ここでも, 点 v の高さおよび深さをそれぞれ $height(v)$ および $depth(v)$ と書き, そのデータ構造の詳細には触れないでおく.

```
void  Height_and_Depth( T )
        入力 : T :     TREE (値呼び)
        出力 : 木 T の各点 v の高さ height(v) および深さ depth(v) : 整数
   /* 木 T を深さ優先探索し, 各点 v の子孫の高さおよび深さを求める */
NODE   v ;
{
        depth( Root(T) ) := 0 ;
        DFS_3( Root(T), T ) ;
} /* Height_and_Depth */ ;
```

```
void  DFS_3( v, T )
        入力 : v :     NODE (値呼び)
               T :     TREE (値呼び)
        出力 : 点 v の各子孫の高さ height(・) および深さ depth(・) : int
   /* 点 v を根とする部分木を深さ優先探索し, v の全ての子孫の高さおよび深さを求める */
NODE   w ;
{
        height(v) := 0 ;
        w := Leftmost_Child( v, T ) ;
        while ( w が存在する ) {
                depth(w) := depth(v) + 1 ;
                DFS_3( w, T ) ;
                height(v) := max[ height(v), height(w) + 1 ] ;
                w := Right_Sibling( w, T ) ;
        }
} /* DFS_3 */ ;
```