usando caminhamento central recupera as chaves na ordem 1, 2, 3, 4, 5, 6 e 7. O procedimento Central, mostrado no Programa 5.10, faz exatamente isso. Observe que este procedimento representa um método de ordenação similar ao Quicksort, no qual a chave na raiz faz o papel do item que particiona o vetor.

***Programa 5.10** Caminhamento central*

```
procedure Central (p: TipoApontador);
begin
  if p <> nil
  then begin
       Central (p^.Esq); writeln (p^.Reg.Chave); Central (p^.Dir);
       end;
end; { Central }
```

Análise O número de comparações em uma pesquisa com sucesso é:

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

O tempo de execução dos algoritmos para árvores binárias de pesquisa depende muito do formato das árvores. Para obter o pior caso, basta que as chaves sejam inseridas em ordem crescente (ou decrescente). Nesse caso, a árvore resultante é uma lista linear, cujo número médio de comparações é $(n+1)/2$.

Para uma **árvore de pesquisa randômica**[1] é possível mostrar que o número esperado de comparações para recuperar um registro qualquer é cerca de $1,39 \log n$, apenas 39% pior que a árvore completamente balanceada (vide seção seguinte).

5.3.2 Árvores Binárias de Pesquisa com Balanceamento

Para uma distribuição uniforme das chaves, em que cada chave é igualmente provável de ser usada em uma pesquisa, a **árvore completamente balanceada**[2] minimiza o tempo médio de pesquisa. Entretanto, o custo para manter a árvore

[1] Uma árvore A com n chaves possui $n+1$ nós externos e estas n chaves dividem todos os valores possíveis em $n+1$ intervalos. Uma inserção em A é considerada *randômica* se ela tem probabilidade igual de acontecer em qualquer um dos $n+1$ intervalos. Uma *árvore de pesquisa randômica* com n chaves é uma árvore construída por meio de n inserções randômicas sucessivas em uma árvore inicialmente vazia.

[2] Em uma árvore completamente balanceada, os nós externos aparecem em no máximo dois níveis adjacentes.

completamente balanceada após cada inserção é muito alto. Por exemplo, para inserir a chave 1 na árvore à esquerda na Figura 5.3 e obter a árvore à direita na mesma figura é necessário movimentar todos os nós da árvore original.

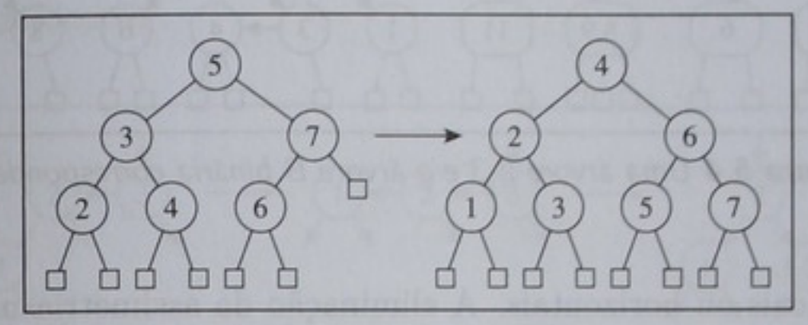

***Figura 5.3** Árvore binária de pesquisa completamente balanceada.*

Uma forma de contornar esse problema é procurar uma solução intermediária que possa manter a árvore "quase balanceada", em vez de tentar manter a árvore completamente balanceada. O objetivo é procurar obter bons tempos de pesquisa, próximos do tempo ótimo da árvore completamente balanceada, mas sem pagar muito para inserir ou retirar da árvore.

Existem inúmeras heurísticas baseadas no princípio acima. Gonnet e Baeza-Yates (1991) apresentam algoritmos que utilizam vários critérios de balanceamento para árvores de pesquisa, tais como restrições impostas na diferença das alturas de subárvores de cada nó da árvore, na redução do **comprimento do caminho interno**[3] da árvore, ou em que todos os nós externos aparecem no mesmo nível. Na seção seguinte, vamos apresentar uma árvore binária de pesquisa com balanceamento em que todos os nós externos aparecem no mesmo nível.

Árvores SBB

As **árvores B** foram introduzidas por Bayer e McCreight (1972) como uma estrutura para memória secundária, conforme mostrado em detalhes na Seção 6.3.1. Um caso especial da árvore B, mais apropriada para memória primária, é a **árvore 2-3**, na qual cada nó tem duas ou três subárvores. Bayer (1971) mostrou que as árvores 2-3 podem ser representadas por árvores binárias, conforme exibido na Figura 5.4.

Quando a árvore 2-3 é vista como uma **árvore B binária**, existe uma assimetria inerente no sentido de que os apontadores à esquerda têm de ser verticais (isto é, apontam para um nó no nível abaixo), enquanto os apontadores à direita

[3] O comprimento do caminho interno corresponde à soma dos comprimentos dos caminhos entre a raiz e cada um dos nós internos da árvore. Por exemplo, o comprimento do caminho interno da árvore à esquerda na Figura 5.3 é $8 = (0+1+1+2+2+2)$.