```
                theTrees[ i ] = null;
                break;
              case 6: /* rhs and carry */
                carry = combineTrees( t2, carry );
                rhs.theTrees[ i ] = null;
                break;
              case 7: /* All three */
                theTrees[ i ] = carry;
                carry = combineTrees( t1, t2 );
                rhs.theTrees[ i ] = null;
                break;
            }
        }

        for( int k = 0; k < rhs.theTrees.length; k++ )
            rhs.theTrees[ k ] = null;
        rhs.currentSize = 0;
    }
```

图 6-55 （续）

```
    /**
     * Remove the smallest item from the priority queue.
     * @return the smallest item, or throw UnderflowException if empty.
     */
    public AnyType deleteMin( )
    {
        if( isEmpty( ) )
            throw new UnderflowException( );

        int minIndex = findMinIndex( );
        AnyType minItem = theTrees[ minIndex ].element;

        Node<AnyType> deletedTree = theTrees[ minIndex ].leftChild;

        // Construct H''
        BinomialQueue<AnyType> deletedQueue = new BinomialQueue<AnyType>( );
        deletedQueue.expandTheTrees( minIndex + 1 );

        deletedQueue.currentSize = ( 1 << minIndex ) - 1;
        for( int j = minIndex - 1; j >= 0; j-- )
        {
            deletedQueue.theTrees[ j ] = deletedTree;
            deletedTree = deletedTree.nextSibling;
            deletedQueue.theTrees[ j ].nextSibling = null;
        }

        // Construct H'
        theTrees[ minIndex ] = null;
        currentSize -= deletedQueue.currentSize + 1;

        merge( deletedQueue );
```

图 6-56 二项队列的 deleteMin，用到 findMinIndex 方法