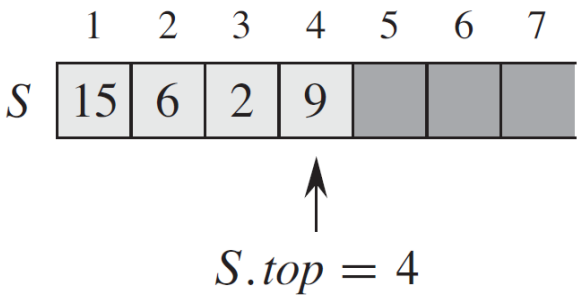
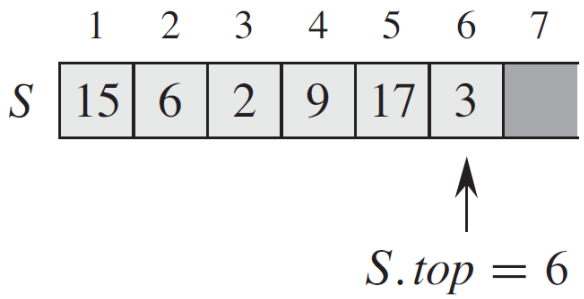
Elementary Data Structures

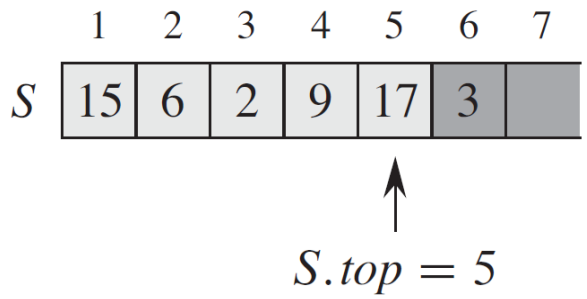
Stacks


1 2 3 4 5 6 7
S
15 6 2 9
S.top = 4

(a)


1 2 3 4 5 6 7
S
15 6 2 9 17 3
S.top = 6

(b)


1 2 3 4 5 6 7
S
15 6 2 9 17 3
S.top = 5

(c)

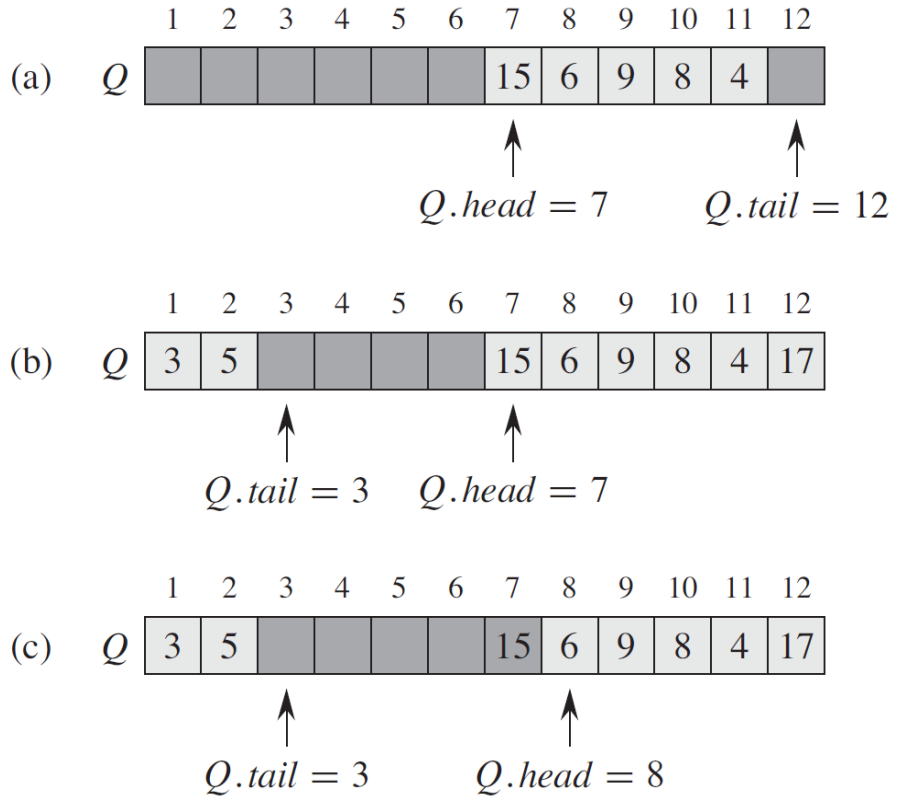
Stacks

```
STACK-EMPTY(S)
1  if S.top == 0
2       return TRUE
3  else return FALSE

PUSH(S, x)
1  S.top = S.top + 1
2  S[S.top] = x

POP(S)
1  if STACK-EMPTY(S)
2       error "underflow"
3  else S.top = S.top − 1
4       return S[S.top + 1]
```

Queues

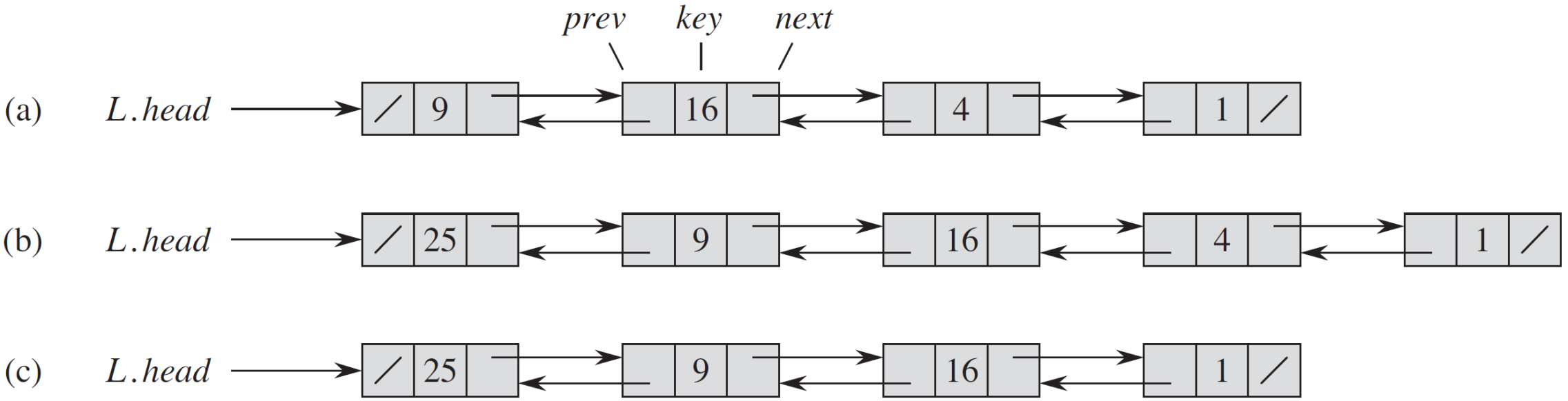

1 2 3 4 5 6 7 8 9 10 11 12
(a) Q
15 6 9 8 4
Q.head = 7
Q.tail = 12
1 2 3 4 5 6 7 8 9 10 11 12
(b) Q
3 5 15 6 9 8 4 17
Q.tail = 3
Q.head = 7
1 2 3 4 5 6 7 8 9 10 11 12
(c) Q
3 5 15 6 9 8 4 17
Q.tail = 3
Q.head = 8

Queues

```
ENQUEUE(Q, x)
1  Q[Q.tail] = x
2  if Q.tail == Q.length
3        Q.tail = 1
4  else Q.tail = Q.tail + 1
```

```
DEQUEUE(Q)
1  x = Q[Q.head]
2  if Q.head == Q.length
3        Q.head = 1
4  else Q.head = Q.head + 1
5  return x
```

Linked Lists



Linked Lists

```
LIST-SEARCH(L, k)
1   x = L.head
2   while x ≠ NIL and x.key ≠ k
3       x = x.next
4   return x
```

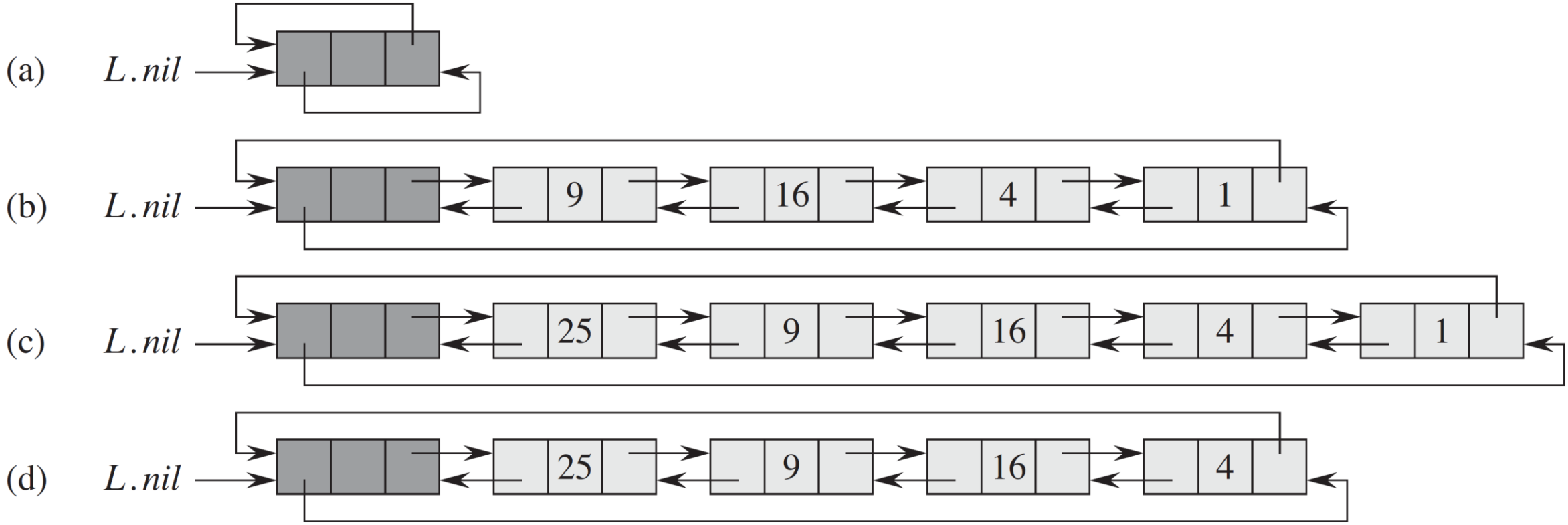
Linked Lists

```
LIST-INSERT(L, x)
1  x.next = L.head
2  if L.head ≠ NIL
3      L.head.prev = x
4  L.head = x
5  x.prev = NIL
```

Linked Lists

```
LIST-DELETE(L, x)
1   if x.prev ≠ NIL
2       x.prev.next = x.next
3   else L.head = x.next
4   if x.next ≠ NIL
5       x.next.prev = x.prev
```

Linked Lists - Sentinels



Linked Lists with sentinels

```
LIST-SEARCH′(L, k)
1   x = L.nil.next
2   while x ≠ L.nil and x.key ≠ k
3       x = x.next
4   return x
```

Linked Lists with sentinels

$\text{LIST-INSERT}'(L, x)$

1 $x.next = L.nil.next$
2 $L.nil.next.prev = x$
3 $L.nil.next = x$
4 $x.prev = L.nil$

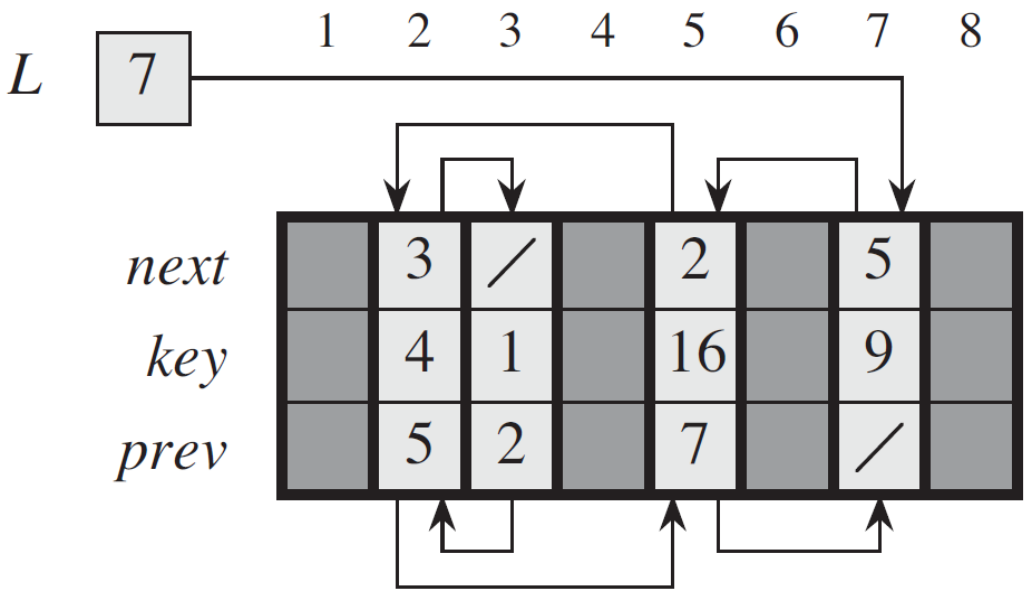
Linked Lists with sentinels

$$\textsc{List-Delete}'(L, x)$$

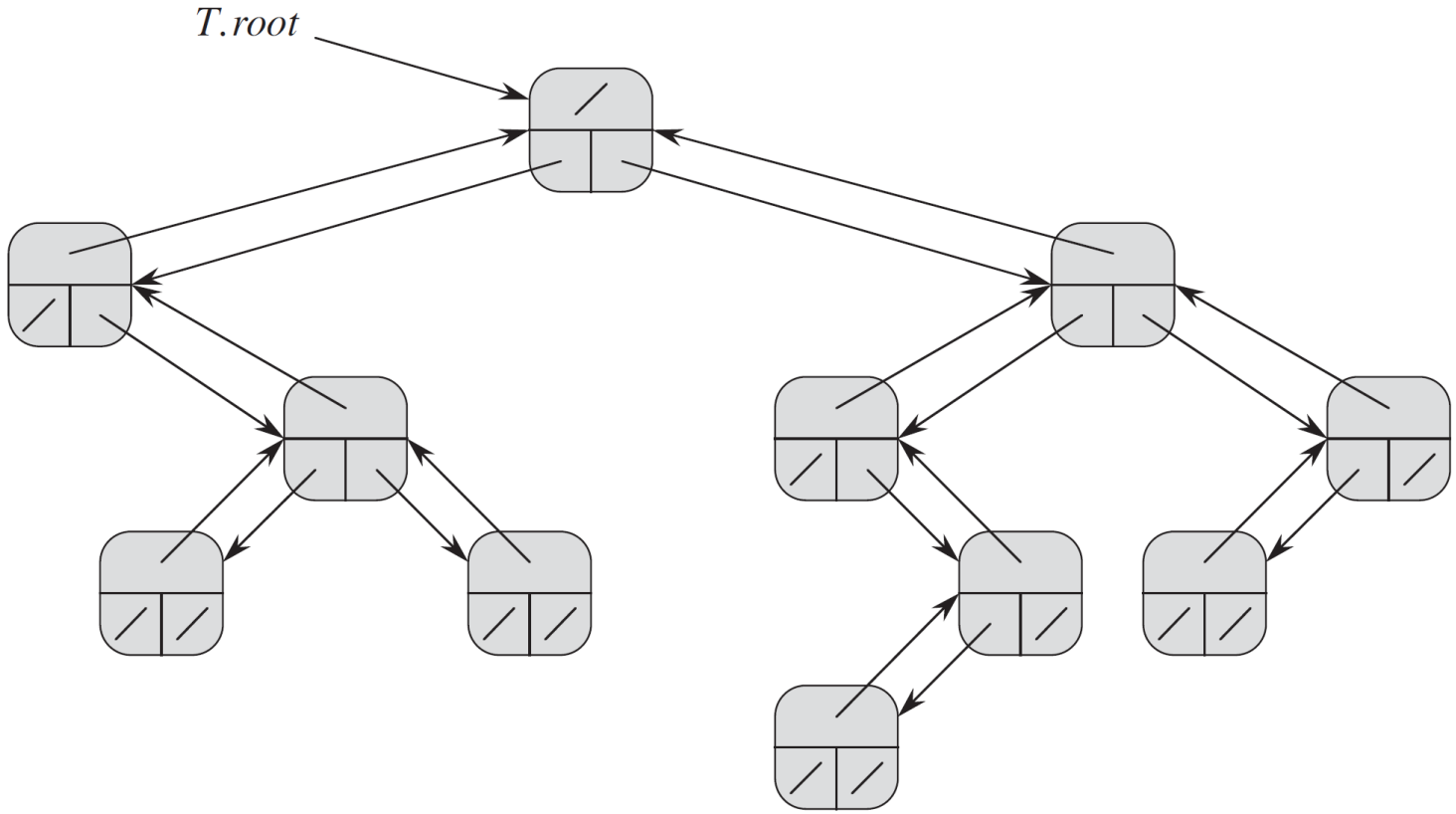
$$1 \quad x.prev.next = x.next$$

$$2 \quad x.next.prev = x.prev$$

Implementation of linked lists by arrays



Rooted trees



Rooted trees