的，从而任何复杂的尝试就都不值得考虑了。

```
1       /**
2        * Find an item in the hash table.
3        * @param x the item to search for.
4        * @return true if x is not found.
5        */
6       public boolean contains( AnyType x )
7       {
8           List<AnyType> whichList = theLists[ myhash( x ) ];
9           return whichList.contains( x );
10      }
11
12      /**
13       * Insert into the hash table. If the item is
14       * already present, then do nothing.
15       * @param x the item to insert.
16       */
17      public void insert( AnyType x )
18      {
19          List<AnyType> whichList = theLists[ myhash( x ) ];
20          if( !whichList.contains( x ) )
21          {
22              whichList.add( x );
23
24                  // Rehash; see Section 5.5
25              if( ++currentSize > theLists.length )
26                  rehash( );
27          }
28      }
29
30      /**
31       * Remove from the hash table.
32       * @param x the item to remove.
33       */
34      public void remove( AnyType x )
35      {
36          List<AnyType> whichList = theLists[ myhash( x ) ];
37          if( whichList.contains( x ) )
38          {
39              whichList.remove( x );
40              currentSize--;
41          }
42      }
```

图 5-10　分离链接散列表的 contains 例程、insert 例程和 remove 例程

我们定义散列表的装填因子(load factor)λ为散列表中的元素个数对该表大小的比。在上面的例子中，λ=1.0。链表的平均长度为 λ。执行一次查找所需要的工作是计算散列函数值所需要的常数时间加上遍历链表所用的时间。在一次不成功的查找中，要考查的节点数平均为 λ。一次成功的查找则需要遍历大约 1 + (λ/2)个链。为了看清这一点，注意被搜索的链表包含一个存储