Engenharia de Computação

Estrutura de Dados 2

Aula 14 – Árvore B

Prof. Muriel de Souza Godoi
muriel@utfpr.edu.br

A invenção das Árvores B

- **Árvores B** são uma generalização da ideia de ABB paginada
  - Não são binárias
  - Conteúdo de uma página não é mantido como uma árvore
  - A construção é *bottom-up*
- Um pouco de história
  - **1972:** Bayer and McGreight publicam o artigo *Organization and Maintenance of Large Ordered Indexes*
  - **1979:** Árvores-B viram praticamente **padrão** em sistemas de arquivos de propósito geral

Árvore B

- Características
  - Completamente balanceadas
  - criação bottom-up (em disco)
  - nós folhas → nó raiz

- Inovação
  - Não é necessário construir a árvore a partir do nó raiz, como é feito para árvores em memória principal e para as árvores anteriores

Árvore B - Bottom-Up

- Consequências

- Chaves “erradas” não são mais alocadas no nó raiz
  - Elimina as questões em aberto de chaves separadoras e de chaves extremas
- Não é necessário tratar o problema de desbalanceamento

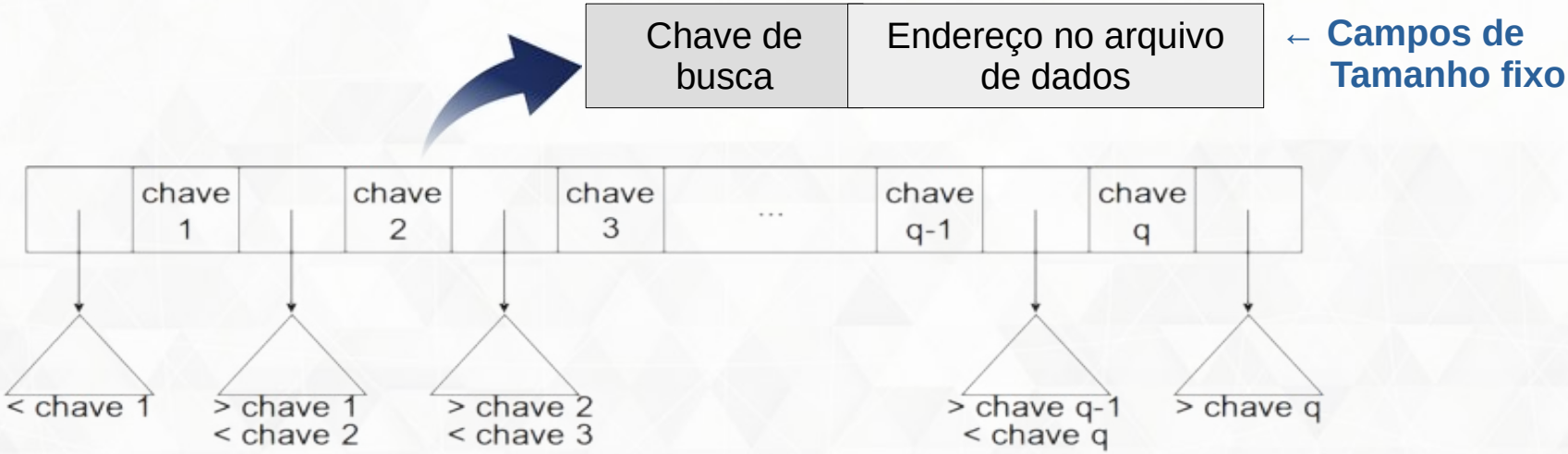
Em uma **Árvore B**, as chaves na raiz da árvore emergem naturalmente

Árvore B - Características

- Características do nó (= página de disco)
  - Sequência ordenada de chaves + Conjunto de ponteiros
    - Número de ponteiros = número de chaves + 1
    - Ordem da árvore: número máximo de ponteiros dos nós
  - Não há uma árvore explícita dentro de uma página
    - ou nó da árvore
  - Registro de tamanho fixo para armazenar

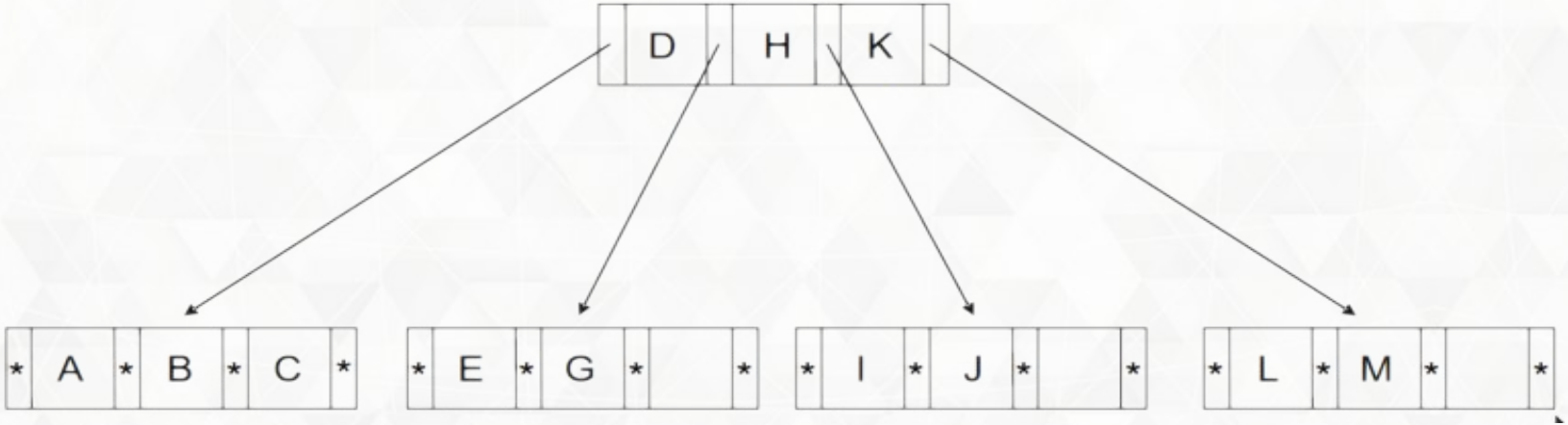
Árvore B - Características

- Estrutura lógica de um nó



Árvore B - Características

- Exemplo de Árvore B de **ordem 4**:
  - Até **3 chaves** por nó;
  - Até **4 filhos**



Árvore B - Características

- **Ordem**
  - Número máximo de ponteiros que pode ser armazenado em um nó
  - **Exemplo**: Árvore B de ordem 8
    - máximo de 7 chaves e 8 ponteiros

- **Observações**
  - Número máximo de ponteiros é igual ao número máximo de descendentes de um nó
  - Nós folhas não possuem filhos, e seus ponteiros são nulos

Árvore B - Inserção de Dados (Chave)

- Característica
  - Sempre realizada nos nós folha
  - A busca binária por uma chave inexistente também termina sempre no nó folha

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

Inserção em árvore vazia

- 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 (NIL)
  - nós internos: RRN do nó filho ou -1
- Nó raiz (= nó folha)

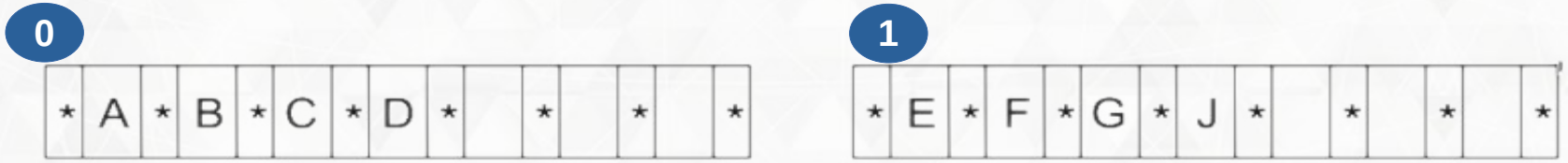
Inserção: situação inicial

- Chaves B C G E F D A
  - inseridas desordenadamente
  - mantidas ordenadas no nó
- Ponteiros (*)
  - nós folhas: -1 ou fim de lista (NIL)
  - nós internos: RRN do nó filho ou -1
- Nó raiz (= nó folha)

```
Raiz = 0
```

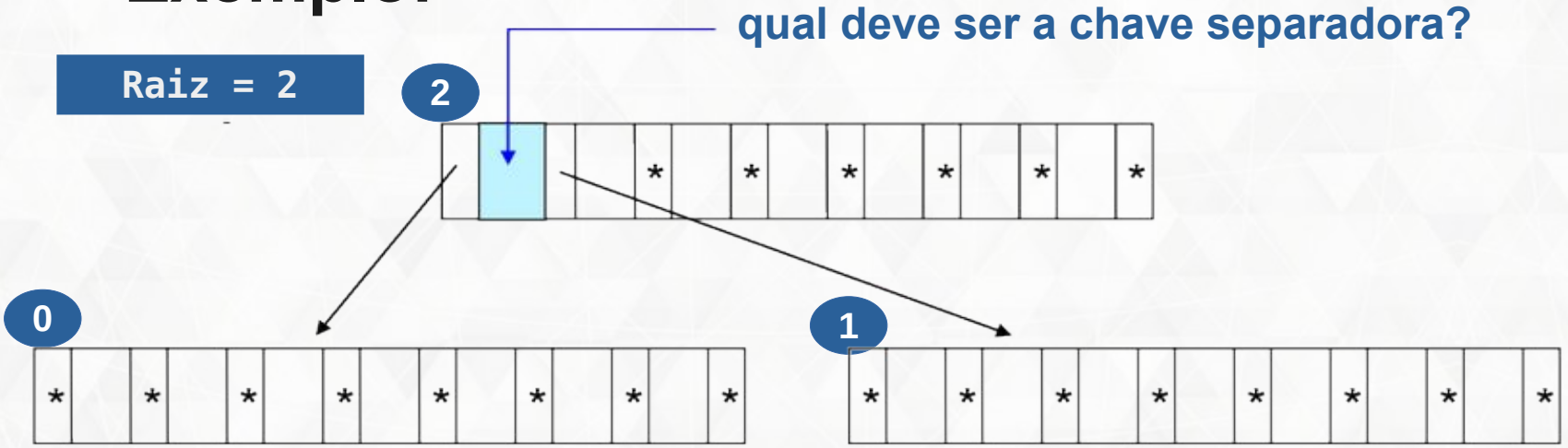
Inserção: Overflow no nó raiz

- **Passo 1** – particionamento do nó (*split*)
  - nó original → nó original + novo nó
  - split 1-to-2
- As chaves são distribuídas uniformemente nos dois nós
  - chaves do nó original + nova chave
- **Exemplo:** inserção de J



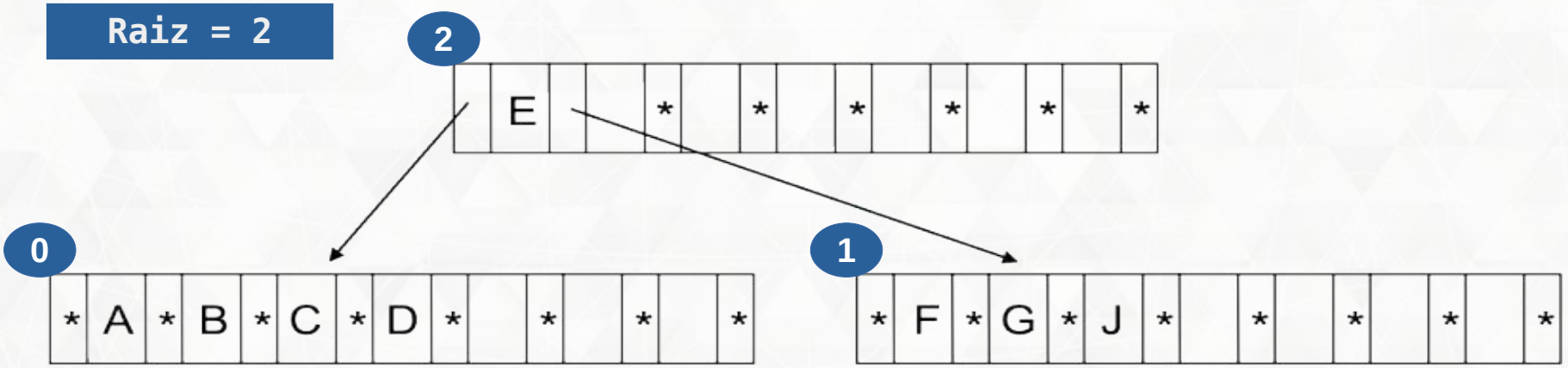
Inserção: Overflow no nó raiz

- **Passo 2 –** criação de uma nova raiz
  - Efeito Bottom-Up → Aumenta altura
  - a existência de um nível mais alto na árvore permite a escolha das folhas durante a pesquisa
- **Exemplo:**



Inserção: Overflow no nó raiz

- **Passo 3** – promoção de chave (*promotion*)
  - a primeira chave do novo nó resultante do particionamento é promovida para o nó raiz
    - É a mediana do conjunto dos dois nós
- **Exemplo:**



Inserção: nós folhas

- **Passo 1** – pesquisa binária
  - inserção sempre nas folhas
  - a árvore é percorrida até encontrar o nó folha no qual a nova chave será inserida

- **Passo 2(a)** – inserção em nó com lugar disponível
  - inserção ordenada da chave no nó (sequencial)
  - alteração dos valores dos campos de referência

Inserção: nós folhas

- **Passo 2(b)** – inserção em nó cheio (*overflow*)
  - Particionamento (*split*)
    - criação de um novo nó
    - nó original → nó original + novo nó
    - distribuição uniforme das chaves nos dois nós
  - Promoção (*promotion*)
    - escolha da primeira chave do novo nó como chave
    - separadora no nó pai
    - Ajuste do nó pai para apontar para o novo nó
    - propagação recursiva de overflow

Exemplo

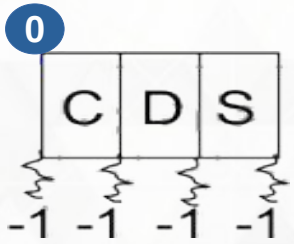
- Insira as seguintes chaves em um índice Árvore B
  - C S D T A M P I B W N G U R K E H O L J Y Q Z F X V

- Ordem da Árvore B: 4
  - Número de chaves: 3
  - Número de ponteiros: 4

Exemplo

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

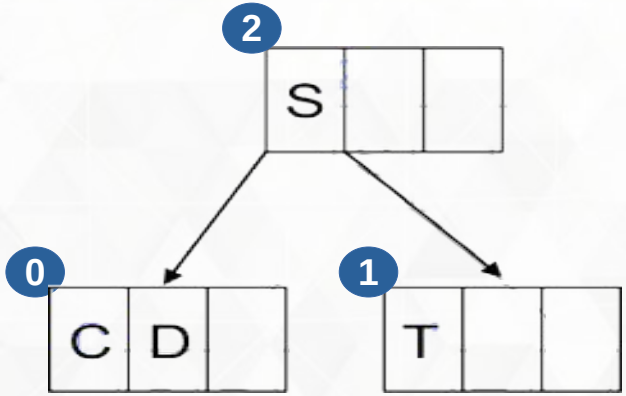
```
Raiz = 0
```



Exemplo

- **Passo 2** – inserção de T
  - nó raiz cheio (*split / promote* ***S***)

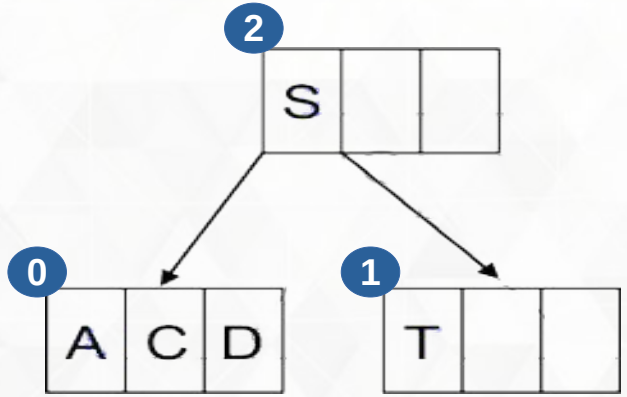
```
Raiz = 2
```



Exemplo

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

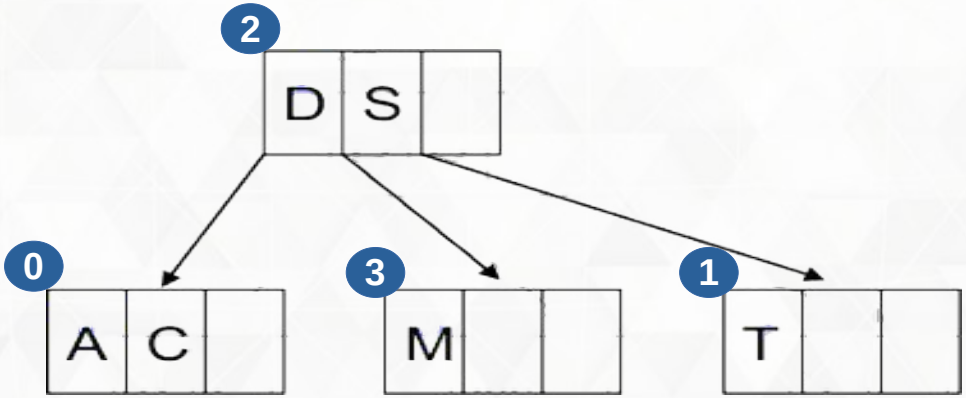
```
Raiz = 2
```



Exemplo

- **Passo 4** – inserção de M
  - nó folha 0 cheio (*split / promote **D***)

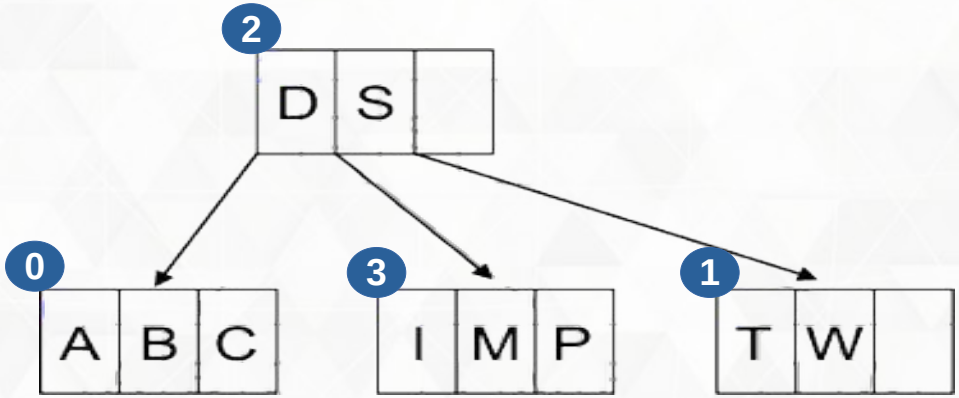
`Raiz = 2`



Exemplo

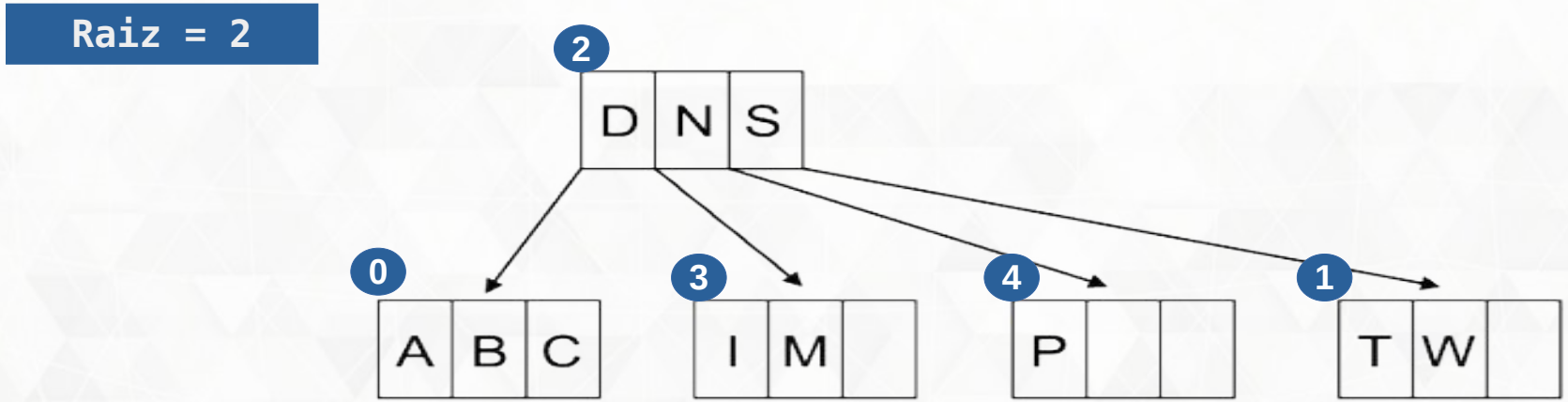
- **Passo 5** – inserção de P, I, B, W
  - nós folhas com espaço

```
Raiz = 2
```



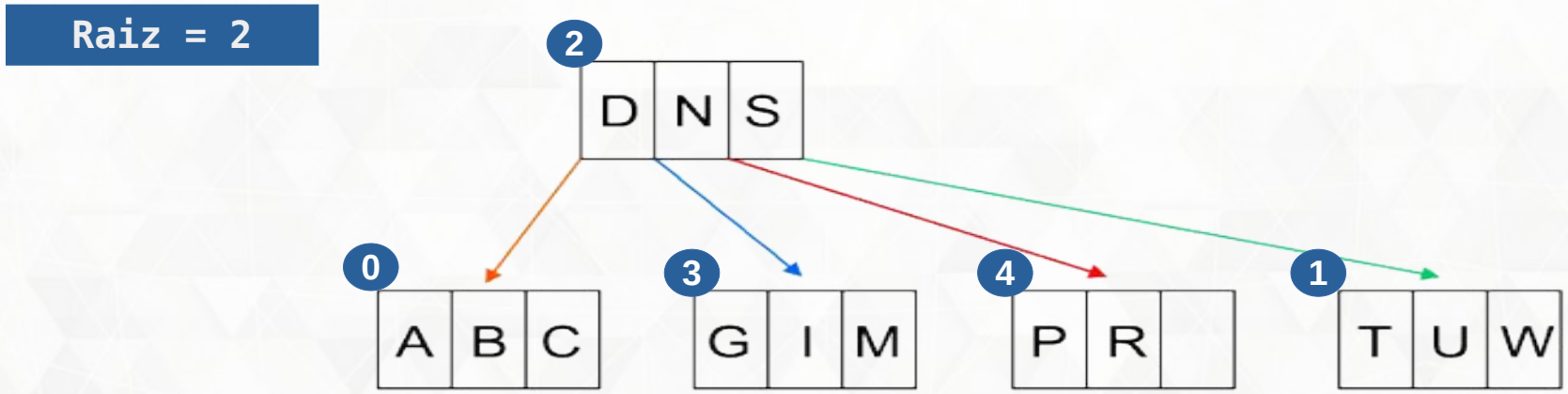
Exemplo

- **Passo 6** – inserção de N
  - nó folha 3 cheio (*split / promote **N***)



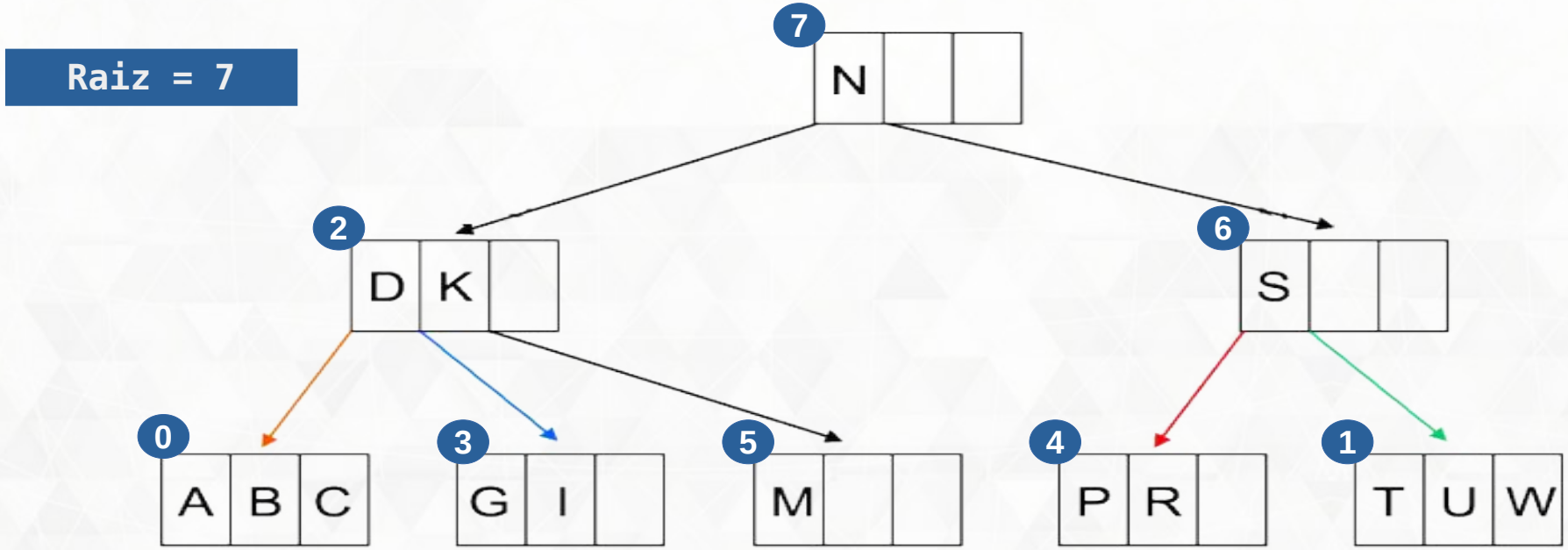
Exemplo

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



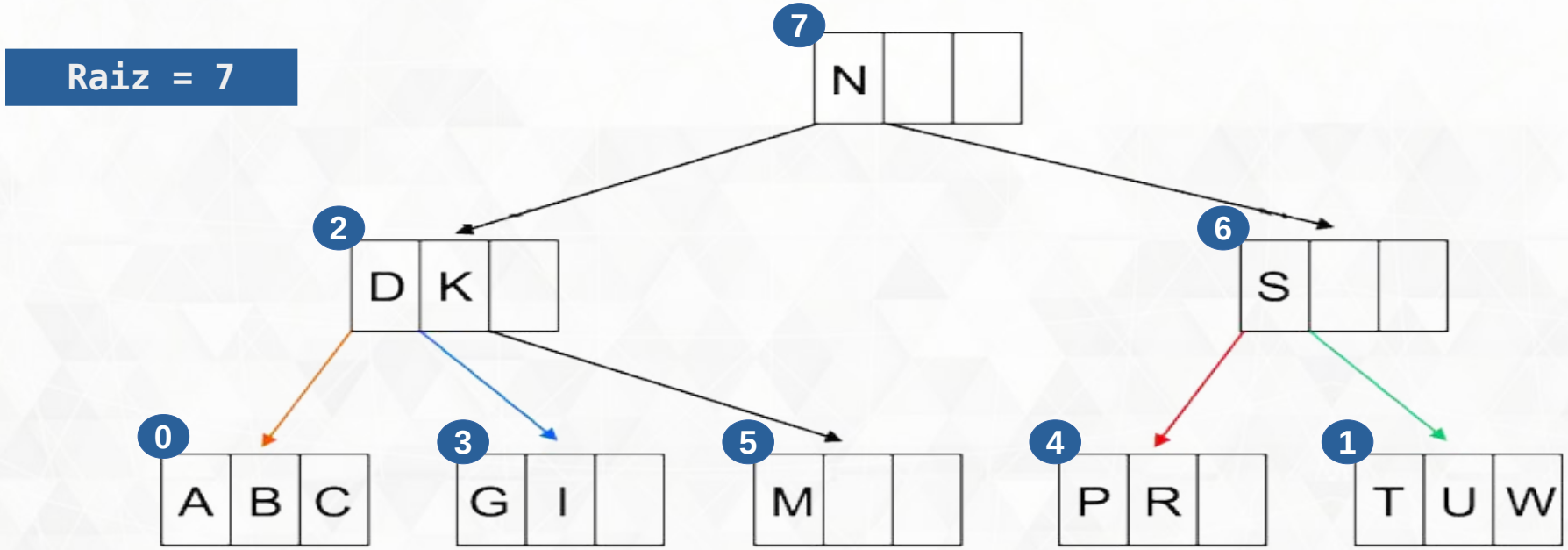
Exemplo

- **Passo 8** – inserção de K
  - nó folha 3 cheio (*split / promote 2x - **K** e **N***)



Exemplo

- **Passo 9** – Finalizar a construção da árvore….
  - Insira: E H O L J Y Q Z F X V



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 J Y Q Z F X V
  - diferentemente do exemplo anterior, escolha o **último elemento do primeiro nó** para promoção durante o particionamento do nó.