```
1       /**
2        * Internal method to merge two roots.
3        * Assumes trees are not empty, and h1's root contains smallest item.
4        */
5       private Node<AnyType> merge1( Node<AnyType> h1, Node<AnyType> h2 )
6       {
7           if( h1.left == null )   // Single node
8               h1.left = h2;       // Other fields in h1 already accurate
9           else
10          {
11              h1.right = merge( h1.right, h2 );
12              if( h1.left.npl < h1.right.npl )
13                  swapChildren( h1 );
14              h1.npl = h1.right.npl + 1;
15          }
16          return h1;
17      }
```

图 6-27　合并左式堆的实际例程

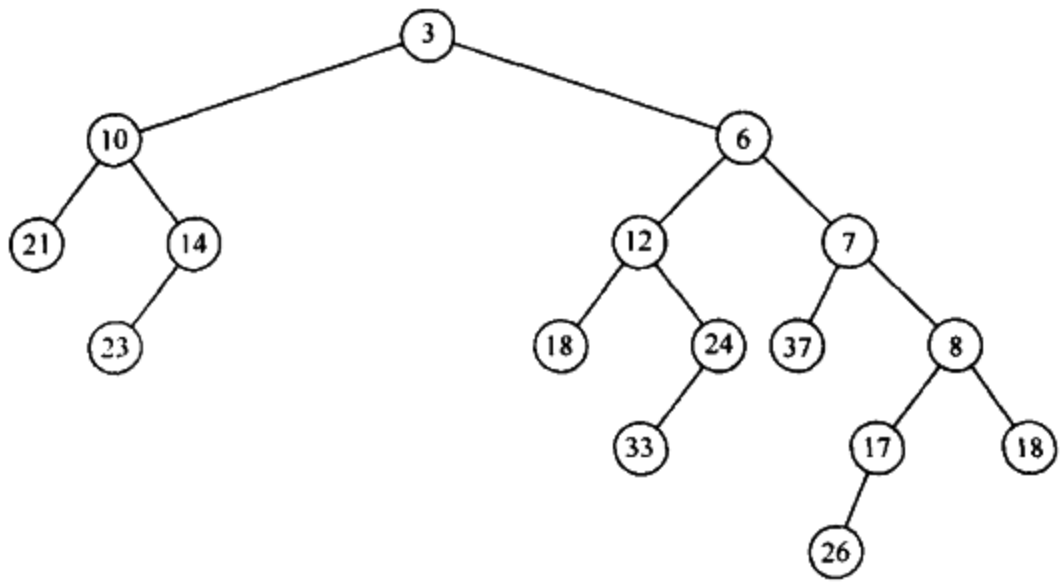


图 6-28　合并 $H_1$ 和 $H_2$ 的右路径的结果

```
1       /**
2        * Insert into the priority queue, maintaining heap order.
3        * @param x the item to insert.
4        */
5       public void insert( AnyType x )
6       {
7           root = merge( new Node<AnyType>( x ), root );
8       }
```

图 6-29　左式堆的插入例程

最后，我们可以通过建立一个二叉堆(显然使用链接实现)来以 O(N)时间建立一个左式堆。尽管二叉堆显然是左式的，但是，这未必是最佳解决方案，因为我们得到的堆可能是最差的左式堆。不仅如此，以相反的层序遍历树用一些链来进行也不那么容易。buildHeap 的效果可以通过递归地建立左右子树然后将根下滤而达到。练习中包括另外一个解决方案。