10.3-4.

It is often desirable to keep all elements of a doubly linked list compact in storage, using, for example, the first m index locations in the multiple-array representation. (This is the case in the paged, virtual-memory computing environment.) Explain how to impoement the procedures ALLOCATE-OBJECT and FREE-OBJECT so that the representation is compact. Assume that there are no pointers to elements of linked list outside the list itself. (*Hint:* Use the array implementation of a stack.)

Answer.

Suppose that at some moment the dynamic set contains $m \leq n$ elements and they are stored compact. Then the first m objects represent elements currently in the dynamic set, and the remaining $n - m$ objects are free.

We keep the free objects in an array implementation of a stack $F[F.top..m]$, where $F[m]$ is the element at the bottom of the stack and $F[F.top]$ is the element at the top. The free list uses an array implementation of the stack operations PUSH and POP to implement the procedures for allocating and freeing objects, respectively. We assume that the global variable F used in the following procedures points to the stack F.

```
ALLOCATE-OBJECT()
if STACK-EMPTY($F$)
    error "out of space"
else $x$ = POP($F$)
    return $x$
```

```
FREE-OBJECT($x$)
$p = F.top - 1$
$p.prev.next = x$
$p.next.prev = x$
$x.key = p.key$
$x.prev = p.prev$
$x.next = p.next$
PUSH($F, p$)
```

The free list initially contains all m unallocated objects. Once the free list has been exhausted, running the ALLOCATE-OBJECT procedure signals an error. To free an object at index x, we first override this slot by the object p left to $F.top$ in the doubly linked list, then join p into the free list, as Figure 1 shows. Both procedures run in $O(1)$ time, which makes them quite efficient.

Email address: informlarry@gmail.com

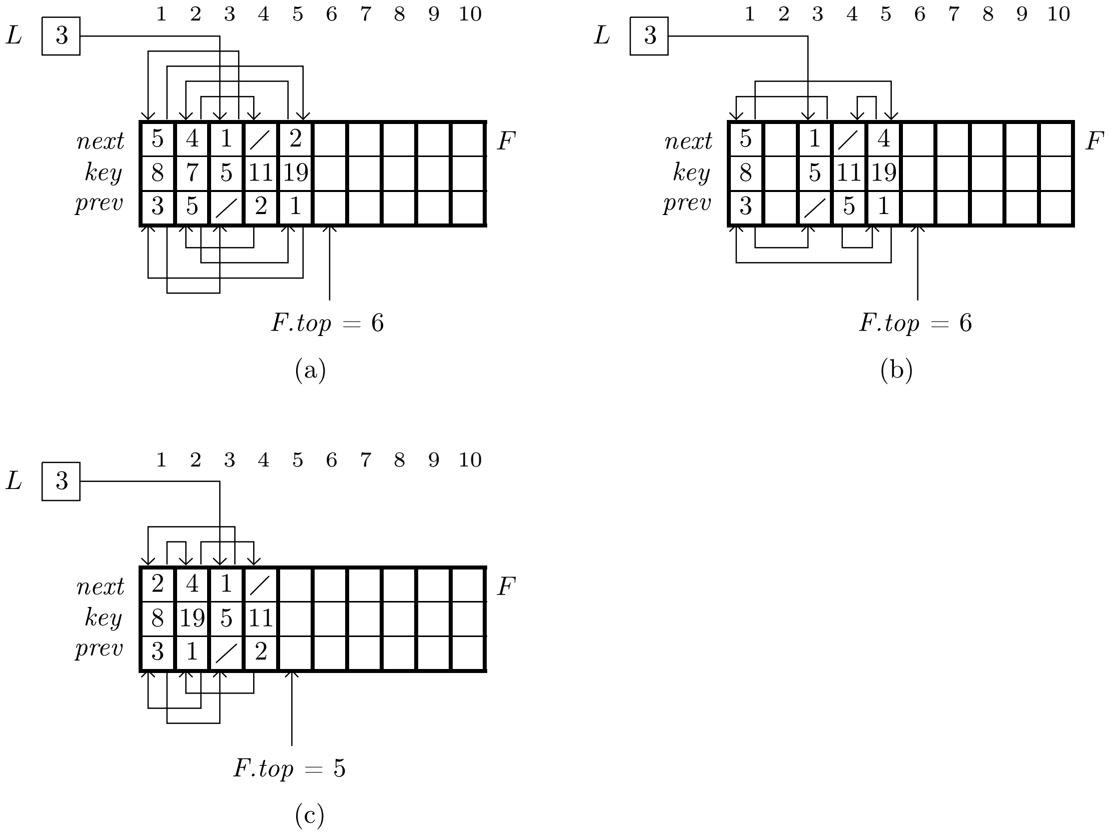

Figure 1. The effect of the FREE-OBJECT procedures. **(a)** The dynamic set $\{5, 8, 19, 7, 11\}$ represented by a doubly linked list, and a free list (first rows of empty slots) whose top element is at index 6. **(b)** The result of deleting the object at index 2, leaving the storage of L truncated by an empty slot. **(c)** After executing FREE-OBJECT(2), the object at index 5 is reallocated to index 2, making L a compact list again.