Chapter 10 Elementary Data Structures

By Y. Meng gmail: y(dot)meng201011

Stacks and queues (*stack and queue are dynamic sets.*)

★ Stacks.

◇ Stack implements a last-ine, first-out (i.e., LIFO) policy.

◇ Stack operations:

  ◇ INSERT - PUSH; DELETE - POP.

◇ The stack consists of elements $S[1..S.top]$, where $S[1]$ is the element at the bottom and $S[S.top]$ is the element at the top.

  ◇ $S.top = 0$ - the stack contains no element, i.e., it is empty.

  ◇ Stack underflows - when attempt to pop an empty stack.

  ◇ Stack overflows - S.top exceeds n.

STACK_EMPTY(S)

if $S.top == 0$
  return $TRUE$
else
  return $FALSE$

PUSH(S, x)

$S.top = S.top + 1$
$S[S.top] = x$

POP(S)

if $STACK\_EMPTY(S)$
  error $UNDERFLOW$
else
  $x = S.top$
  $S.top = S, top - 1$
  return x

◇ Analysis: running time of PUSH and POP are $O(1)$.

a pystack

In [1]:

```
# @author: meng (gmail: y(dot)meng201011)
class Stack:
    """A sample implement of stack with list"""
    def __init__(self):
        self.items = []

    def pop(self):
        if (self.is_empty()):
            return 'UNDERFLOW'
        else:
            return self.items.pop()

    def push(self, item):
        self.items.append(item)

    def is_empty(self):
        return (self.items == [])
```

In [2]:

```
stack = Stack()
stack.push('stack')
stack.push('sample')
stack.push('a')
stack.push(1)
stack.push(2)
stack.push(3)
while not stack.is_empty():
    print stack.pop()

print stack.pop()
```

```
3
2
1
a
sample
stack
UNDERFLOW
```

★ Queues

- Queue implements a first-in, first out (i.e., FIFO) policy.
- INSERT - ENQUEUE; DELETE - DEQUEUE
- Queue has a head and a tail. Every element will go in from tail and go out from head.
- When $Q.head = Q.tail$, the queue is empty.

ENQUEUE(Q, x)

$Q[Q.tail] = x$
$if\ Q.tail == Q.length$
$\quad Q.tail = 1$
$else$
$\quad Q.tail = Q.tail + 1$

DEQUEUE(Q)

$x = Q[Q.head]$
$if\ Q.head == Q.length$
$\quad Q.head = 1$
$else$
$\quad Q.head = Q.head + 1$
$return\ x$

- Analysis: Each of ENQUEUE and DEQUEUE takes $O(1)$ time.

a pyqueue

In [3]:

```
# @author meng (gmail: y(dot)meng201011)
class PyQueue:
    """Implement queue using list"""
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if (self.is_empty()):
            return "UNDERFLOW"
        else:
            item = self.items[0]
            self.items = self.items[1:]
            return item

    def is_empty(self):
        return (self.items == [])
```

In [4]:

```
queue = PyQueue()
queue.enqueue("a")
queue.enqueue("sample")
queue.enqueue("queue")
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
while not (queue.is_empty()):
    print queue.dequeue()

print queue.dequeue()
```

```
a
sample
queue
1
2
3
UNDERFLOW
```

In [ ]:

```

```