Continuação do Programa 5.15

```
procedure Antecessor (q: TipoApontador; var r: TipoApontador;
                      var Fim: boolean);
begin
  if r^.Dir <> nil
  then begin
       Antecessor (q, r^.Dir, Fim);
       if not Fim then DirCurto (r, Fim);
       end
  else begin
       q^.Reg := r^.Reg; q := r;
       r := r^.Esq;       dispose (q);
       if r <> nil then Fim := true;
       end;
end; { Antecessor }
begin { IRetira }
  if Ap = nil
  then begin writeln ('Chave nao esta na arvore'); Fim := true; end
  else if x.Chave < Ap^.Reg.Chave
       then begin
            IRetira (x, Ap^.Esq, Fim);
            if not Fim then EsqCurto (Ap, Fim);
            end
       else if x.Chave > Ap^.Reg.Chave
       then begin
            IRetira (x, Ap^.Dir, Fim);
            if not Fim then DirCurto (Ap, Fim);
            end
       else begin { Encontrou chave }
            Fim := false; Aux := Ap;
            if Aux^.Dir = nil
            then begin
                 Ap := Aux^.Esq; dispose (Aux);
                 if Ap <> nil then Fim := true;
                 end
            else if Aux^.Esq = nil
                 then begin
                      Ap := Aux^.Dir; dispose (Aux);
                      if Ap <> nil then Fim := true;
                      end
                 else begin
                      Antecessor (Aux, Aux^.Esq, Fim);
                      if not Fim then EsqCurto (Ap, Fim);
                      end;
            end;
end; { IRetira }
begin { Retira }
  IRetira (x, Ap, Fim)
end; { Retira }
```


A Figura 5.8 mostra o resultado obtido quando se retira uma sequência de chaves da árvore SBB: a árvore à esquerda é obtida após a retirada da chave 7 da árvore à direita na Figura 5.7; a árvore do meio é obtida após a retirada da chave 5 da árvore anterior; a árvore à direita é obtida após a retirada da chave 9 da árvore anterior.

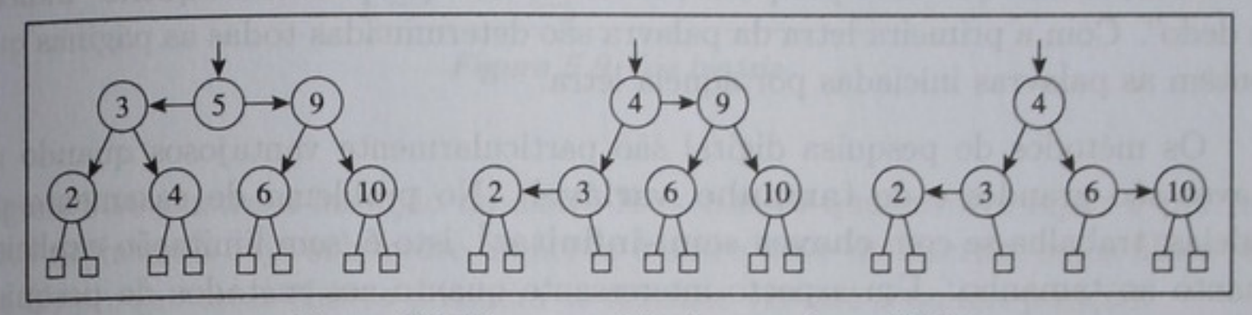

Figura 5.8 *Decomposição de uma árvore SBB.*

Análise Para as árvores SBB é necessário distinguir dois tipos de **altura**. Uma delas é a altura vertical h, necessária para manter a altura uniforme e obtida por meio da contagem do número de apontadores verticais em qualquer caminho entre a raiz e um nó externo. A outra é a altura k, que representa o número máximo de comparações de chaves obtidas mediante contagem do número total de apontadores no maior caminho entre a raiz e um nó externo. A altura k é maior que a altura h sempre que existirem apontadores horizontais na árvore. Para uma árvore SBB com n nós internos, temos:

$$h \leq k \leq 2h.$$

De fato, Bayer (1972) mostrou que:

$$log(n+1) \leq k \leq 2\log(n+2) - 2.$$

O custo para manter a propriedade SBB é exatamente o custo para percorrer o caminho de pesquisa para encontrar a chave, seja para inseri-la seja para retirá-la. Logo, esse custo é $O(\log n)$.

O número de comparações em uma pesquisa com sucesso na árvore SBB é:

$$\begin{array}{ll} \text{melhor caso} & : C(n) = O(1), \\ \text{pior caso} & : C(n) = O(\log n), \\ \text{caso médio} & : C(n) = O(\log n). \end{array}$$

Na prática, o caso médio para C_n é apenas cerca de 2% pior que o C_n para uma árvore completamente balanceada, conforme mostrado em Ziviani e Tompa (1982).