

10.2-5.

Implement the dictionary operations INSERT, DELETE, and SEARCH using singly linked, circular lists. What are the running times of your procedures?

Answer.

As shown in Figure 1, each element of a circular, singly linked list L is an object with an attribute

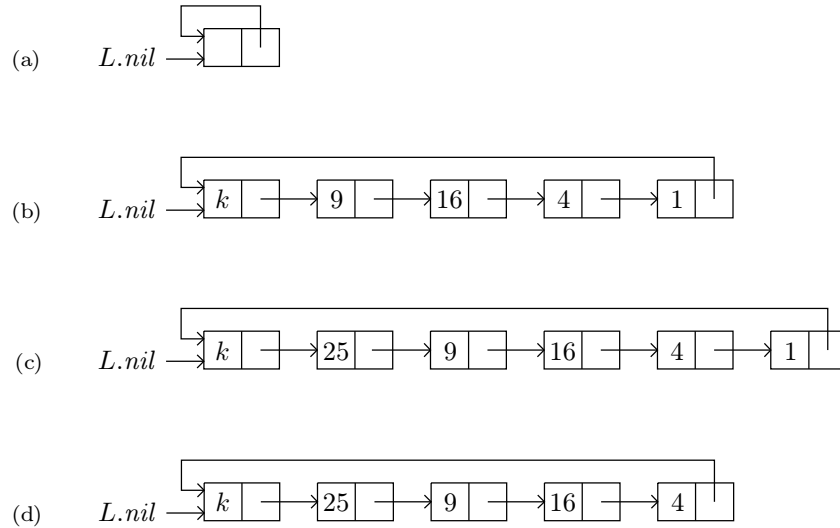


Figure 1. A circular, singly linked list with a sentinel. The sentinel $L.nil$ appears between the head and tail. The attribute $L.head$ is no longer needed, since we can access the head of the list by $L.nil.next$. (a) An empty list. (b) The linked list from Figure 10.3(a), with key 9 at the head and key 1 at the tail. (c) The list after executing $LIST-INSERT''(L, x)$, where $x.key = 25$. The new object becomes the head of the list. (d) The list after deleting the object with key 1. The new tail is the object with key 4.

key and an attribute of *next* pointer. The list also equips with a sentinel $L.nil$, which lies between the head and tail. The attribute $L.nil.next$ points to the head of the list, and the *next* pointer of the last element points to this sentinel.

The code for SEARCH remains almost identical to $LIST-SEARCH'$, except that we put the key k of an element to search in the sentinel to eliminate check for the tail of a list (see exercise 10.2-4).

$LIST-SEARCH''(L, k)$

```


1   $L.nil.key = k$ 
2   $x = L.nil.next$ 
3  while  $x.key \neq k$ 
4       $x = x.next$ 
5  return  $x$ 
```

The following procedure inserts an element into the list:

$LIST-INSERT''(L, x)$

```

1   $x.next = L.nil.next$ 
2   $L.nil.next = x$ 
```

*. Creative Commons  2014, Lawrence X. Amlord (颜世敏, aka 颜序).
Email address: informlarry@gmail.com

In order to maintain the chain after deleting an element, a traversal on the list for its predecessor is required.

```
LIST-DELETE''( $L, x$ )
1   $pre = L.nil$ 
2  while  $pre.next.key \neq x$ 
3      if  $pre.next == L.nil$ 
4          error "element not exist"
5       $pre = pre.next$ 
6   $pre.next = x.next$ 
```

Figure 1 shows the effects of LIST-INSERT'' and LIST-DELETE'' on a sample list. To search a list of n objects, both LIST-SEARCH'' and LIST-DELETE'' take $\Theta(n)$ time in the worst case, since they may have to search the entire list. The running time for LIST-INSERT'' remains the same, $\Theta(1)$.