CS3000: Algorithms & Data — Summer '25 — Laney Strange

Stacks and Queues Pseudocode

We went over the basic operations in class to insert and remove elements from a Queue and a Stack; this handout is so you can see the algorithms typeset in the CLRS pseudocode style.

For the Queue data structure, we implement ENQUEUE and DEQUEUE algorithms to insert and remove elements respectively. We refer to the queue itself with the variable name Q and maintain the head of the Queue, the tail of the Queue, and the maximum size. During the ENQUEUE operation, if the tail extends beyond the maximum length of the queue, it wraps around to start again at position 1.

For the Stack data structure, we implement PUSH, POP, and STACK-EMPTY algorithms to insert and remove elements. We refer to the stack itself with the variable name S and maintain the top of the Stack and the maximum size. During the PUSH operation, if the expected new top of the Stack would exceed the maximum size, we throw an overflow error. During the POP operation, we throw an underflow error if the stack is empty.

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

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

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

```
PUSH(S, x)
1  if S.top == S.size
2        error "overflow"
3  else
4        S.top = S.top + 1
5        S[S.top] = x
```

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

Here's how we typeset the above functions in CLRS style:

```
\begin{codebox}
\Procname{$\proc{Enqueue}(Q, x)$}
\li $Q[Q.\id{tail}] \gets x$
\li \If $Q.\id{tail} == Q.\id{size}$
\Then
\li $Q.\id{tail} \gets 1$
\End
\li \Else
\Then
\li $Q.\id{tail} \gets Q.\id{tail} + 1$
\End
\end{codebox}

\begin{codebox}
\Procname{$\proc{Dequeue}(Q)$}
\li $x \gets Q[Q.\id{head}]$
\li \If $Q.\id{head} == Q.\id{size}$
\Then
\li $Q.\id{head} \gets 1$
\End
\li \Else
\Then
\li $Q.\id{head} \gets Q.\id{head} + 1$
\End
\li \Return $x$
\end{codebox}

\begin{codebox}
\Procname{$\proc{Stack-Empty}(S)$}
\li \If $S.\id{top} == 0$
\Then
\li \Return \const{True}
\End
\li \Else
\Then
\li \Return \const{False}
\End
\end{codebox}
```

```
\begin{codebox}
\Procname{$\proc{Push}(S, x)$}
\li \If $S.\id{top} == S.size$
\Then
\li \Error "overflow"
\End
\li \Else
\Then
\li $S.\id{top} \gets S.\id{top} + 1$
\li $S[S.\id{top}] \gets x$
\End
\end{codebox}

\begin{codebox}
\Procname{$\proc{Pop}(S)$}
\li \If $\proc{Stack-Empty}(S)$
\Then
\li \Error "underflow"
\End
\li \Else
\Then
\li $S.\id{top} \gets S.\id{top} - 1$
\li \Return $S[S.\id{top} + 1]$
\End
\end{codebox}
```