Prof. Pietro S. Oliveto

CS217 Data Structures and Algorithm Analysis
Fall term 2025

Solutions for Exercise Sheet 8

Handout: Oct 28th — Deadline: Nov 4th 4pm

Question 8.1 (0.25 marks)

Draw the following data structures after each of the following operations. Assume that the data structures are initially empty. You don't need to draw pointers for stacks and queues.

1. Consider a stack S and the operations $\textsc{Push}(S,7)$, $\textsc{Push}(S,4)$, $\textsc{Push}(S,5)$, $\textsc{Pop}(S)$, $\textsc{Push}(S,8)$, $\textsc{Pop}(S)$, $\textsc{Pop}(S)$.
2. Consider a queue Q and the operations $\textsc{Enqueue}(Q,7)$, $\textsc{Enqueue}(Q,4)$, $\textsc{Enqueue}(Q,5)$, $\textsc{Dequeue}(Q)$, $\textsc{Enqueue}(Q,8)$, $\textsc{Dequeue}(Q)$, $\textsc{Dequeue}(Q)$.
3. Consider a singly-linked list L and the operations $\textsc{List-Prepend}(L,7)$, $\textsc{List-Prepend}(L,4)$, $\textsc{List-Prepend}(L,5)$, $\textsc{List-Delete}(L,4)$, $\textsc{List-Prepend}(L,8)$, $\textsc{List-Delete}(L,7)$, $\textsc{List-Delete}(L,8)$.

Solution: You can draw stacks vertically or horizontally, in the context of the array implementation or not. Here we draw them vertically.

Stack:

$\textsc{Push}(S,7)$: $\begin{array}{|c|}\hline 7\\\hline\end{array}$ $\quad$ $\textsc{Push}(S,4)$: $\begin{array}{|c|}\hline 4\\\hline 7\\\hline\end{array}$ $\quad$ $\textsc{Push}(S,5)$: $\begin{array}{|c|}\hline 5\\\hline 4\\\hline 7\\\hline\end{array}$

$\textsc{Pop}(S)$: $\begin{array}{|c|}\hline 4\\\hline 7\\\hline\end{array}$ $\quad$ $\textsc{Push}(S,8)$: $\begin{array}{|c|}\hline 8\\\hline 4\\\hline 7\\\hline\end{array}$ $\quad$ $\textsc{Pop}(S)$: $\begin{array}{|c|}\hline 4\\\hline 7\\\hline\end{array}$ $\quad$ $\textsc{Pop}(S)$: $\begin{array}{|c|}\hline 7\\\hline\end{array}$

Queue:

$\textsc{Enqueue}(Q,7)$: $\begin{array}{|c|}\hline 7\\\hline\end{array}$ $\quad$ $\textsc{Enqueue}(Q,4)$: $\begin{array}{|c|c|}\hline 7 & 4\\\hline\end{array}$ $\quad$ $\textsc{Enqueue}(Q,5)$: $\begin{array}{|c|c|c|}\hline 7 & 4 & 5\\\hline\end{array}$

$\textsc{Dequeue}(Q)$: $\begin{array}{|c|c|}\hline 4 & 5\\\hline\end{array}$ $\quad$ $\textsc{Enqueue}(Q,8)$: $\begin{array}{|c|c|c|}\hline 4 & 5 & 8\\\hline\end{array}$ $\quad$ $\textsc{Dequeue}(Q)$: $\begin{array}{|c|c|}\hline 5 & 8\\\hline\end{array}$

$\textsc{Dequeue}(Q)$: $\begin{array}{|c|}\hline 8\\\hline\end{array}$

Linked List:

$\textsc{List-Prepend}(L,7)$: $L.\text{head}\rightarrow\boxed{7}$ $\quad$ $\textsc{List-Prepend}(L,4)$: $L.\text{head}\rightarrow\boxed{4}\rightarrow\boxed{7}$

$\textsc{List-Prepend}(L,5)$: $L.\text{head}\rightarrow\boxed{5}\rightarrow\boxed{4}\rightarrow\boxed{7}$ $\quad$ $\textsc{List-Delete}(L,4)$: $L.\text{head}\rightarrow\boxed{5}\rightarrow\boxed{7}$

$\textsc{List-Prepend}(L,8)$: $L.\text{head}\rightarrow\boxed{8}\rightarrow\boxed{5}\rightarrow\boxed{7}$ $\quad$ $\textsc{List-Delete}(L,7)$: $L.\text{head}\rightarrow\boxed{8}\rightarrow\boxed{5}$

$\textsc{List-Delete}(L,8)$: $L.\text{head}\rightarrow\boxed{5}$

Question 8.2 (0.5 marks) Explain how to implement two stacks S_1 and S_2 in one array $A[1:n]$ in such a way that neither stack overflows unless all the n elements of A are full. Present the pseudocodes for operations $\textsc{PushS}_1(A,x)$,$\textsc{PushS}_2(A,x)$,$\textsc{PopS}_1(A)$, and $\textsc{PopS}_2(A)$.

Solution: The first stack starts at index 1 and grows up towards index n, while the second starts from index n and grows down towards index 1. Stack overflow happens when an element is pushed when the two stack pointers are adjacent.

PushS1(A, x)

```
if S1.top == S2.top −1 then
    error "overflow"
else
    S1.top = S1.top +1
    A[S1.top] = x
```

PushS2(A, x)

```
if S1.top == S2.top −1 then
    error "overflow"
else
    S2.top = S2.top -1
    A[S2.top] = x
```

PopS1(A)

```
if Stack-Empty(S1) then
    error "underflow"
else
    S1.top = S1.top -1
    return A[S1.top +1]
```

PopS2(A)

```
if S2.top == n+1 then
    error "underflow"
else
    S2.top = S2.top +1
    return A[S2.top -1]
```

Question 8.3 (0.25 marks) Rewrite Enqueue and Dequeue to detect underflow and overflow of a queue.

Solution: If head and tail point to the same position in the stack, then the queue is empty (as when the queue is initialised before any elements are enqueued). Once the tail is in the position behind the head no more elements can be inserted. Otherwise the tail would match the head and the array would look empty while it is full instead.

Enqueue(Q, x)

```
if Q.head == (Q.tail + 1) then
    error "overflow"
else if Q.head == 1 and Q.tail == n then
    error "overflow"
else
    ...
```

Dequeue(Q)

```
if Q.head == Q.tail then
    error "underflow"
else
    ...
```

Question 8.4 (0.5 marks) Show how to implement a Queue using 2 stacks S_1 and S_2. Provide the pseudo-code of the operations Enqueue and Dequeue. You don't need to check for underflow and overflow. Analyse the runtime of the two operations.

Solution: Enqueue by pushing on S_1. To dequeue you need to pop all elements from S_1 to reach the head: push these elements to S_2. Now S_2 has the queue elements in reverse order: the head on top and the tail at the bottom. Form now on always enqueue by pushing to S_1 and dequeue by popping from S_2. Once S_2 is empty move all elements from S_1 to S_2 again at the next dequeue call.

ENQUEUE(Q, x)

```
PUSH(S_1, x)
```

DEQUEUE(Q)

```
if STACK-EMPTY(S_2) then
        while !STACK-EMPTY(S_1) do
                PUSH(S_2, POP(S_1))
return POP(S_2)
```

Question 8.5 (0.5 marks) Implement an Integer Calculator that takes a postfix expression in input using integers as operands and $\{+, -, *\}$ as operators. The algorithm should use a stack. (See Judge)

Question 8.6 (1 mark) Implement the problems "Finding Adjacent Value" and "Jet Bridge Allocation" on the Judge system.