Árvores-B (Parte II)

Leandro C. Cintra

M.C.F. de Oliveira

Fonte: Folk & Zoelick, File Structures

Histórico

A invenção da B-*tree*

- Bayer and McGreight, 1972, publicaram o artigo: "***Organization and Maintenance of Large Ordered Indexes*".**

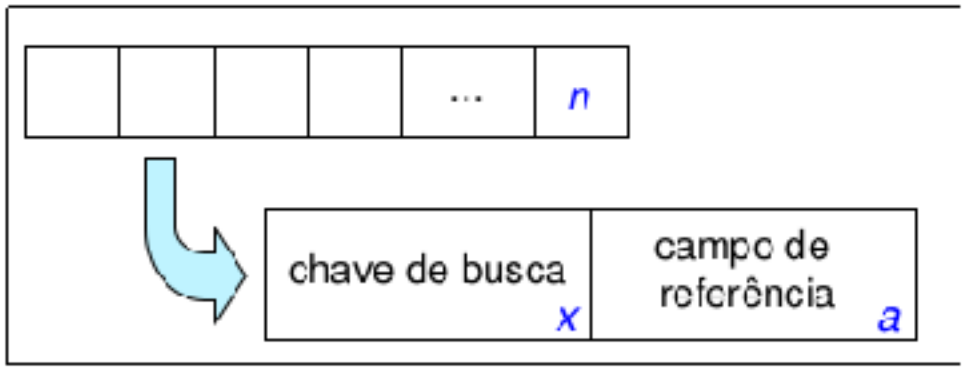
- Em 1979, o uso de árvores-B já era praticamente o padrão adotado em sistemas de arquivos de propósito geral para a manutenção de índices para bases de dados.

De onde vem o ‘B’ ?

- Árvores B são uma generalização das árvores binárias de busca, daí o termo ‘B’.

Característica Geral

- Organizar e manter um índice para um arquivo de acesso aleatório altamente dinâmico, em disco
- Índice
  - N elementos (x, a) de tamanho fixo



Característica Geral

- Índice
  - Extremamente volumoso
- O pool de buffers é pequeno
  - Apenas uma parcela do índice pode ser carregada em memória principal
  - Operações, portanto, são baseadas em disco

Construção de árvores-B

Construção *Top-Down* de árvores paginadas

- É simples construir uma árvore paginada se temos todo o conjunto de chaves antes de iniciar a construção
  - Inicia-se pela chave do meio para obter uma árvore balanceada
- Porém, é complicado se estamos recebendo as chaves em uma seqüência aleatória

Construção *Top-Down* de árvores paginadas

- **Ordem:** C S D T A M P I B W N G U R K E H O L J Y Q Z F X V

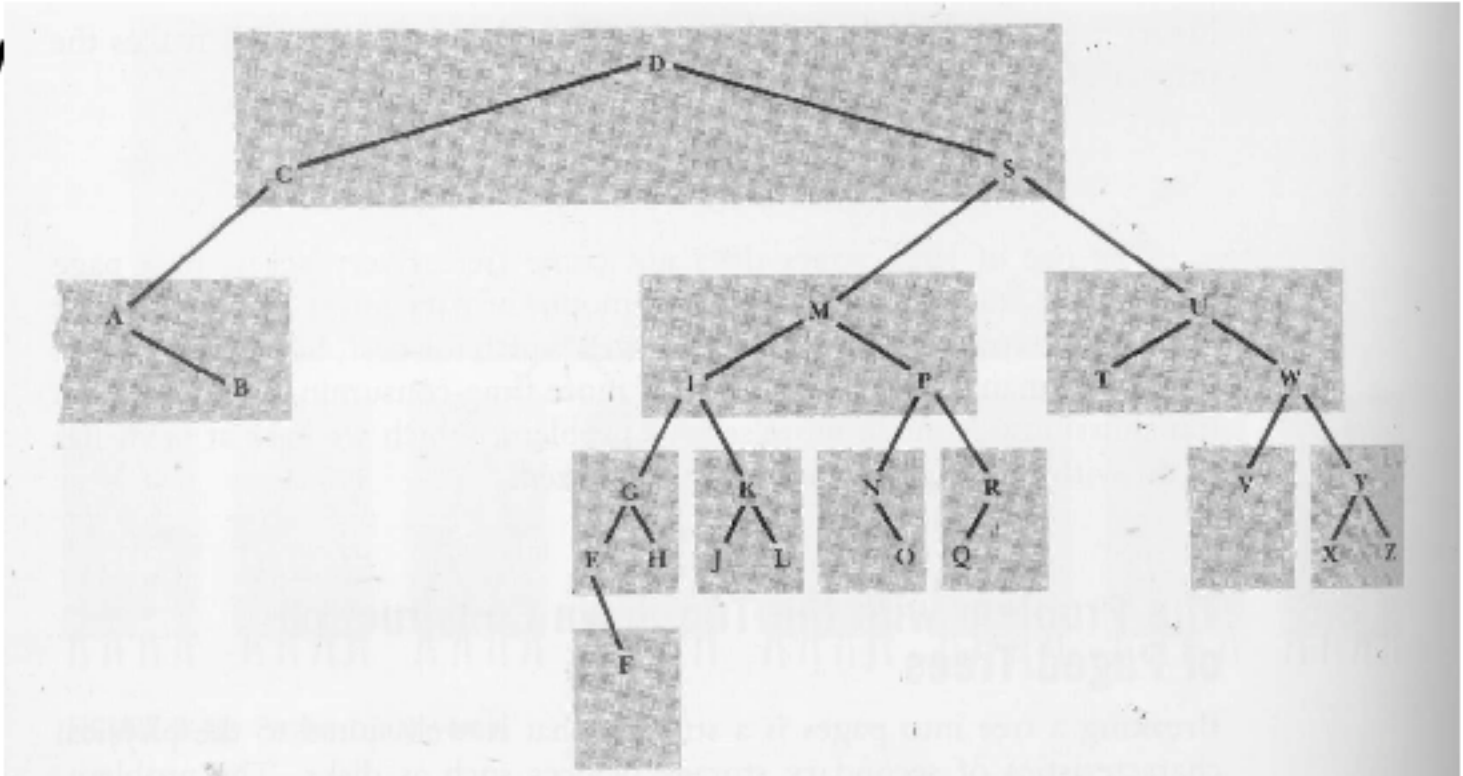


FIGURE 8.13 Paged tree constructed from keys arriving in random input sequence.

Construção *Top-Down* de árvores paginadas

- Na figura anterior, a construção foi feita *top-down*, a partir da raiz
- cada vez que uma chave é inserida a árvore dentro da página sofre uma rotação, sempre que necessário, para manter o balanceamento
- A construção a partir da raiz implica em que as chaves iniciais estarão necessariamente na raiz
- C e D não deveriam estar no topo, pois acabam desbalanceando a árvore de forma definitiva
- Esta árvore não está muito ruim, mas o que aconteceria se as chaves fossem fornecidas em ordem alfabética?

Construção *Top-Down* de árvores paginadas

- **Questões:**
  - como garantir que as chaves na página raiz são boas separadoras, i.e., dividem o conjunto de chaves de maneira balanceada ?
  - como impedir o agrupamento de chaves que não deveriam estar na mesma página (como C, D e S, por exemplo)
  - como garantir que cada página contenha um número mínimo de chaves ?

Construção *Bottom-Up*: Árvores-B

- Bayer e McCreight propuseram que as árvores fossem construídas de baixo para cima *(Bottom-Up)*
- Desta forma, as chaves na raiz da árvore emergem naturalmente
- Cada página é formada por uma seqüência ordenada de chaves e um conjunto de ponteiros
- Não existe uma árvore explícita dentro de uma página (ou nó da árvore)

Construção *Bottom-Up*: Árvores-B

FIGURE 8.14 Initial leaf of a B-tree with a page size of seven.

FIGURE 8.15 Splitting the leaf to accommodate the new *J* key.

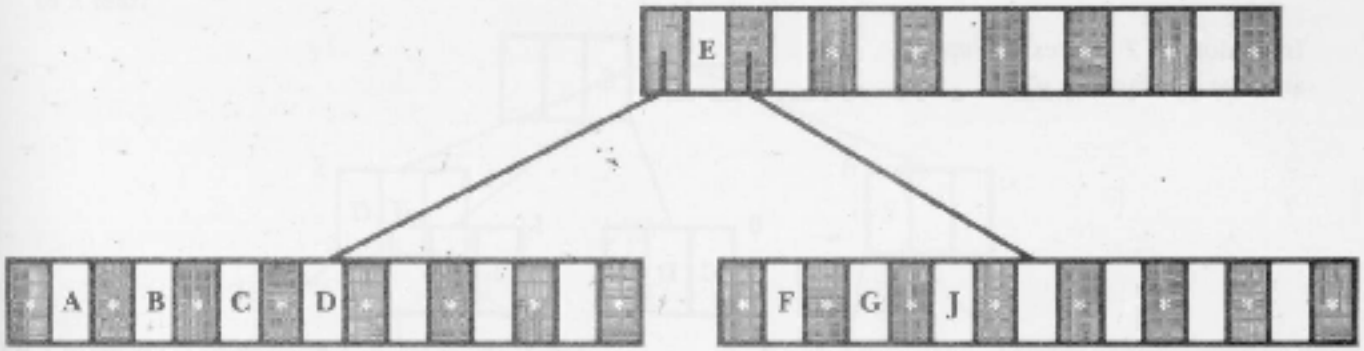


FIGURE 8.16 Promotion of the *E* key into a root node.

Construção *Bottom-Up*: Árvores-B

- O número de ponteiros em um nó excede o número de chaves em 1
- O número máximo de ponteiros que podem ser armazenados em um nó é a **ordem** da árvore
- O número máximo de ponteiros é igual ao número máximo de descendentes de um nó
- **Exemplo:** uma árvore-B de ordem 8 possui nós com, no máximo, 7 chaves e 8 filhos
- Os nós folha não possuem filhos, e seus ponteiros são nulos

Splitting (sub-divisão)

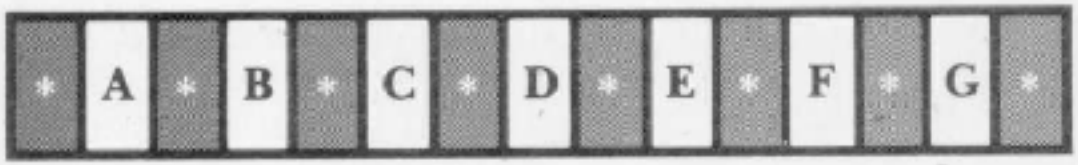
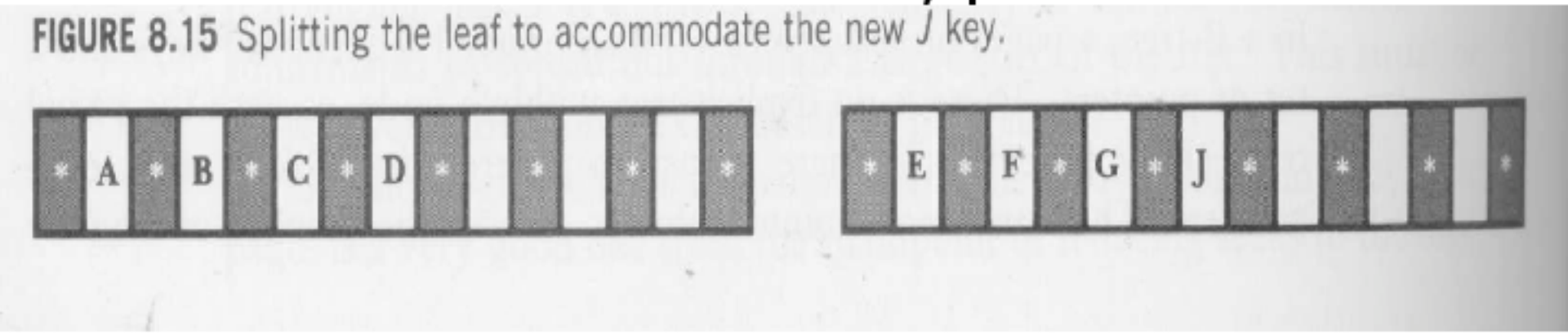


FIGURE 8.14 Initial leaf of a B-tree with a page size of seven.

- Esta folha que, coincidentemente, é também a raiz da árvore, está cheia
- **Como inserir uma nova chave, digamos J ?**

Splitting (sub-divisão)

- Sub-dividimos (*split*) o nó folha em dois nós folhas, distribuindo as chaves igualmente entre os dois nós
- Temos duas folhas, precisamos de



Promoting (promoção)

- Criamos a nova raiz "promovendo", ou "subindo", uma das chaves que estão nos limites de separação das folhas
- Promoção da chave E

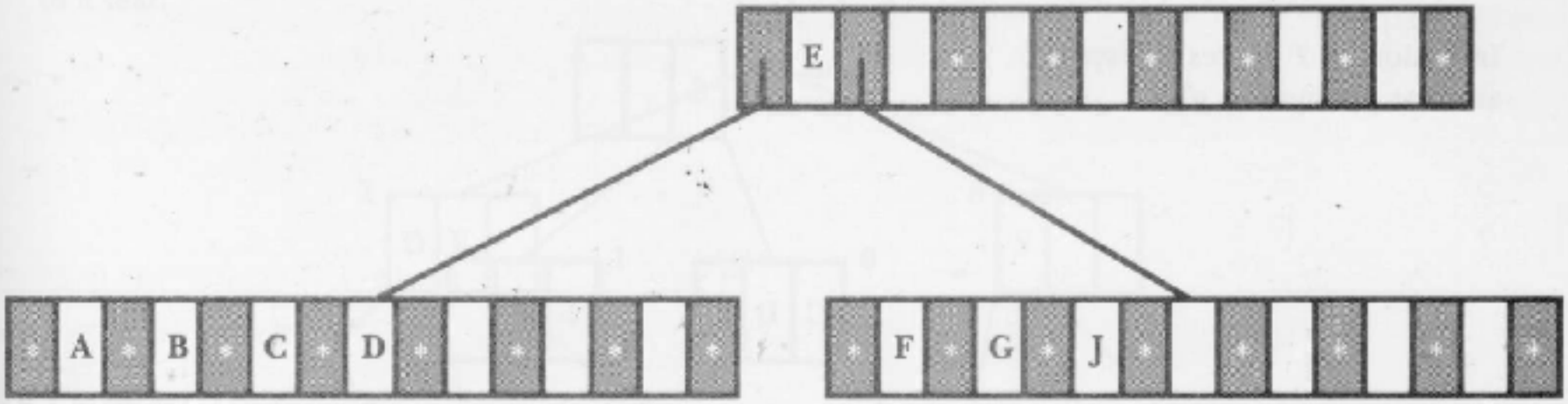


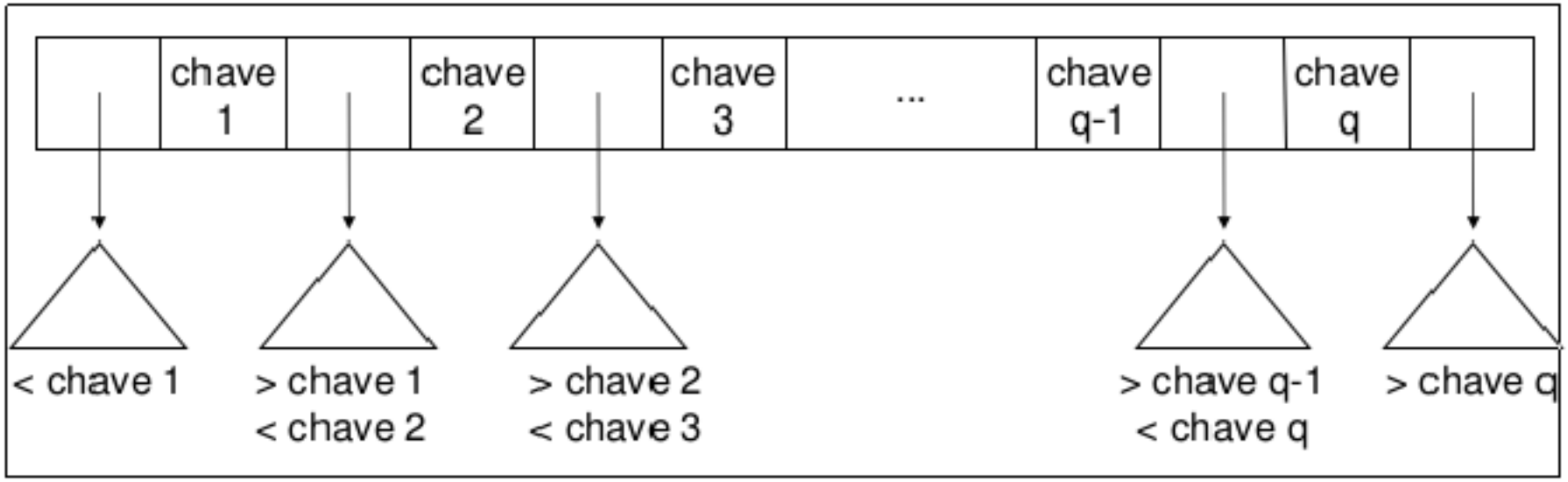
FIGURE 8.16 Promotion of the *E* key into a root node.

Propriedades

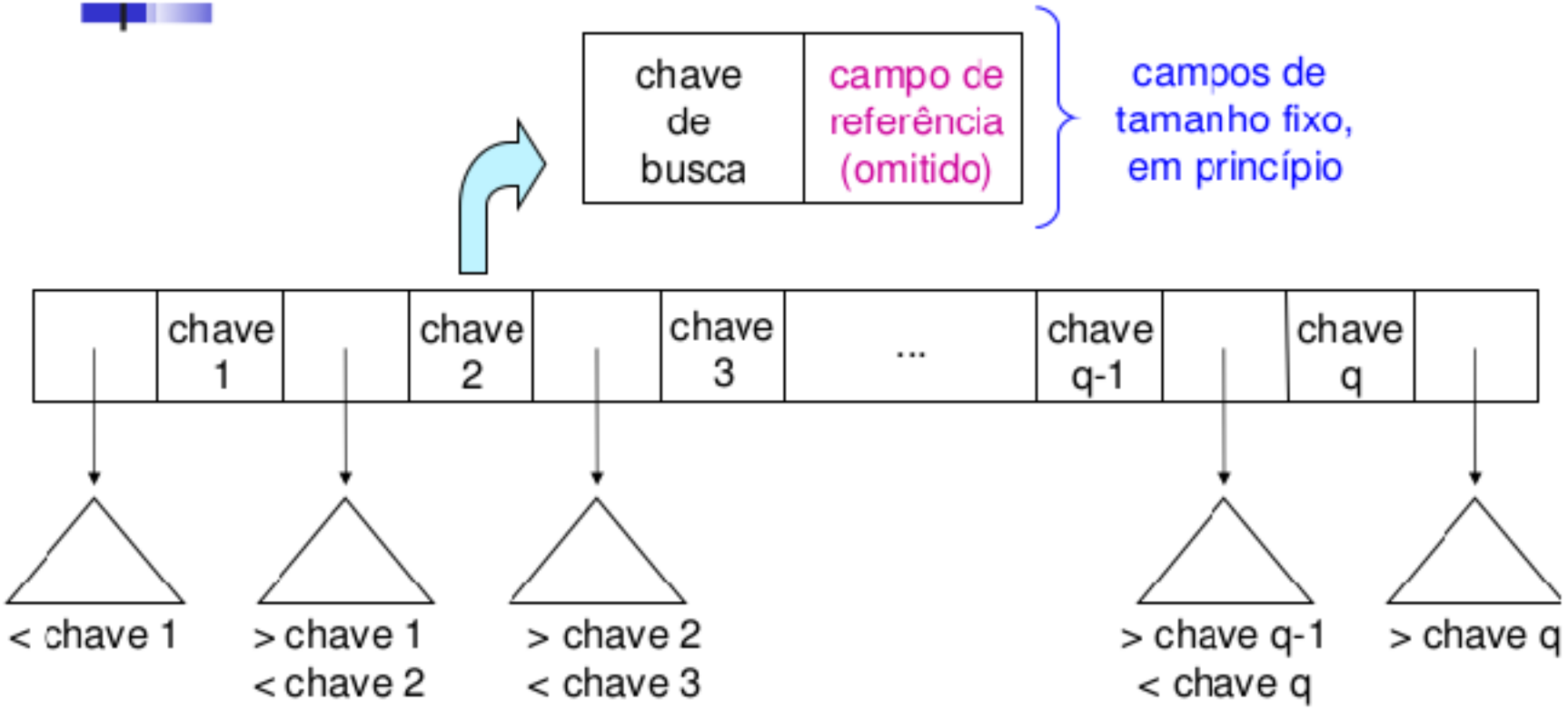
Uma árvoreB é nária pois possui mais de 2 descendentes por nó (página). Numa árvoreB de ordem m

1. Cada página tem:
    - no máximo, m descendentes e m-1 registros
    - no mínimo $\lceil m/2 \rceil$ descendentes (exceto raiz e folhas)
2. A raiz tem, no mínimo, dois descendentes
    - a menos que seja uma folha
3. Todas as folhas estão no mesmo nível
4. Uma página não folha com k descendentes contém k-1 chaves
5. Uma página folha contém, no mínimo $\lceil m/2 \rceil$ -1 e, no máximo, m-1 chaves

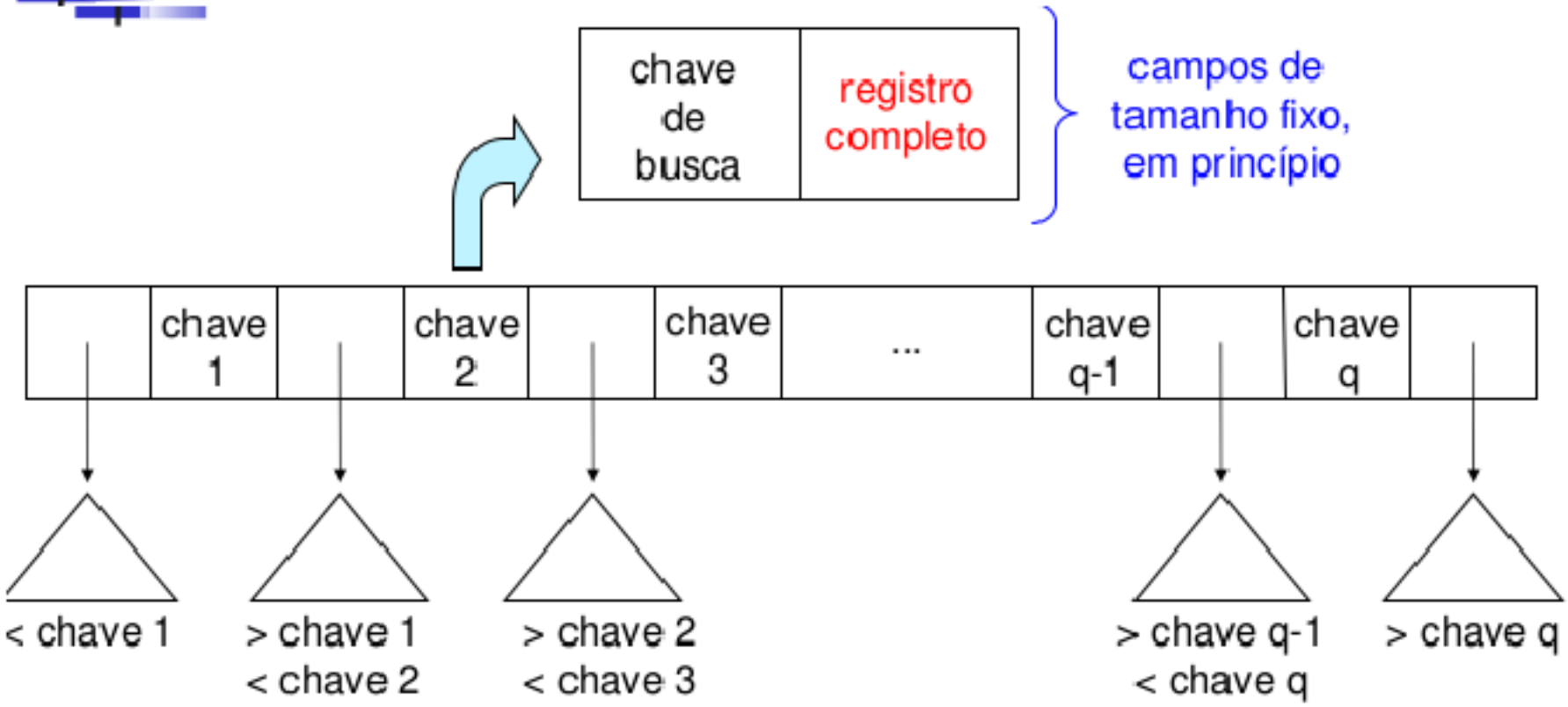
Estrutura lógica de um nó



Estrutura lógica de um nó



Estrutura lógica de um nó



Estrutura típica de um nó

```
const m = 2;    // ordem da arvore-B
typedef struct node_Btree Btree;
struct node_Btree {
    int num_keys;       // numero de chaves armazenadas
    char keys[2*m-1]; // vetor de chaves
    Btree *desc[2*m]; // ponteiros para os descendentes
    bool leaf;          // flag folha da arvore
};
```

Inserção de Dados (Chave)

- Característica
  - Sempre realizada nos nós **folha**

- Situações a serem analisadas
  1. árvore vazia
  2. *overflow* no nó raiz
  3. inserção em nós folha

Inserção em árvore vazia

Inserção: situação inicial

- Criação e preenchimento do nó
  - primeira chave: criação do nó raiz
  - demais chaves: inserção até a capacidade limite do nó

- Exemplo
  - nó com capacidade para 7 chaves → ordem 8
  - chaves: letras do alfabeto
  - situação inicial: árvore vazia

Inserção: situação inicial

- Chaves B C G E F D A
  - inseridas desordenadamente
  - mantidas ordenadas no nó
- Ponteiros (*)
  - nós folhas: -1 ou fim de lista (NULL)
  - nós internos: referência para o nó filho ou -1
- Nó raiz (= nó folha nesse momento)

Overflow no nó raiz

Inserção: *overflow* nó raiz

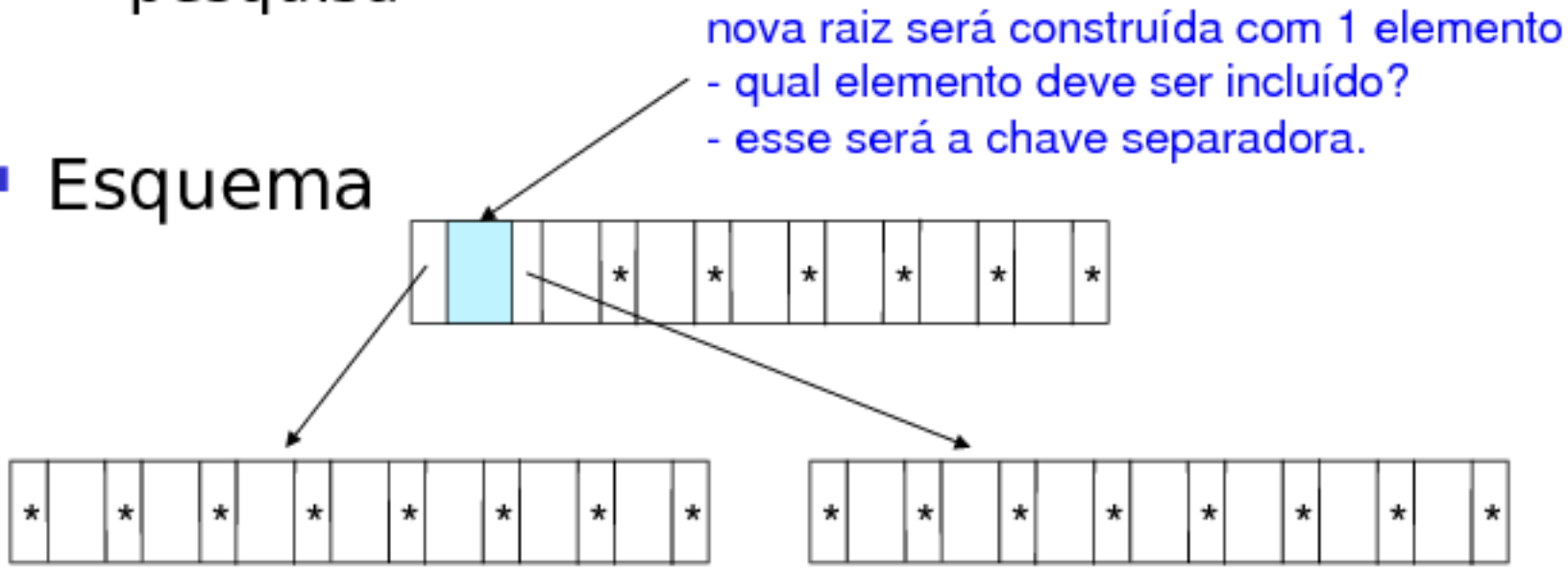
- Passo 1 – particionamento do nó (*split*)
  - nó original → nó original + novo nó
    - *split* 1-to-2
  - as chaves são distribuídas uniformemente nos dois nós
    - considerando chaves do nó original + nova chave

- Exemplo: inserção de J em

  – particionamento da página

Inserção: *overflow* nó raiz

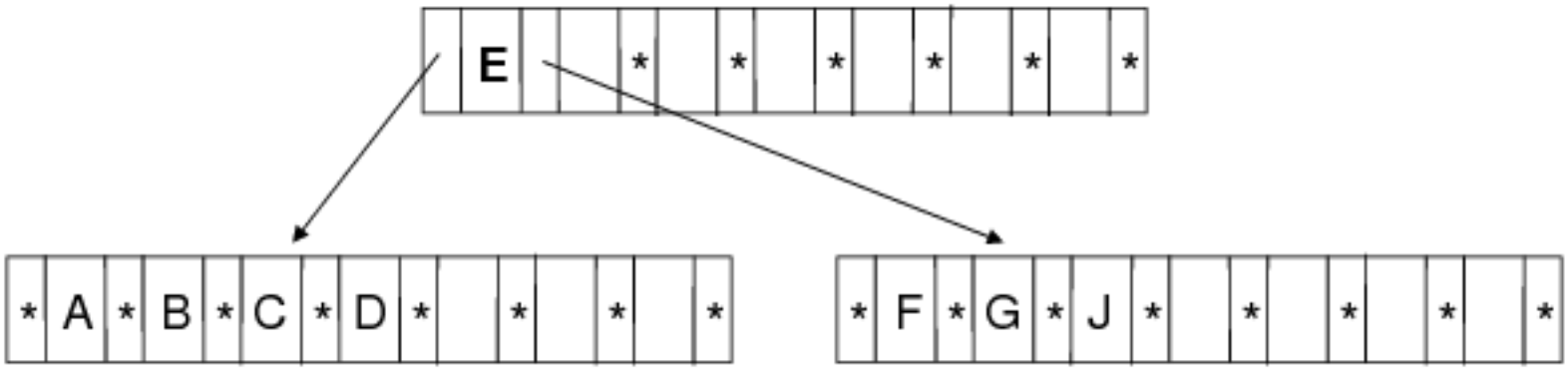
- Passo 2 – criação de uma nova raiz
  - a existência de um nível mais alto na árvore permite a escolha das folhas durante a pesquisa
- Esquema



Inserção: *overflow* nó raiz

- Passo 3 – promoção de chave (*promotion*)
  - a primeira chave do novo nó após particionamento é promovida para o nó raiz

- Exemplo



Inserção em nós folha

Inserção: nós folhas

- Passo 1 – pesquisa
  - a árvore é percorrida até encontrar o nó folha no qual a nova chave será inserida

- Passo 2 – inserção em nó com espaço
  - ordenação da chave após a inserção
  - alteração dos valores dos campos de referência

nó folha em memória principal

Inserção: nós folhas

- Passo 2 – inserção em nó cheio
  - particionamento
    - criação de um novo nó
      (nó original → nó original + novo nó)
    - distribuição uniforme das chaves nos dois nós
  - promoção
    - escolha da primeira chave do novo nó como chave separadora no nó pai (nó por onde a pesquisa passou antes)
    - ajuste do nó pai para apontar para o novo nó
    - propagação de *overflow*

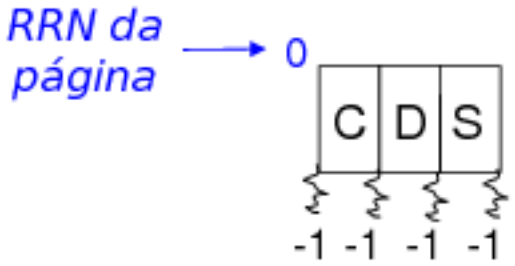
Exemplo

- Insira as seguintes chaves em um índice árvore-B
  - C S D T A M P I B W N G U K

- Ordem da árvore-B: 4
  - em cada nó (página de disco)
    - número de chaves: 3
    - número de ponteiros: 4

C S D T A M P I B W N G U

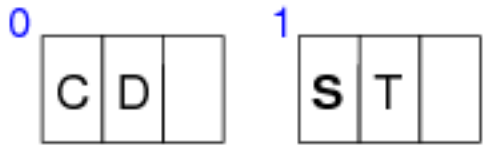
- Passo 1 – inserção de C, S, D
  - criação do nó raiz
    - C
    - C S
    - C D S



C S D T A M P I B W N G U

- Passo 2 – inserção de T
  - nó raiz cheio

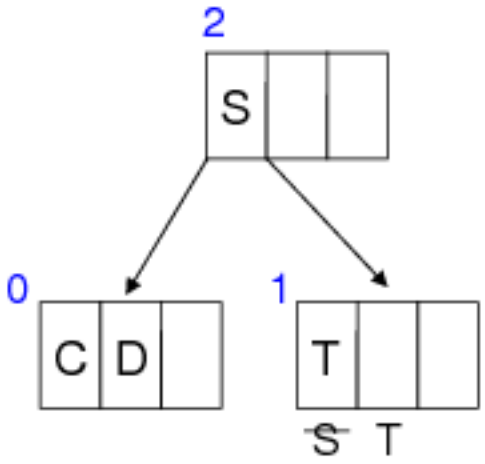
- particionamento do nó
- criação de uma nova raiz
- promoção de S



C S D T A M P I B W N G U

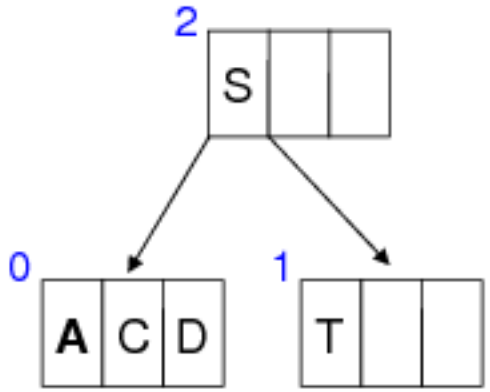
- Passo 2 – inserção de T
  - nó raiz cheio

- particionamento do nó
- criação de uma nova raiz
- promoção de S



C S D T A M P I B W N G U

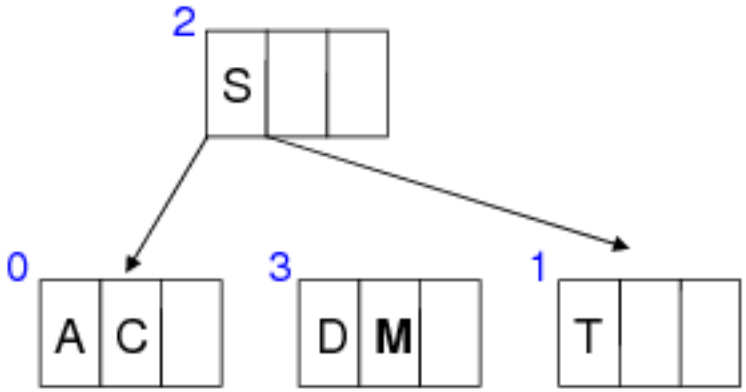
- Passo 3 – inserção de A
  - nó folha com espaço



C S D T A M P I B W N G U

- Passo 4 – inserção de M
  - nó folha 0 cheio

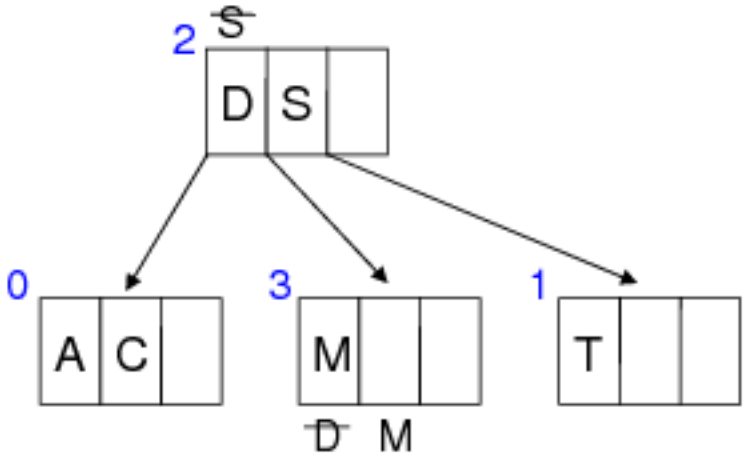
• particionamento do nó



C S D T A M P I B W N G U

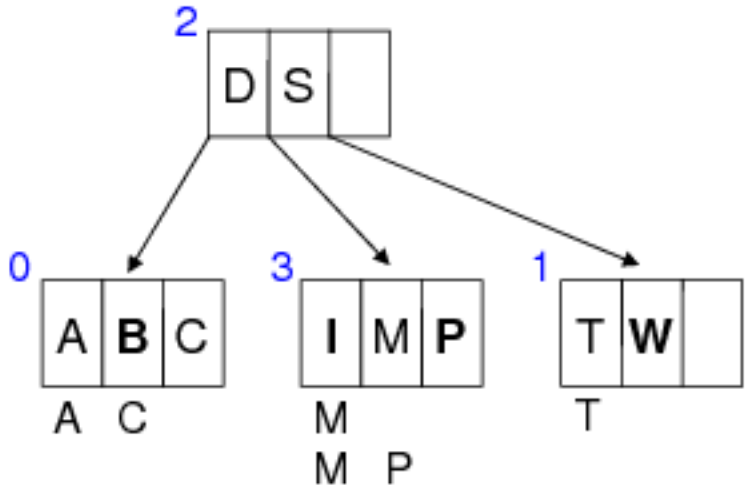
- Passo 4 – inserção de M
  - nó folha 0 cheio

• particionamento do nó
• promoção de D



C S D T A M P I B W N G U

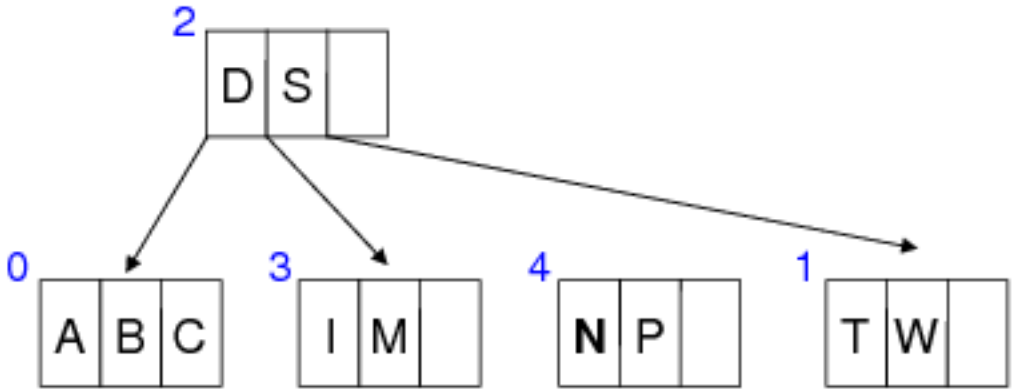
- Passo 5 – inserção de P, I, B, W
  - nós folhas com espaço



C S D T A M P I B W N G U

- Passo 6 – inserção de N
  - nó folha 3 cheio

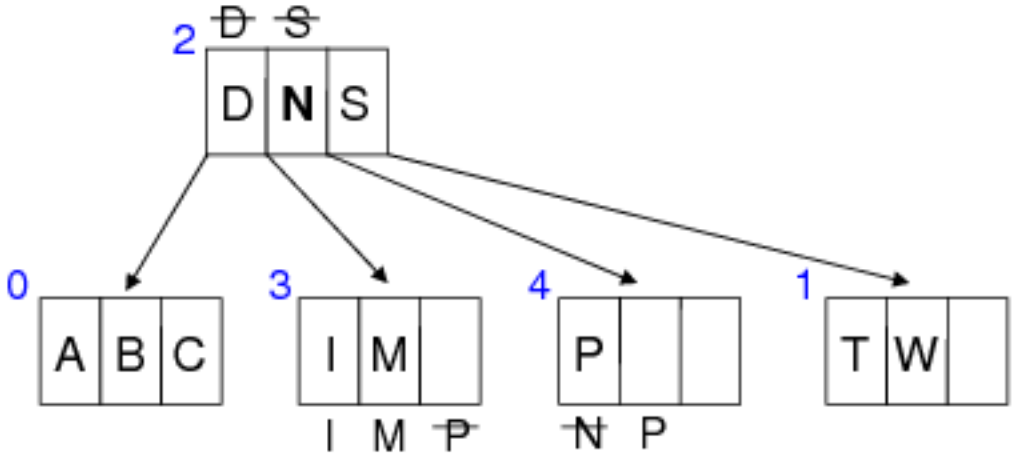
• particionamento do nó



C S D T A M P I B W N G U

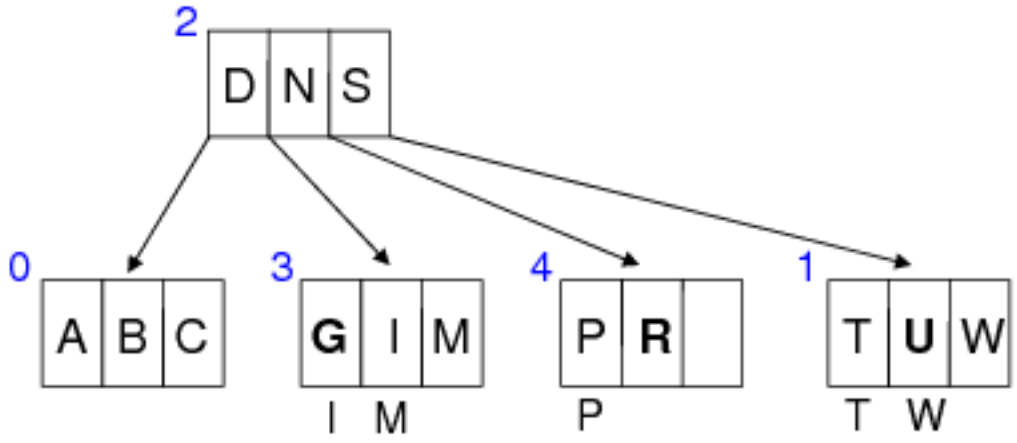
- Passo 6 – inserção de N
  - nó folha 3 cheio

• particionamento do nó
• promoção de N



C S D T A M P I B W N G U

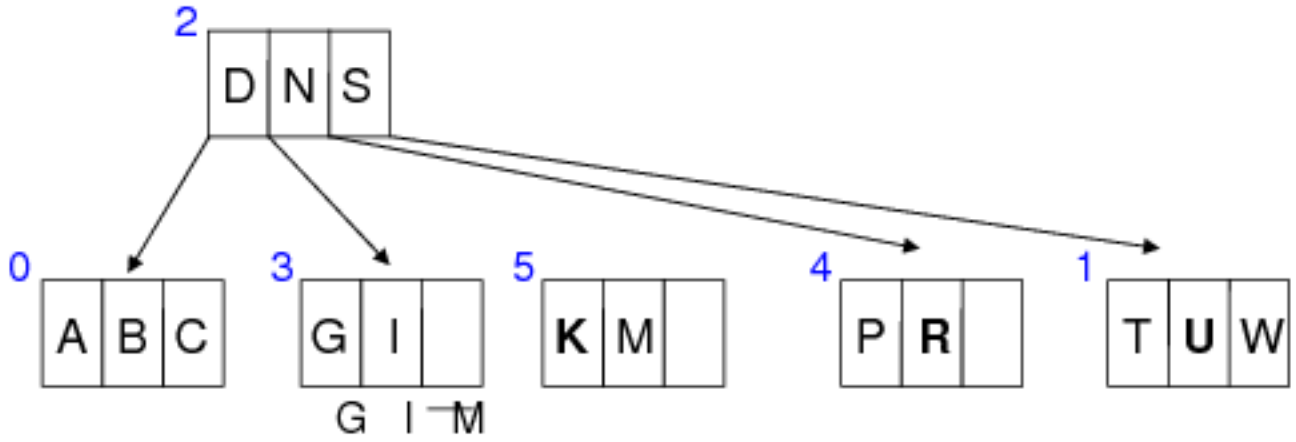
- Passo 7 – inserção de G, U, R
  - nós folhas com espaço

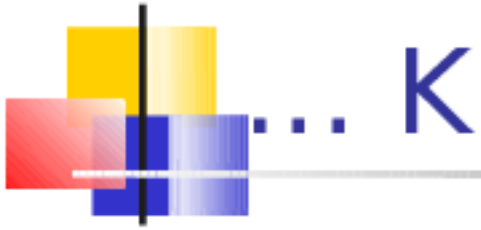


• particionamento do nó 3

... K

- Passo 8 – inserção de K
  - nó folha 3 cheio

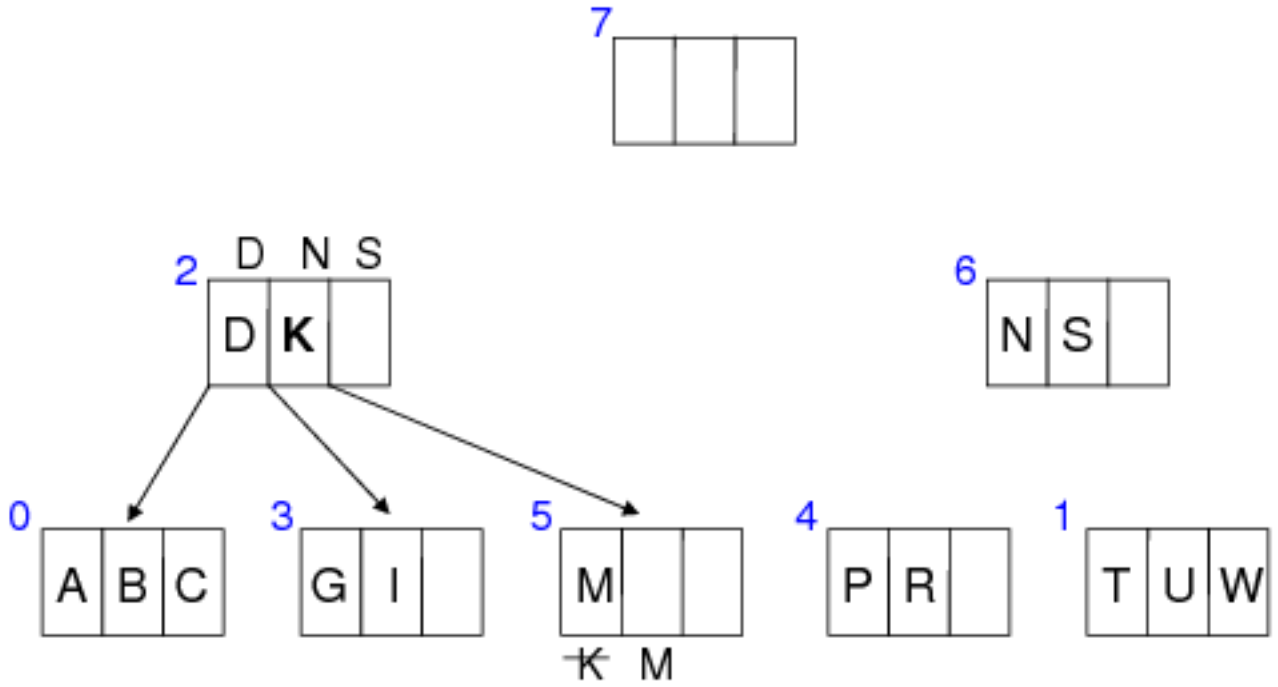




... K

- particionamento do nó 3
- promoção de K
- particionamento do nó 2 e criação de nova raiz

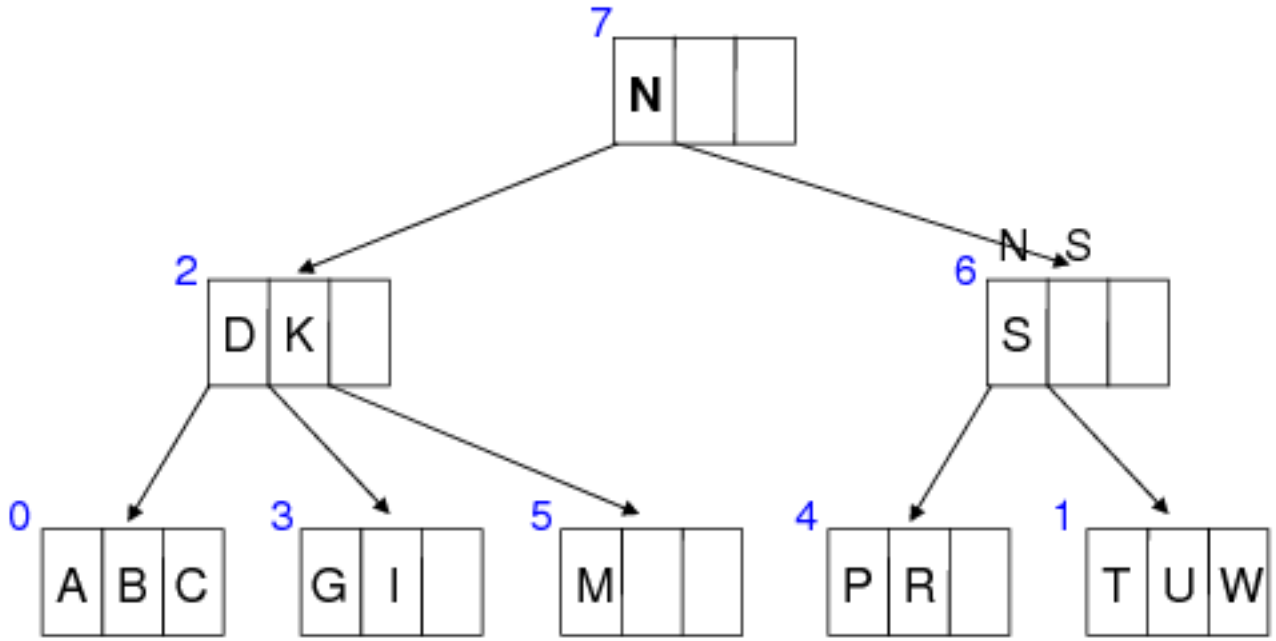
- Passo 8 – inserção de K



... K

- particionamento do nó 3
- promoção de K
- particionamento do nó 2
- promoção de N

- Passo 8 – inserção de K



Exercício

- Na árvore-B do exemplo anterior, insira a chave $, sendo que $ < A

Exercício

- Insira as seguintes chaves em um índice árvore-B
  - C S D T A M P I B W N G U R K E H O L
  - diferentemente do exemplo anterior, escolha o último elemento do primeiro nó para promoção durante o particionamento do nó.

Exercício

- Construa uma árvore-B de ordem 3 pela inserção das chaves A, B, C, D, E, F, G, H e I, nessa ordem

  - Qual o efeito da inserção das chaves em ordem alfabética? A árvore degenerou?