Data Structures and Algorithms

Fourth Practice

Stack representation

Let S be a stack containing numbers. We can implement a stack of at most n elements with an array S[1, . . . ,n]. The array has an attribute **S.top** that indexes the most recently inserted element.

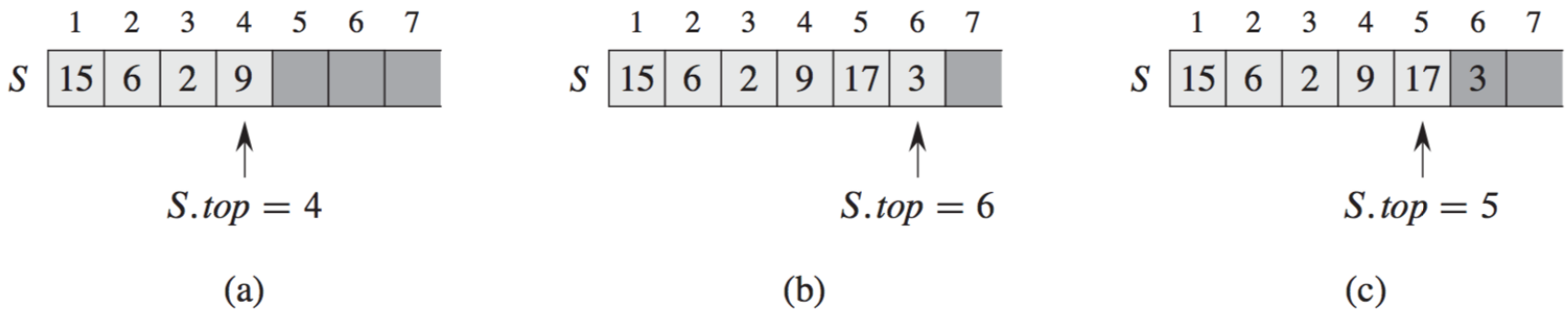


Figure 10.1 An array implementation of a stack S. Stack elements appear only in the lightly shaded positions. **(a)** Stack S has 4 elements. The top element is 9. **(b)** Stack S after the calls PUSH$(S, 17)$ and PUSH$(S, 3)$. **(c)** Stack S after the call POP(S) has returned the element 3, which is the one most recently pushed. Although element 3 still appears in the array, it is no longer in the stack; the top is element 17.

Stack representation – exercise 1

Using Figure 10.1 as a model, illustrate the result of each operation in the sequence

PUSH(S,3)
PUSH(S,9)
PUSH(S,2)
POP(S)
PUSH(S,1)
POP(S)
POP(S)
PUSH(S,4)
PUSH(S,7)

on an initially empty stack S stored in array S[1...6].

Stack representation – exercise 1, solution

Operation	1	2	3	4	5	6	S.top	return value
PUSH(S,3)	3						1	-
PUSH(S,9)	3	9					2	-
PUSH(S,2)	3	9	2				3	-
POP(S)	3	9					2	2
PUSH(S,1)	3	9	1				3	-
POP(S)	3	9					2	1
POP(S)	3						1	9
PUSH(S,4)	3	4					2	-
PUSH(S,7)	3	4	7				3	-

Stack representation – exercise 2

Using Figure 10.1 as a model, illustrate the result of each operation in the sequence

PUSH(S,1)
PUSH(S,5)
POP(S)
PUSH(S,2)
POP(S)
POP(S)
PUSH(S,3)
POP(S)
POP(S)

on an initially empty stack S stored in array S[1...5].

Stack representation – exercise 3

Using Figure 10.1 as a model, illustrate the result of each operation in the sequence

PUSH(S,8)
PUSH(S,6)
POP(S)
PUSH(S,5)
PUSH(S,1)
PUSH(S,9)
POP(S)
PUSH(S,2)
PUSH(S,4)
PUSH(S,7)

on an initially empty stack S stored in array S[1...5].

Stack representation – exercise 4

Implement two stacks in one array A[1...n] in such a way that neither stack overflows unless the total number of elements in both stacks together is n. The PUSH and POP operations should run in $\mathcal{O}(1)$ time.

Stack representation – exercise 4, solution

We will call the stacks T and R. Initially, set $T.top = 0$ and $R.top = n + 1$. Essentially, stack T uses the first part of the array and stack R uses the last part of the array. In stack T, the top is the rightmost element of T. In stack R, the top is the leftmost element of R.

Algorithm 1 PUSH(S,x)

```
1: if S == T then
2:     if T.top + 1 == R.top then
3:         error "overflow"
4:     else
5:         T.top = T.top + 1
6:         T[T.top] = x
7:     end if
8: end if
9: if S == R then
10:    if R.top − 1 == T.top then
11:        error "overflow"
12:    else
13:        R.top = R.top − 1
14:        T[T.top] = x
15:    end if
16: end if
```

Queue representation

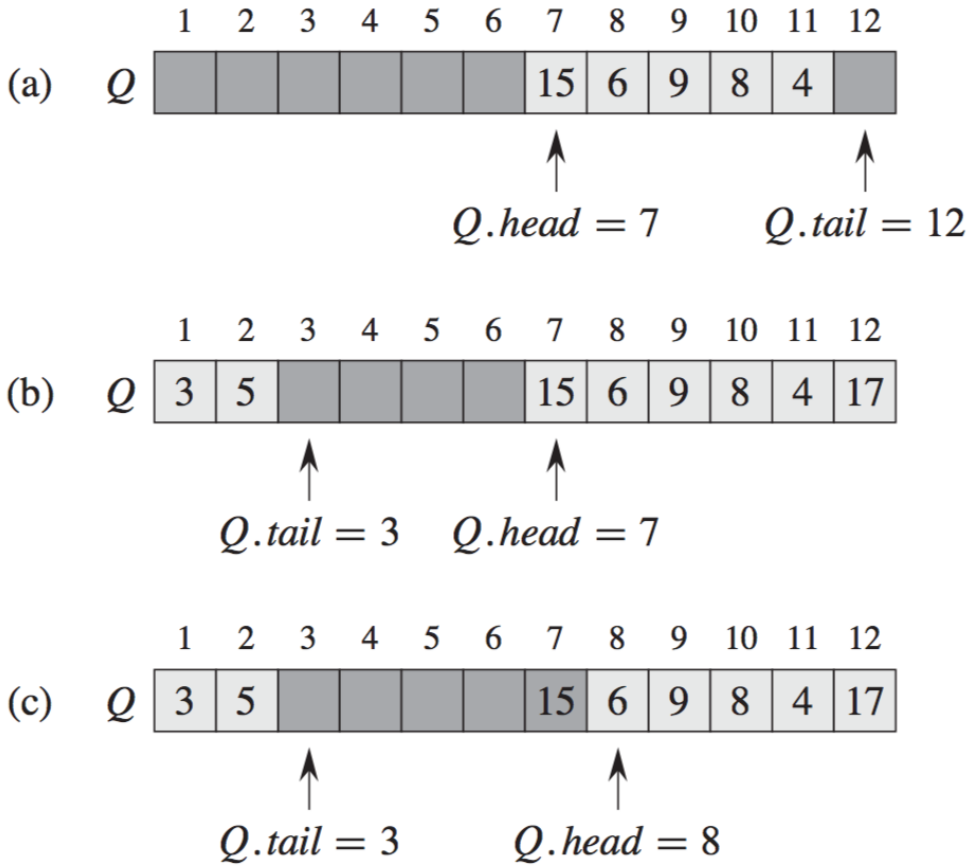


Figure 10.2 A queue implemented using an array $Q[1..12]$. Queue elements appear only in the lightly shaded positions. **(a)** The queue has 5 elements, in locations $Q[7..11]$. **(b)** The configuration of the queue after the calls ENQUEUE$(Q, 17)$, ENQUEUE$(Q, 3)$, and ENQUEUE$(Q, 5)$. **(c)** The configuration of the queue after the call DEQUEUE(Q) returns the key value 15 formerly at the head of the queue. The new head has key 6.

Queue representation – exercise 1

Using Figure 10.2 as a model, illustrate the result of each operation in the sequence

ENQUEUE(Q,4)
DEQUEUE(Q)
ENQUEUE(Q,7)
ENQUEUE(Q,2)
ENQUEUE(Q,9)
ENQUEUE(Q,5)
ENQUEUE(Q,3)
DEQUEUE(Q)
ENQUEUE(Q,1)

on an initially empty cyclic queue Q stored in array Q[1...6].

Queue representation – exercise 1, solution

Operation	1	2	3	4	5	6	Q.head	Q.tail	return value
ENQUEUE(Q,4)	4						1	2	-
DEQUEUE(Q)							2	2	4
ENQUEUE(Q,7)		7					2	3	-
ENQUEUE(Q,2)		7	2				2	4	-
ENQUEUE(Q,9)		7	2	9			2	5	-
ENQUEUE(Q,5)		7	2	9	5		2	6	-
ENQUEUE(Q,3)		7	2	9	5	3	2	1	-
DEQUEUE(Q)			2	9	5	3	3	1	7
ENQUEUE(Q,1)	1		2	9	5	3	3	2	-

Queue representation – exercise 2

Using Figure 10.2 as a model, illustrate the result of each operation in the sequence

ENQUEUE(Q,5)
ENQUEUE(Q,1)
ENQUEUE(Q,7)
DEQUEUE(Q)
ENQUEUE(Q,3)
ENQUEUE(Q,8)
DEQUEUE(Q)
ENQUEUE(Q,2)
ENQUEUE(Q,4)

on an initially empty cyclic queue Q stored in array Q[1...5].

Queue representation – exercise 3

Using Figure 10.2 as a model, illustrate the result of each operation in the sequence

ENQUEUE(Q,a)
ENQUEUE(Q,c)
DEQUEUE(Q)
ENQUEUE(Q,b)
DEQUEUE(Q)
DEQUEUE(Q)
ENQUEUE(Q,d)
DEQUEUE(Q)
DEQUEUE(Q)

on an initially empty cyclic queue Q stored in array Q[1...7].

Queue representation – exercise 4

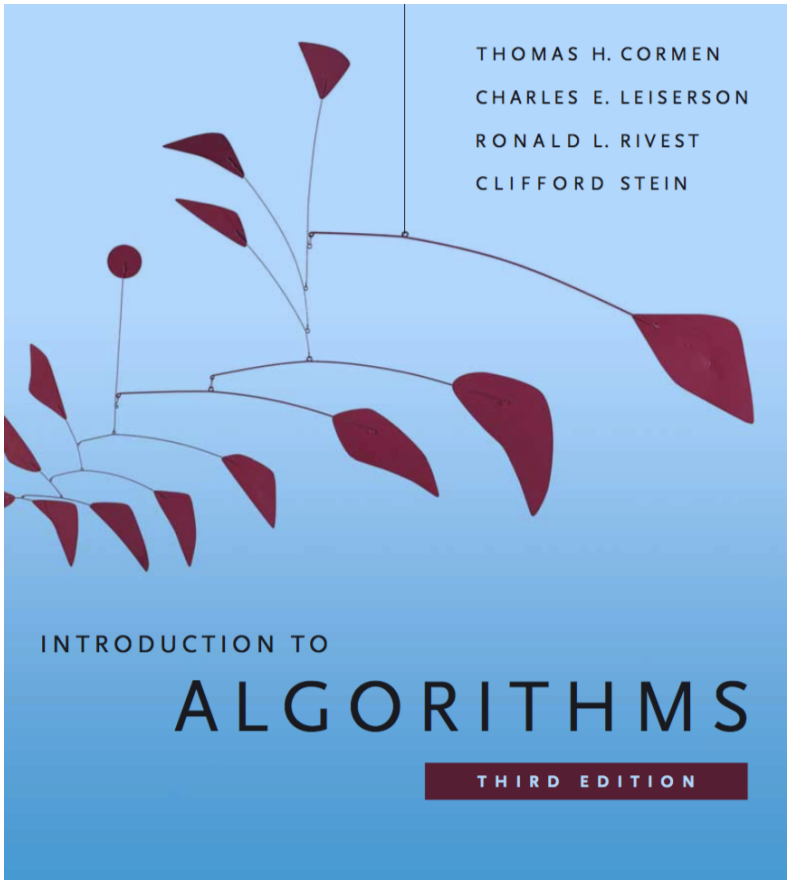
Rewrite ENQUEUE and DEQUEUE to detect underflow and overflow of a queue.

Queue representation – exercise 4, solution

Algorithm 3 ENQUEUE

if $Q.head == Q.tail + 1$, or $Q.head == 1$ and $Q.tail == Q.length$ **then**
    error "overflow"
end if
$Q[Q.tail] = x$
if $Q.tail == Q.length$ **then**
    $Q.tail = 1$
else
    $Q.tail = Q.head + 1$
end if

References


THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST
CLIFFORD STEIN
INTRODUCTION TO
ALGORITHMS
THIRD EDITION