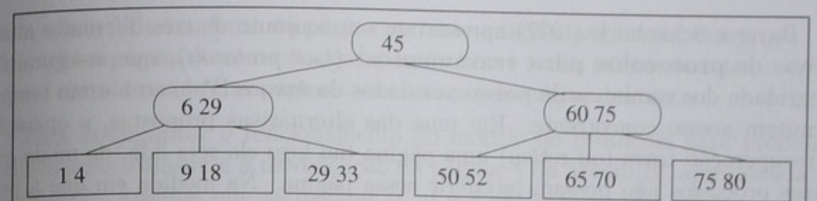
O procedimento Pesquisa deve ser implementado como no Programa 6.11.

Programa 6.11 *Procedimento para pesquisar na árvore B**

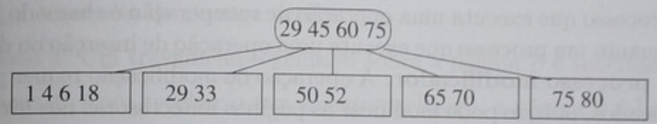
```
procedure Pesquisa (var x: TipoRegistro; var Ap: TipoApontador);
var i: integer;
begin
  if Ap^.Pt = Interna
  then with Ap^ do
       begin
       i := 1;
       while (i < ni) and (x.Chave > ri[i]) do i := i + 1;
       if x.Chave < ri[i]
       then Pesquisa(x, pi[i-1])
       else Pesquisa(x, pi[i])
       end
  else with Ap^ do
       begin
       i := 1;
       while (i < ne) and (x.Chave > re[i].Chave) do i := i + 1;
       if x.Chave = re[i].Chave
       then x := re[i]
       else writeln('Registro nao esta presente na arvore');
    end;
end;
```

A operação de **Inserção** de um registro em uma árvore B* é essencialmente igual à inserção de um registro em uma árvore B. A única diferença é que, quando uma folha é dividida em duas, o algoritmo promove uma cópia da chave que pertence ao registro do meio para a página pai no nível anterior, retendo o registro do meio na página folha da direita.

A operação de **Retirada** em uma árvore B* é relativamente mais simples do que em uma árvore B. O registro a ser retirado reside sempre em uma página folha, o que torna sua remoção simples, não havendo necessidade de utilização do procedimento para localizar a chave antecessora (vide procedimento Antecessor do Programa 6.9). Desde que a página folha fique pelo menos com metade dos registros, as páginas do índice não precisam ser modificadas, mesmo que uma cópia da chave que pertence ao registro a ser retirado esteja no índice. A Figura 6.16 mostra a árvore B* resultante quando a seguinte sequência de chaves é retirada da árvore B* da Figura 6.15: 5 19 22 60. Observe que a retirada da chave 9 da árvore da Figura 6.16(a) provoca a redução da árvore.



(a) Retirada dos registros 5, 19, 22, 60 da árvore da Figura 6.15



(b) Retirada do registro 9 em (a): a estrutura do índice é modificada

Figura 6.16 *Retirada de registros em árvores B*.*

6.3.3 Acesso Concorrente em Árvores B*

Em muitas aplicações, o acesso simultâneo ao banco de dados por mais de um usuário é um fator importante. Nesses casos, permitir acesso para apenas um processo de cada vez pode criar um gargalo inaceitável para o sistema de banco de dados. A concorrência é então introduzida para aumentar a utilização e melhorar o tempo de resposta do sistema. Desse modo, o uso de árvores B* em tais sistemas deve permitir o processamento simultâneo de várias solicitações diferentes.

Entretanto, existe a necessidade de criar mecanismos chamados **protocolos** para garantir a integridade tanto dos dados quanto da estrutura. Considere a situação em que dois processos estejam simultaneamente acessando o banco de dados. Em determinado momento, um dos processos está percorrendo uma página para localizar o intervalo no qual a chave de pesquisa se encaixa e seguir o apontador para a subárvore correspondente, enquanto o outro processo está inserindo um novo registro que provoca divisões de páginas no mesmo caminho da árvore. Pode acontecer de o processo que está percorrendo a página obtenha um apontador para uma subárvore errada ou para um endereço inexistente.

Uma página é chamada **segura** quando se sabe que não existe possibilidade de modificações na estrutura da árvore, como consequência de uma operação de inserção ou de retirada naquela página. Cabe lembrar que a operação de recuperação não altera a estrutura da árvore, ao contrário das operações de inserção ou retirada, que podem provocar modificações na sua estrutura. No caso de operações de inserção, uma página é considerada segura se o número atual de chaves naquela página é menor do que $2m$. No caso de operações de retirada, uma página é considerada segura quando o número de chaves na página é maior do que m. Os algoritmos para acesso concorrente fazem uso desses fatos para aumentar o nível de concorrência em uma árvore B*.