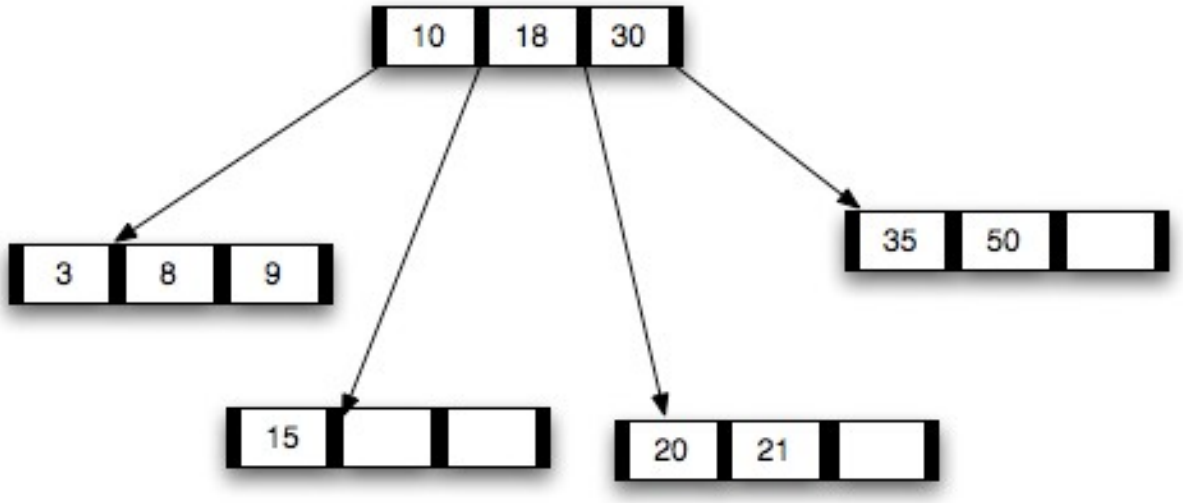
ACH2024

Aula 16 – Árvores B (remoção)

Árvore B - Definição

- Uma *árvore* B é uma árvore com as seguintes propriedades:

  1. Cada nó x contém os seguintes campos:
     - $n[x]$, o número de chaves atualmente armazenadas no nó x;
     - as $n[x]$ chaves, armazenadas em ordem não decrescente, de modo que $key_1[x] \leq key_2[x] \leq \ldots \leq key_{n[x]}[x]$;
     - $leaf[x]$, um valor booleano indicando se x é uma folha (TRUE) ou um nó interno (FALSE).
     - se x é um nó interno, x contém $n[x] + 1$ ponteiros $c_1[x], c_2[x], \ldots c_{n[x]+1}[x]$ para seus filhos.

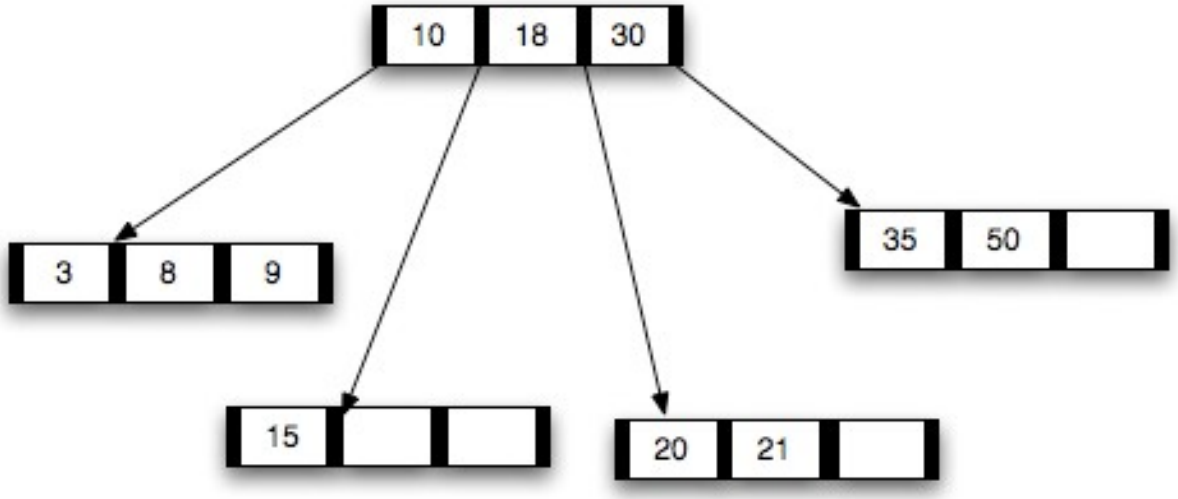


Árvore B - Definição

2. As chaves $key_i[x]$ separam as faixas de valores armazenados em cada subárvore: denotando por k_i uma chave qualquer armazenada na subárvore com nó $c_i[x]$, tem-se

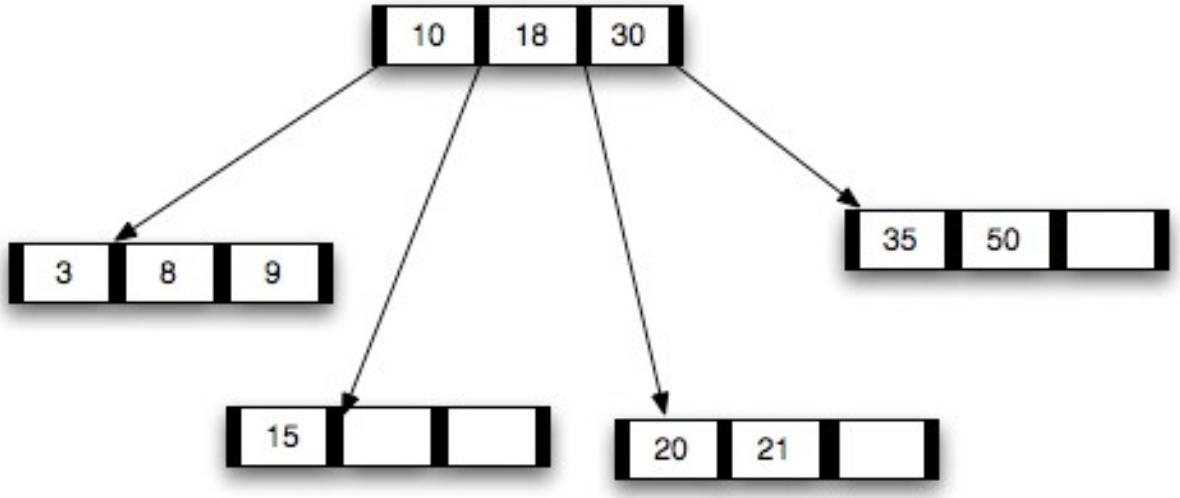
$$k_1 \leq key_1[x] \leq k_2 \leq key_2[x] \leq \ldots \leq key_{n[x]}[x] \leq k_{n[x]+1}$$

3. Todas as folhas aparecem no mesmo nível, que é a altura da árvore, h.



Árvore B - Definição

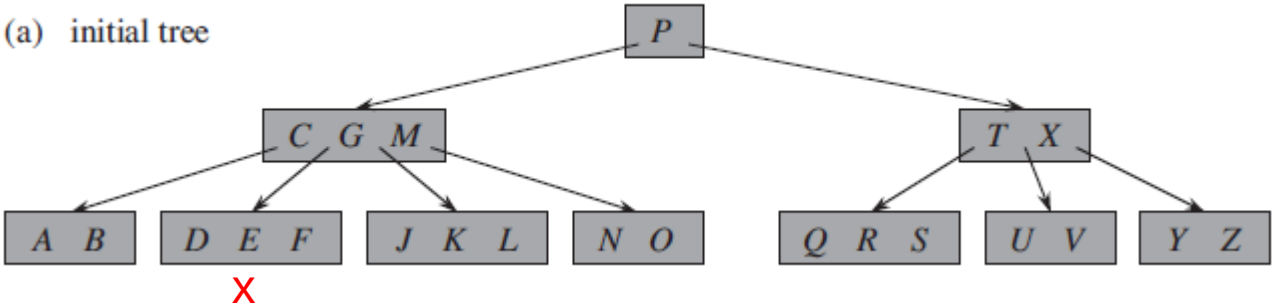
4. Há um limite inferior e superior no número de chaves que um nó pode conter, expressos em termos de um inteiro fixo $t \geq 2$ chamado o *grau mínimo* (ou *ordem*) da árvore.
   - Todo nó que não seja a raiz deve conter pelo menos $t - 1$ chaves. Todo nó interno que não seja a raiz deve conter pelo menos t filhos.
   - Todo nó deve conter no máximo $2t - 1$ chaves (e portanto todo nó interno deve ter no máximo $2t$ filhos). Dizemos que um nó está *cheio* se ele contiver exatamente $2t - 1$ chaves



Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

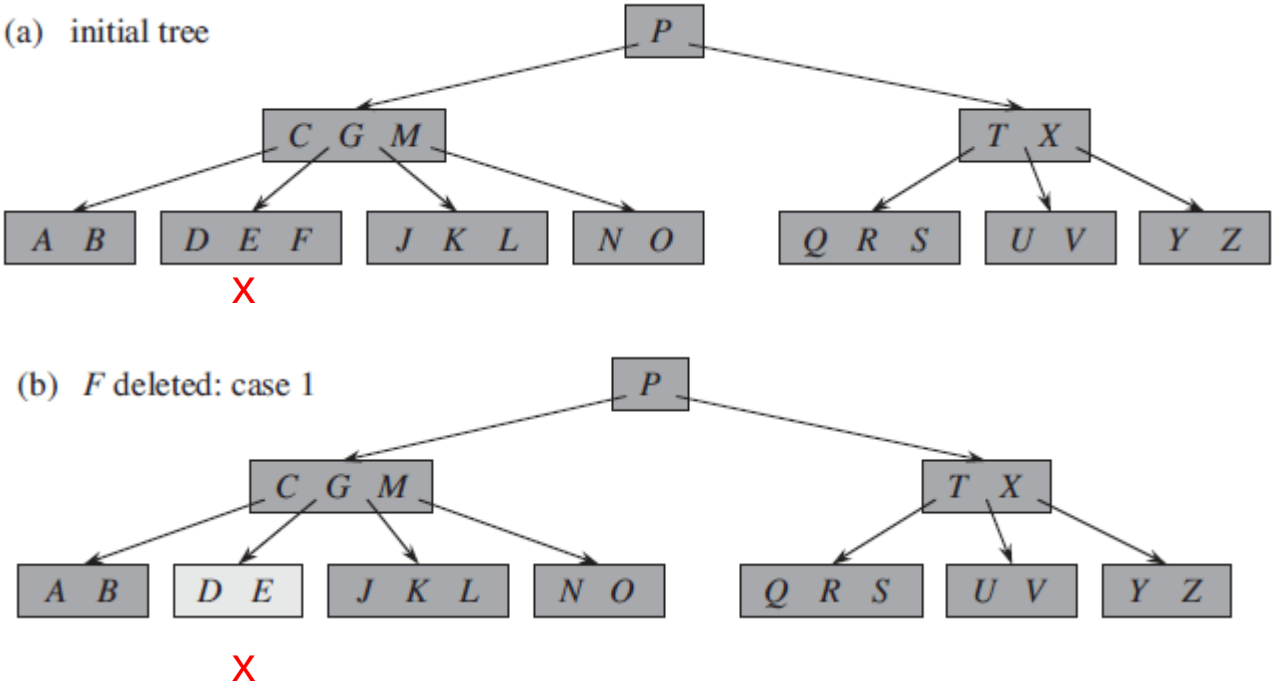
1. Se a chave k está no nó x e x é uma folha,



Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

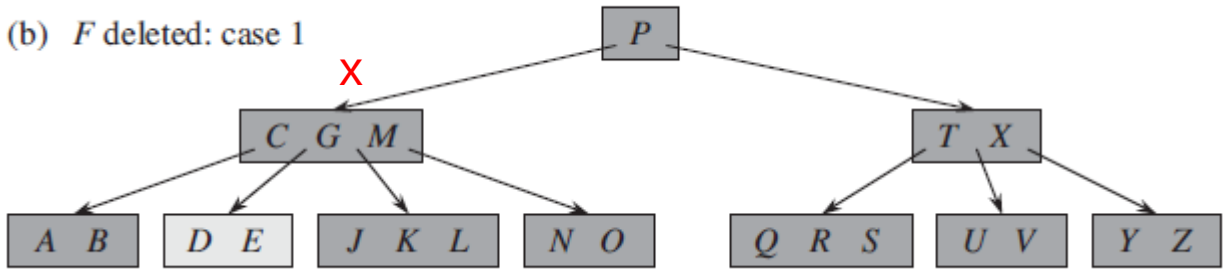
1. Se a chave k está no nó x e x é uma folha, exclua a chave k de x.



Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  2. Se a chave k está no nó x e x é um nó interno, faça:



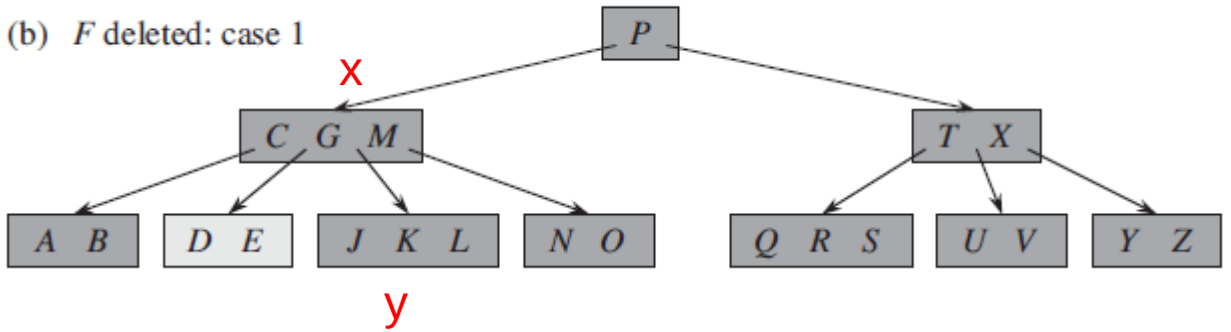
(c) *M* deleted: case 2a

Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  2. Se a chave k está no nó x e x é um nó interno, faça:

     a) Se o filho y que precede k no nó x tem pelo menos t chaves, então

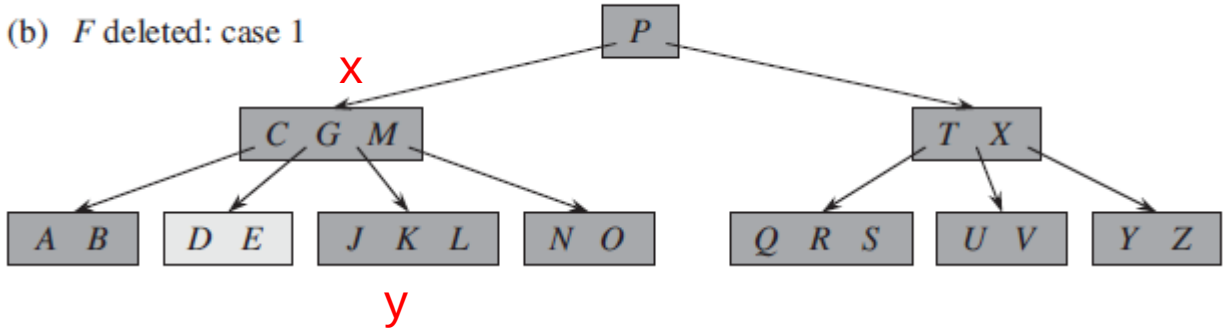


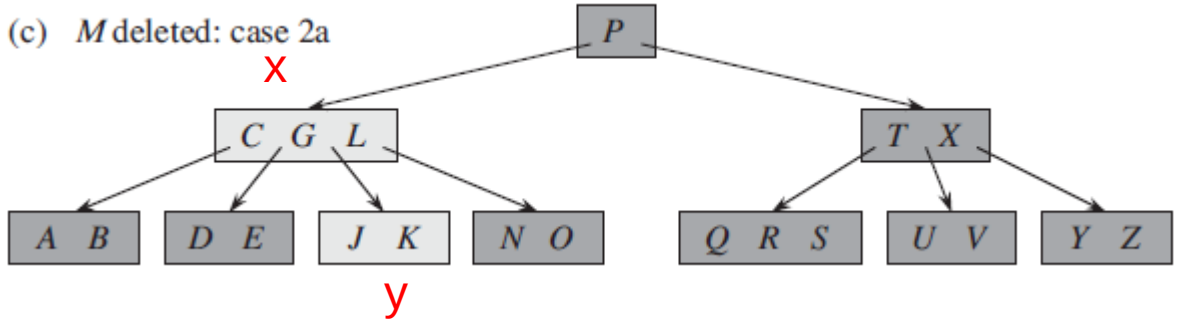
Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  2. Se a chave k está no nó x e x é um nó interno, faça:

     a) Se o filho y que precede k no nó x tem pelo menos t chaves, então encontre o predecessor k' de k na subárvore com raiz y. Delete recursivamente k', e substitua k por k' em x.





Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  2. Se a chave k está no nó x e x é um nó interno, faça:

     a) Se o filho y que precede k no nó x tem pelo menos t chaves, então encontre o predecessor k' de k na subárvore com raiz y. Delete recursivamente k', e substitua k por k' em x.

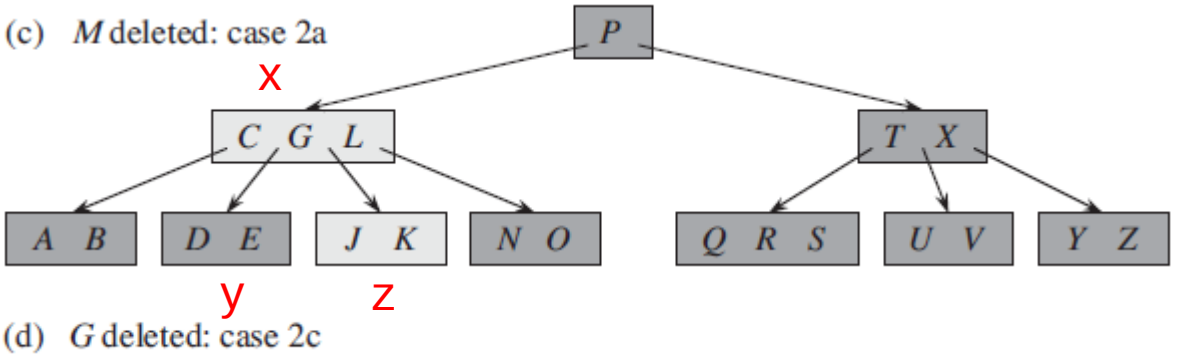
     b) Simetricamente, se o filho z imediatamente após k no nó x tem pelo menos t chaves, então encontre o sucessor k' de k na subárvore com raiz z. Delete recursivamente k', e substitua k por k' em x.

Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  2. Se a chave k está no nó x e x é um nó interno, faça:

     c) Caso contrário, se ambos y e z possuem apenas $t - 1$ chaves, faça a

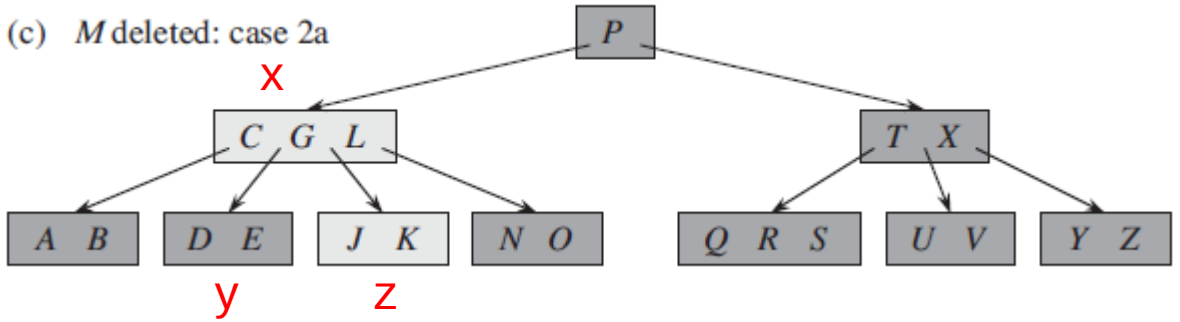


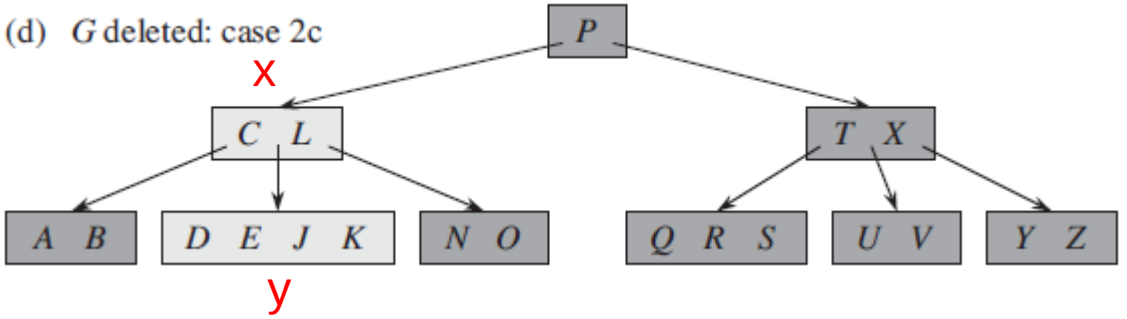
Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  2. Se a chave k está no nó x e x é um nó interno, faça:

     c) Caso contrário, se ambos y e z possuem apenas $t-1$ chaves, faça a junção de k e todas as chaves de z em y, de forma que x perde tanto a chave k como o ponteiro para z, e y agora contém $2t-1$ chaves. Então, libere z e delete recursivamente k de y.

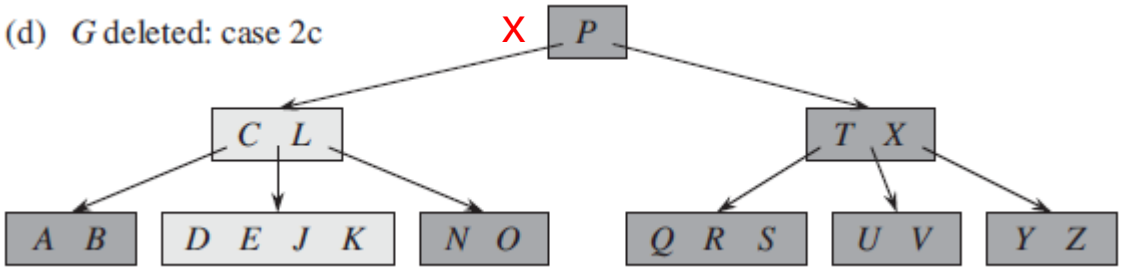




Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  3. Se a chave k não está presente no no interno x,



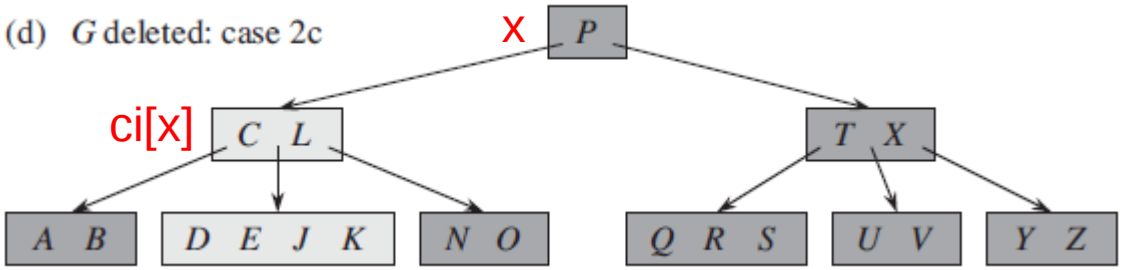
(e) D deleted:

Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  3. Se a chave k não está presente no no interno x, determine a raiz $c_i[x]$ da subárvore apropriada que deve conter k (se k estiver presente na árvore).

Algum problema?

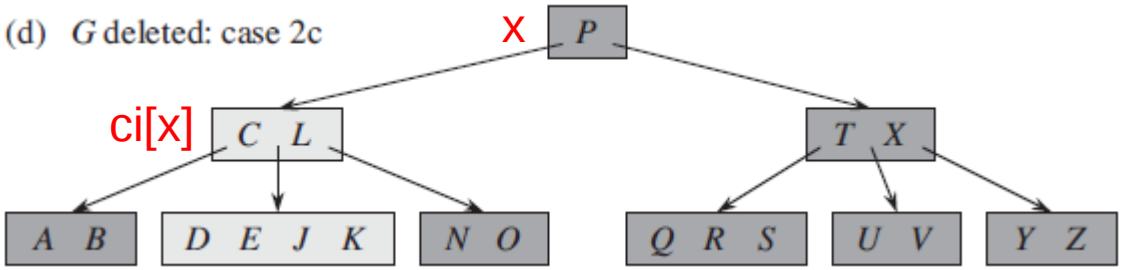


(e) *D* deleted:

Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  3. Se a chave k não está presente no no interno x, determine a raiz $c_i[x]$ da subárvore apropriada que deve conter k (se k estiver presente na árvore). Se $c_i[x]$ tem apenas $t - 1$ chaves,

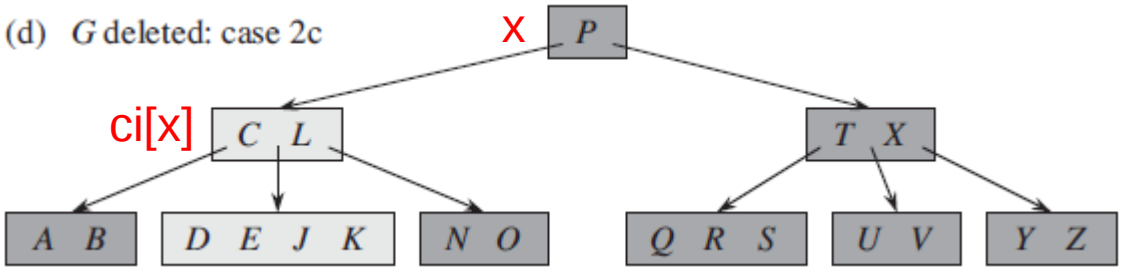


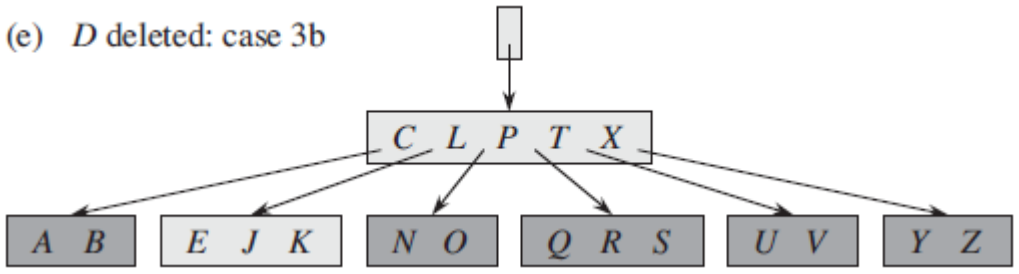
(e) D deleted:

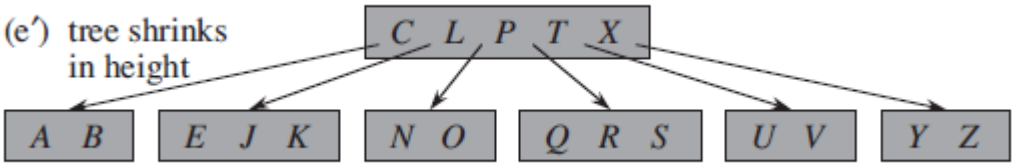
Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  3. Se a chave k não está presente no no interno x, determine a raiz $c_i[x]$ da subárvore apropriada que deve conter k (se k estiver presente na árvore). Se $c_i[x]$ tem apenas $t - 1$ chaves,



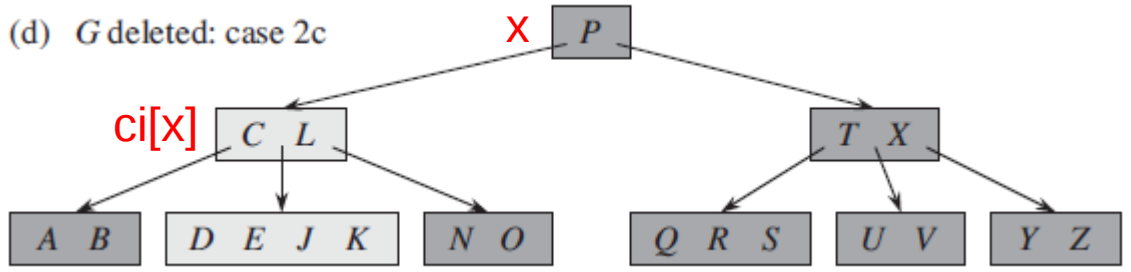


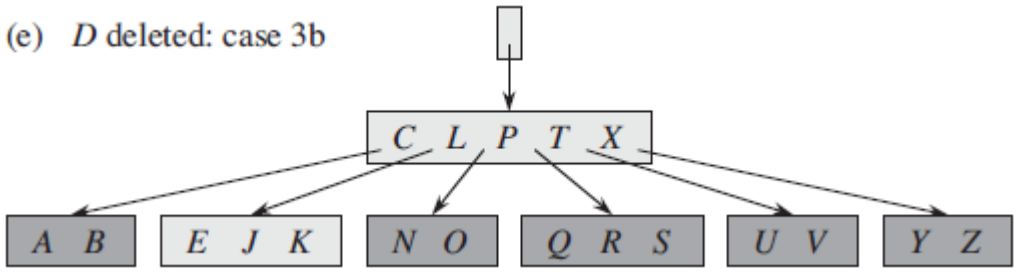


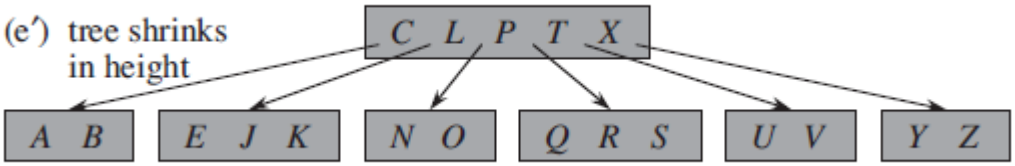
Remoção em árvores B

- B-Tree-Delete(x,k): remoção da chave k da subárvore com raiz x.

  3. Se a chave k não está presente no no interno x, determine a raiz $c_i[x]$ da subárvore apropriada que deve conter k (se k estiver presente na árvore). Se $c_i[x]$ tem apenas $t-1$ chaves, execute o passo 3a ou 3b conforme necessário para garantir que o algoritmo desça para um nó contendo pelo menos t chaves. Então, continue no filho apropriado de x.

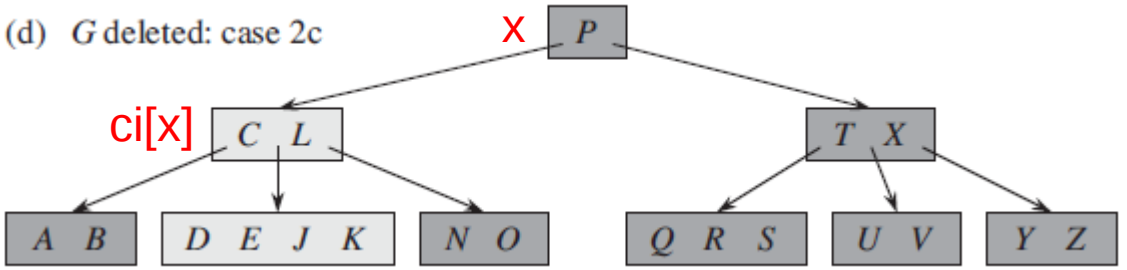


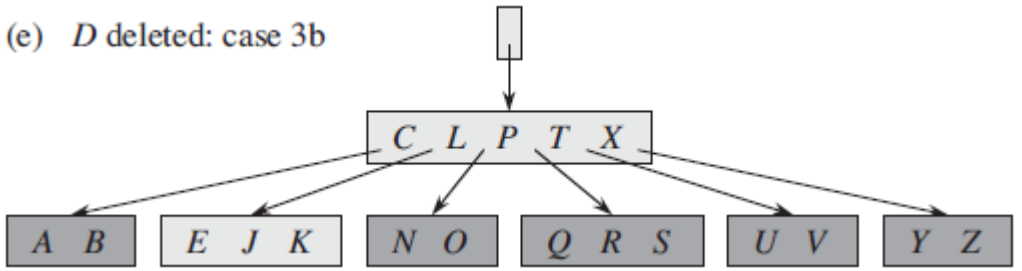


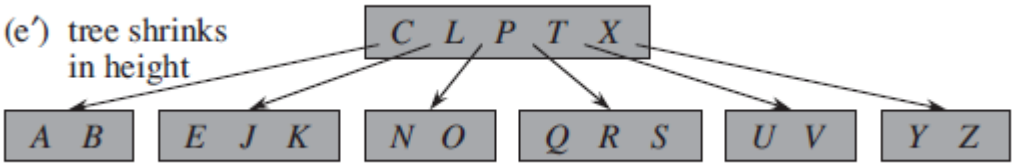


a)

b) Se $c_i[x]$ e ambos os irmãos imediatos de $c_i[x]$ contêm $t - 1$ chaves, faça a junção de $c_i[x]$ com um de seus irmãos. Isso implicará em mover uma chave de x para o novo nó fundido (que se tornará a chave mediana para aquele nó).

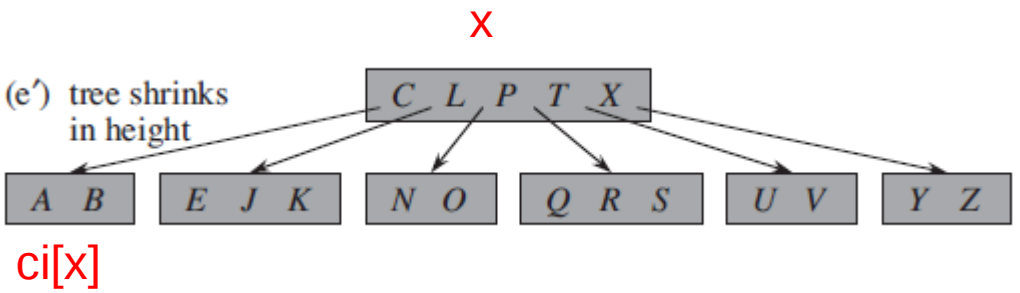


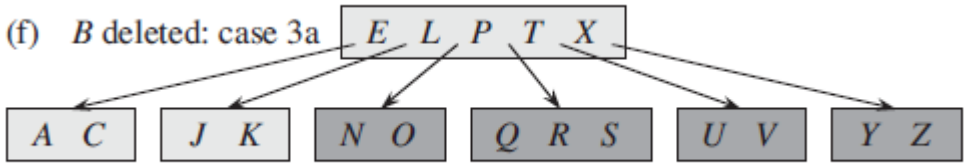




a) Se $c_i[x]$ contém apenas $t-1$ chaves mas tem um irmão imediato com pelo menos t chaves, dê para $c_i[x]$ uma chave extra movendo uma chave de x para $c_i[x]$, movendo uma chave do irmão imediato de $c_i[x]$ à esquerda ou à direita, e movendo o ponteiro do filho apropriado do irmão para o nó $c_i[x]$.

b)





Remoção em árvores B

- Atenção quando um dos nós for a raiz:
  - Ela pode ter menos do que t-1 chaves
  - Se ficar com zero chaves precisa desalocar o bloco e atualizar quem é a nova raiz.
  - Precisa de uma camada extra sobre a chamada da deleção:

Remoção em árvores B

```
B-Tree-Delete-From-Root(T, k){
        r ← raiz[T];
        se (n[r] = 0) retorna;
        senão B-Tree-Delete(r,k);
        se (n[r] = 0  E  (! leave[r]){
                raiz[T] ← c1[r];
                desaloca(r);
        }
}
```

Remoção em árvores B

IMPLEMENTEM EM CASA!!!!

É QUASE CERTO QUE PRECISEM

PARA O EP 2 !!!!!!!!!!

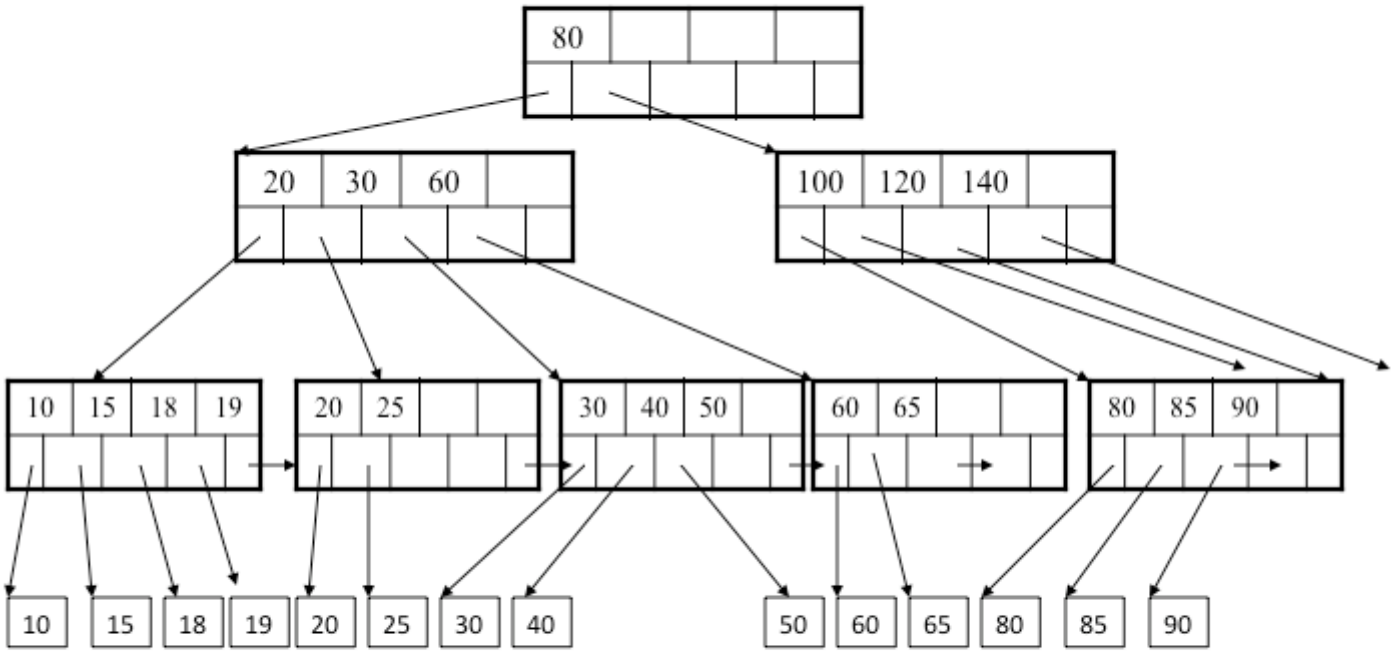
Comentários sobre árvores B+

ÁRVORES B+

Variação da árvore B, na qual:

- os nós internos armazenam apenas os índices (ponteiros de filhos e chaves)
- as folhas armazenam os registros de dados (conectadas da esquerda para a direita, permitindo acesso sequencial mais eficiente)

- Blocagem maior (cabem mais registros nos nós internos → altura menor)



Nós internos (de índices)

Nós folhas (de dados)

Comentários sobre árvores B+

Adaptações dos algoritmos:

- Busca: tem sempre que descer às folhas
- Inserção: só nas folhas
  - Split :
    - mediana de uma folha é COPIADA para o pai
    - Mediada de um nó interno é MOVIDO para o pai
- Remoção: nas folhas
  - Se k (chave a ser removida) ocorrer em um nó interno, o valor deve ser substituído pelo valor à esquerda na folha

Outras variações

- Árvores B*: preenchimento mínimo de 2/3