Árvore-B
Busca e Inserção

6897/1 e 5187/31 – Organização e Recuperação de Dados

Profa. Valéria D. Feltrim

UEM – CTC – DIN

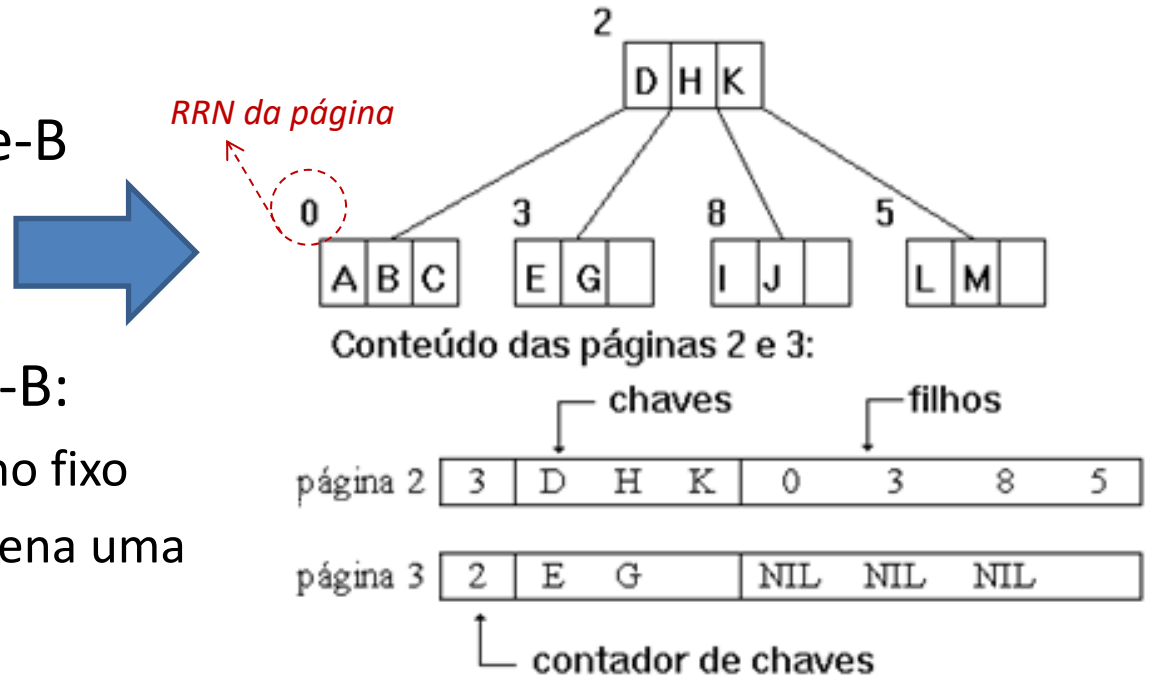
Slides preparados com base no Cap. 8 do livro FOLK, M.J. & ZOELLICK, B. *File Structures*. 2nd Edition, Addison-Wesley Publishing Company, 1992.

Busca e inserção em árvore-B

- Possível estrutura de página de árvore-B em C

```
struct {
  short CONTACHAVES;              /* número de chaves na página */
  char CHAVE[MAXCHAVES];          /* vetor que armazena as chaves */
  short FILHO[MAXCHAVES+1];       /* RRNs dos filhos */
} PAGINA;
```

- Parte de uma árvore-B de ordem 4



- O arquivo da árvore-B:
  - Registros de tamanho fixo
  - Cada registro armazena uma página da árvore-B

Pesquisa em árvore-B

- Algoritmo de busca na árvore-B
  - Recursivo
    - Descida na árvore
  - Trabalha em 2 etapas:
    - Alterna entre busca de páginas e busca “dentro” da página

Pesquisa em árvore-B

```
FUNÇÃO busca (RRN, CHAVE, RRN_ENCONTRADO, POS_ENCONTRADA)
 se RRN == NULL então     /* condição de parada */
    retorne NAO_ENCONTRADO
 senão
   leia a página armazenada no RRN para PAG
   busque CHAVE em PAG.CHAVE[] e faça POS receber a posição em que CHAVE
   ocorre ou deveria ocorrer se estivesse em PAG
   se CHAVE foi encontrada então
        RRN_ENCONTRADO := RRN     /* RRN da página que contém a chave */
        POS_ENCONTRADO := POS     /* posição da chave na página*/
        retorne ENCONTRADO
   senão          /* siga o ponteiro para a próxima página da busca */
        retorne(busca(PAG.FILHO[POS], CHAVE, RRN_ENCONTRADO, POS_ENCONTRADO))
   fim se
 fim se
fim FUNÇÃO
```

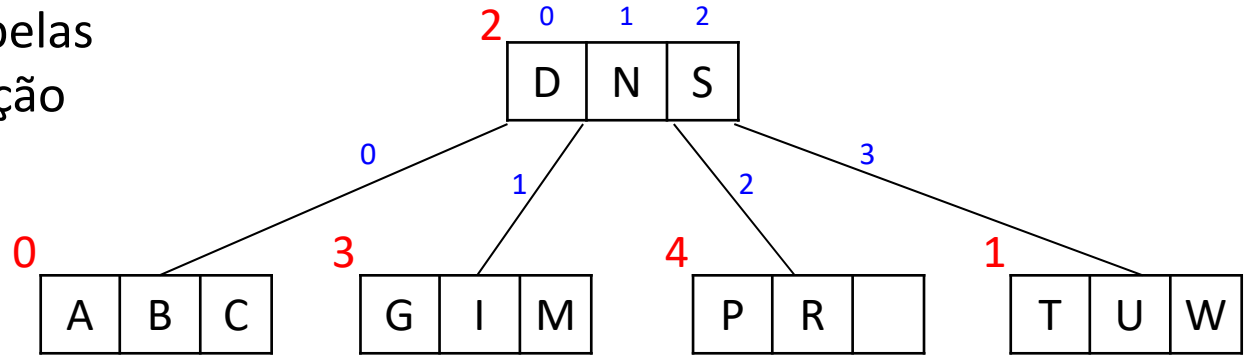
Pesquisa em árvore-B

- A função ***busca*** faz uma busca **recursiva** que se inicia na página raiz da árvore-B
- Para cada página, busca-se internamente pela chave
  - Se a chave for encontrada:
    - A função retorna ENCONTRADA
    - O RRN da página na qual a chave foi encontrada é retornado em RRN_ENCONTRADO
    - A posição chave no vetor de chaves da página volta em POS_ENCONTRADO
  - Se a chave não for encontrada, a busca prossegue até encontrar NULL em uma das folhas, retornando NAO_ENCONTRADO

- Na primeira chamada da função ***busca***, o RRN da página raiz é passado como parâmetro

Pesquisa em árvore-B

Exercício: Simule a busca pelas chaves ***K*** e ***P*** usando a função ***search***




```
FUNÇÃO busca (RRN, CHAVE, RRN_ENCONTRADO, POS_ENCONTRADA)
 se RRN == NULL então
    retorne NAO_ENCONTRADO
 senão
    leia a página armazenada no RRN para PAG
    busque CHAVE em PAG.CHAVE[] e faça POS receber a posição em que CHAVE ocorre
    ou deveria ocorrer se estivesse em PAG
    se CHAVE foi encontrada então
        RRN_ENCONTRADO := RRN
        POS_ENCONTRADO := POS
        retorne ENCONTRADO
    senão
        retorne(busca(PAG.FILHO[POS], CHAVE, RRN_ENCONTRADO, POS_ENCONTRADO))
    fim se
 fim se
fim FUNÇÃO
```

Inserção, divisão e promoção

- Algoritmo de inserção em árvore-B
  - O processo começa com uma busca
    - Inicia na raiz e continua até atingir uma folha
  - Uma vez localizada a posição de inserção (**SEMPRE em uma folha**), pode ser necessário realizar divisões e promoções, sempre de baixo para cima
  - O algoritmo pode então ser pensado em 3 partes:
    1. Uma busca pela chave na página atual, como em ***busca***, antes da chamada recursiva
    2. Uma chamada recursiva para "descer" um nível na árvore (a descida é feita até encontrar um ponteiro nulo, que estará em uma folha)
    3. Inserção, divisão e promoção (se necessário) executadas no retorno das chamadas recursivas, fazendo com que esses processos ocorram na "subida" da árvore

Inserção, divisão e promoção

- **FUNÇÃO *insere* (RRN_ATUAL, CHAVE, FILHO_D_PRO, CHAVE_PRO)**
  - Argumentos:
    1. **RRN_ATUAL**: contém o RRN da página que está atualmente em uso (inicialmente, a raiz)
    2. **CHAVE**: contém a chave a ser inserida
    3. **CHAVE_PRO**: usado para armazenar um valor de retorno.
       Se a inserção da chave resultar em divisão e promoção, CHAVE_PROMO conterá a chave promovida
    4. **FILHO_D_PRO**: usado para armazenar um valor de retorno.
       Se houver uma divisão, os níveis superiores da sequência de chamadas devem inserir não apenas a chave promovida, mas também o RRN da nova página criada na divisão
       - Quando existe uma CHAVE_PROMO, FILHO_D_PROMO conterá o ponteiro para o seu filho direito (que corresponde a nova página resultante da divisão)

Inserção, divisão e promoção

- Valores de retorno da função *insere*
  1. PROMOCAO, se uma chave está sendo promovida
  2. SEM_PROMOCAO, se a inserção foi feita sem necessidade de dividir a página
  3. ERRO, se a inserção não puder ser realizada (se a chave já está na árvore)
- Variáveis locais importantes da função *insere*:
  - PAG: página que está sendo atualmente examinada
  - NOVAPAG: nova página que é criada caso ocorra uma divisão
  - POS: posição da chave em PAG, se ela estiver lá; caso contrário, a posição em que deve ser inserida (ou a posição do ponteiro para a próxima página)
  - RRN_PRO: recebe o valor do RRN da página promovida para o nível corrente (via FILHO_D_PRO)
    - Se uma divisão ocorre no nível imediatamente inferior, RRN_PRO contém o RRN da nova página criada durante a divisão. RRN_PRO é o filho direito que deve ser inserido junto com CHV_PRO em PAG
  - CHV_PRO: recebe o valor da chave promovida para o nível corrente (via CHAVE_PRO)

Inserção, divisão e promoção

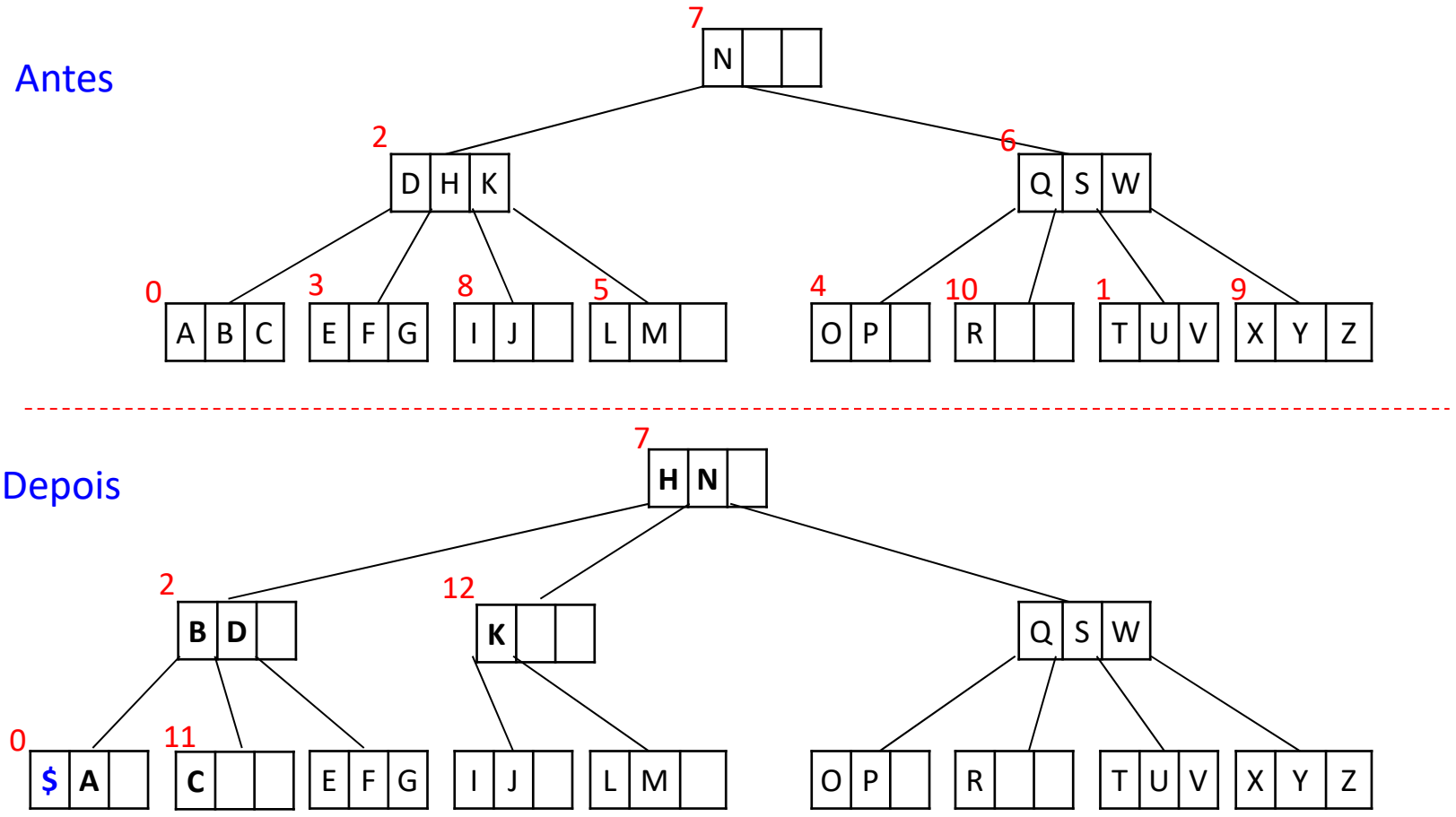
```
FUNÇÃO insert (RRN_ATUAL, CHAVE, FILHO_D_PRO, CHAVE_PRO)
   se RRN_ATUAL == NULL então /* condição de parada */
        CHAVE_PRO := CHAVE
        FILHO_D_PRO := NULL
        retorne PROMOCAO
   senão
        leia a página armazenada em RRN_ATUAL para PAG
        busque CHAVE em PAG e faça POS receber a posição em que
       CHAVE ocorre ou deveria ocorrer em PAG
   fim se
   se CHAVE foi encontrada então
        imprima mensagem de chave duplicada
        retorne ERRO
   fim se
   RETORNO:= insere(PAG.FILHO[POS], CHAVE, RRN_PRO, CHV_PRO)
   …
                        /*continua no próximo slide*/
```

Inserção, divisão e promoção

```
 /*continuação da função insere, logo após a chamada recursiva*/
…
se RETORNO == SEM_PROMOCAO ou ERRO então
     retorne RETORNO
senão
   se existe espaço em PAG para inserir CHV_PRO então
     insira CHV_PRO e RRN_PRO (chave promovida e filha) em PAG
     escreva PAG no arquivo em RRN_ATUAL
     retorne SEM_PROMOCAO
   senão
     divide(CHV_PRO, RRN_PRO, PAG, CHAVE_PRO, FILHO_D_PRO, NOVAPAG)
     escreva PAG no arquivo em RRN_ATUAL
     escreva NOVAPAG no arquivo em FILHO_D_PRO
     retorne PROMOCAO
   fim se
fim se
fim FUNÇÃO
```

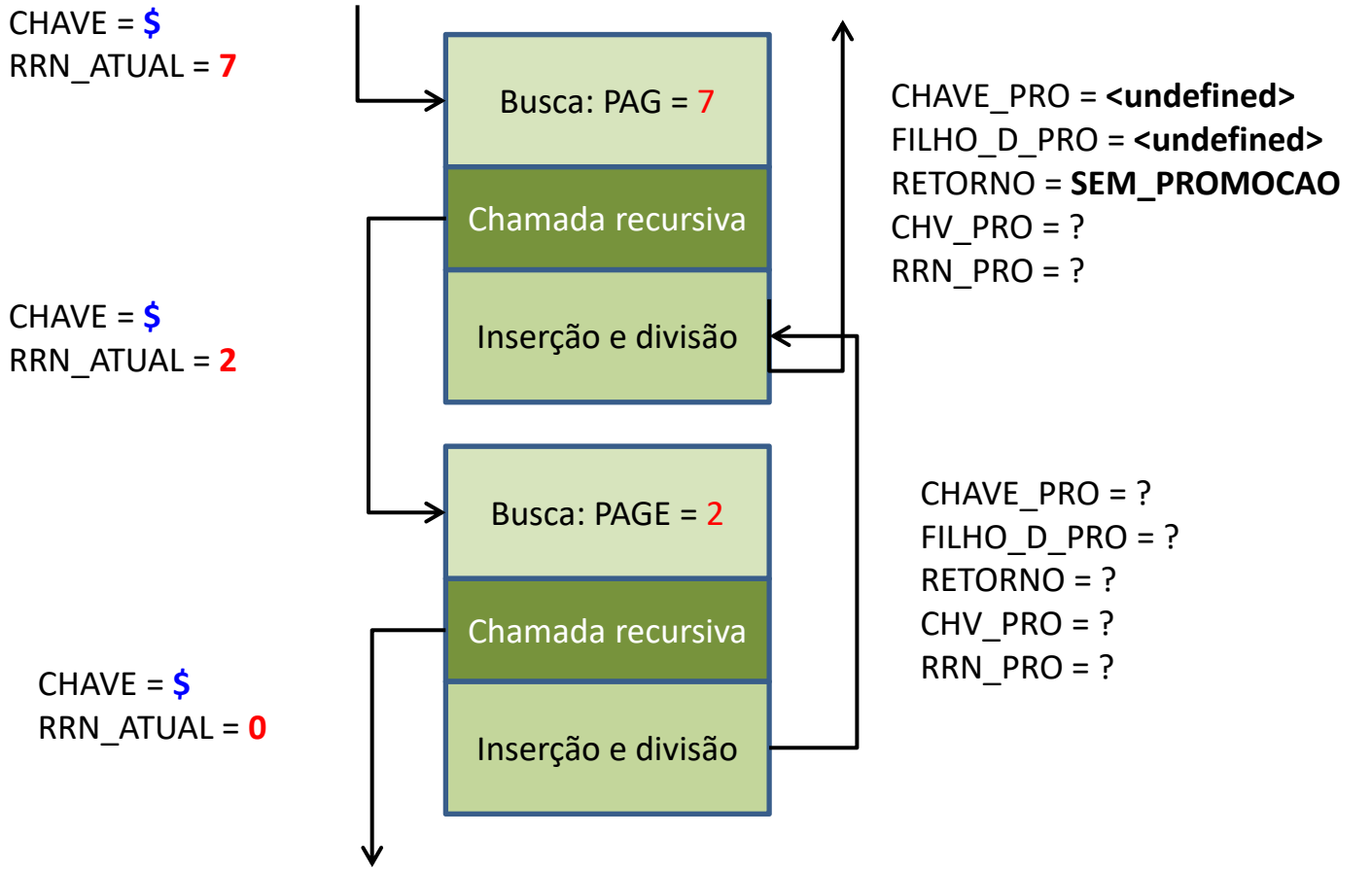
Inserção, divisão e promoção

- Exemplo: inserção do caracter **$** na árvore-B abaixo



Inserção, divisão e promoção

- Exercício: simular as chamadas recursivas para a **inserção da chave "$"** na árvore do slide anterior usando a função *insere*
  (na tabela ASCII, o símbolo "$" vem antes de qualquer letra)



Inserção, divisão e promoção

- Em implementações reais, a função *insere* usa várias funções de suporte
- Uma delas é a função ***divide***
  - Cria uma nova página
  - Distribui as chaves entre a página atual e a nova página
    - Usa uma página auxiliar para isso
  - Determina qual chave e qual RRN (ponteiro para o filho direito) promover
    - Chave → sempre é a chave mediana da página auxiliar
      - Como a página auxiliar tem um tamanho fixo, a chave mediana sempre estará na mesma posição
    - RRN (ponteiro do filho direito) → sempre é o RRN da nova página
      - A nova página sempre é gravada no fim do arquivo → função ***novoRRN*()**

Inserção, divisão e promoção

PROC ***divide*(CHAVE_I, RRN_I, PAG, CHAVE_PRO, FILHO_D_PRO, NOVAPAG)**

Copie PAG para uma ***página auxiliar*** **PAGAUX**, que terá espaço para uma chave e um ponteiro extras

Insira CHAVE_I e RRN_I nos lugares apropriados em PAGAUX

Aloque e inicialize uma *nova página* NOVAPAG

Faça CHAVE_PRO receber o valor da chave mediana de PAGAUX, que será promovida após o retorno da função ***divide***

Faça FILHO_D_PRO receber o RRN de NOVAPAG

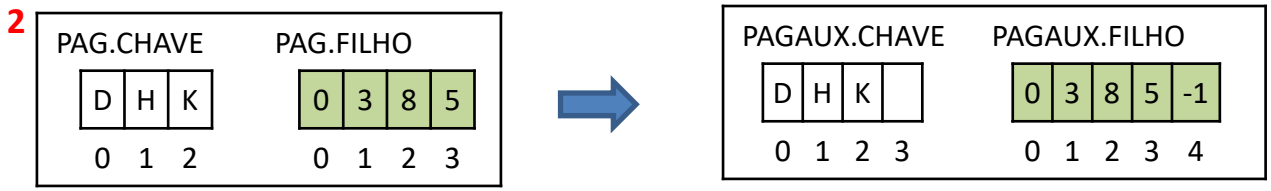
Copie as chaves e ponteiros que vêm antes de CHAVE_PRO em PAGAUX para PAG

Copie as chaves e ponteiros que vêm depois de CHAVE_PRO em PAGAUX para NOVAPAG

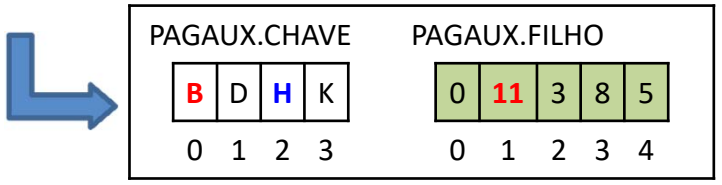
fim PROCEDIMENTO

Inserção, divisão e promoção

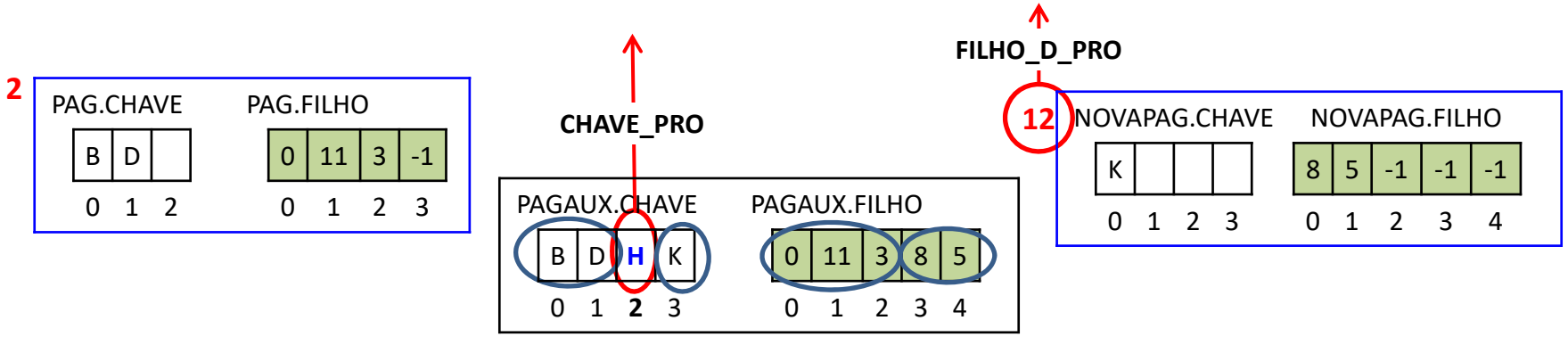
- Exemplo: ***divide*** **(CHAVE_I = 'B', RRN_I = 11, PAG = 2, CHAVE_PRO, FILHO_D_PRO, NOVAPAG)**



  - Insira CHAVE_I (**B**) e RRN_I (**11**) na posição apropriada em PAGAUX



  - Divida o conteúdo de PAGAUX entre PAG e NOVAPAG, exceto pela chave mediana (**H**) → **H** será promovida juntamente com o RRN da NOVAPAG(**12**)



Inserção, divisão e promoção

- Como saber qual será o RRN de uma nova página?
  - Sempre que uma página é criada ela é gravada no fim do arquivo
  - As páginas têm tamanho fixo e conhecido → *sizeof(PAG)*

```
FUNÇÃO novoRRN()
   faça um seek para o fim do arquivo
   faça BYTEOFFSET receber o byte-offset do fim do arquivo
   faça TAMANHOPAG receber o tamanho em bytes de uma página
   faça TAMANHOCAB receber o tamanho em bytes do cabeçalho
   retorne (BYTEOFFSET/TAMANHOPAG – TAMANHOCAB)
fim FUNÇÃO
```

Árvore-B

- Procedimento principal: usado para ativar a função de inserção
  - Abre/cria o arquivo com a árvore-B e identifica/cria a página raiz
    - Assume que o RRN da raiz está armazenado no cabeçalho do arquivo da árvore-B, se o arquivo existir
    - Se a árvore-B ainda não existe, cria o arquivo, inicializa a raiz e grava a primeira página
  - Lê as chaves a serem armazenadas na árvore-B e chama a função *insere()*
  - Cria uma nova raiz quando houver divisão da raiz atual
    - Quando a função *insere()* retornar PROMOÇÃO
    - Cria a página que será a nova raiz
    - Atualiza o RRN da raiz

Árvore-B

```
PROCEDIMENTO PRINCIPAL main
   se (o arquivo B-tree existe) então
      abra o arquivo B-tree
      leia o cabeçalho e armazene em RAIZ
   senão
      crie o arquivo B-tree
      faça RAIZ = 0 e a escreva no cabeçalho
      inicialize uma página e a escreva no arquivo
   leia uma chave e armazene em CHAVE
   enquanto (existirem chaves a serem inseridas) faça
      se (insere(RAIZ, CHAVE, FILHO_D_PRO, CHAVE_PRO) == PROMOCAO) então
         crie uma nova página e chame-a de NOVAPAG
         NOVAPAG.CHAVE[0] = CHAVE_PRO
         NOVAPAG.FILHO[0] = RAIZ          /* filho esquerdo */
         NOVAPAG.FILHO[1] = FILHO_D_PRO   /* filho direito */
         faça RAIZ receber o RRN de NOVAPAG
      fim_se
      leia a próxima chave e armazene em CHAVE
   fim_enquanto
   escreva RAIZ no cabeçalho do arquivo B-tree
   feche o arquivo B-tree
fim PROCEDIMENTO PRINCIPAL
```