO
   3.1. ENQUANTO existir overflow em um nível que não seja o da raiz FAÇA
      3.1.1. SE existe espaço disponível no irmão à direita ou no irmão da esquerda ENTÃO
         3.1.1.1. Redistribuir os registros, incluindo o registro que causou overflow, entre o nó que sofreu overflow e seu irmão com espaço disponível.
      3.1.2. SENÃO
         3.1.2.1. Dividir o nó com *overflow* em 3 nós e redistribuir os seguintes registros
            (1) seus registros, incluindo o registro que causou overflow,
            (2) os registros do seu irmão à direita (ou se não existir, os registros do seu irmão à esquerda), e
            (3) o registro de comparação no nó pai
         3.1.2.2. Ajustar os 3 ponteiros no nó pai
   3.2. SE existe *overflow* no nó raiz ENTÃO
      3.2.1. Criar novo nó raiz numa posição pai do nó raiz atual.
      3.2.2. Dividir o nó com overflow em dois nós.
      3.2.3. Subir o registro do meio para seu nó pai e posiciona-lo considerando o nó ordenado.
      3.2.4. Dividir os registros restantes entre os dois nós. Os registros antes do registro do meio são posicionados no nó da esquerda e os demais no nó da direita.
      3.2.5. 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.

- A seguinte fórmula geral pode ser utilizada para determinar a posição do registro de comparação (que será posicionado no nó pai).

  Posição do registro de comparação = $\left\lfloor \frac{(r+1)}{n} \right\rfloor$, onde *r* é o número de registros a serem redistribuídos e *n* é o número de o número de nós folhas nos quais os registros estão sendo redistribuídos.

  Exemplo: Redistribuir 8 registros em 2 nós → $\left\lfloor \frac{(9)}{2} \right\rfloor = 4$

Exercício

Em uma 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

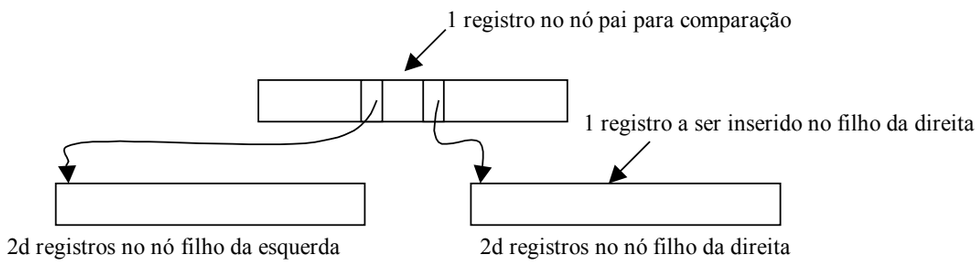
Definição e Análise

- A definição formal de uma árvore B# com ordem de capacidade *d* é a mesma de uma árvore B de capacidade *d* exceto os limites para as chaves e os ponteiros dos nós dos níveis intermediários (aqueles que não incluem a raiz, as folhas e os nós no nível imediatamente próximo à raiz) que são maiores.

$$\frac{4d-2}{3} \le chaves \le 2d$$

$$\frac{4d+1}{3} \le ponteiros \le 2d+1$$

- Como estes limites foram obtidos?
  - Quando a divisão (*split*) é necessário, a porção da árvore B# afetada tem a seguinte estrutura.



  - O número total de registros a ser redistribuído é igual a 4*d*+2. Já que dois nós estão sendo divididos em 3, nós necessitamos de dois registros do nó pai para comparações. Isto faz com que 4*d* registros sejam distribuídos em 3 nós. Cada nó tem pelo menos $\left\lfloor \frac{4d}{3} \right\rfloor$ registros.
  - Removendo a função "piso" obtém-se:
    $\frac{4d}{3} - \frac{2}{3} = \frac{4d-2}{3}$ registros para o limite de capacidade mínima.
  - Como cada nó tem um ponteiro a mais do que o número de registros, então um nó tem pelo menos:
    $\frac{4d-2}{3} + 1 = \frac{4d-2+3}{3} = \frac{4d+1}{3}$ ponteiros.
  - Percentual de utilização:

$$\frac{\frac{4d-2}{3}}{2d} = \frac{4d-2}{6d}$$ ➔ para *d* muito grande o temo –2 torna-se insignificante resultado num percentual de aproximadamente $\frac{2}{3}$