M2 ch14: Queues and Stacks

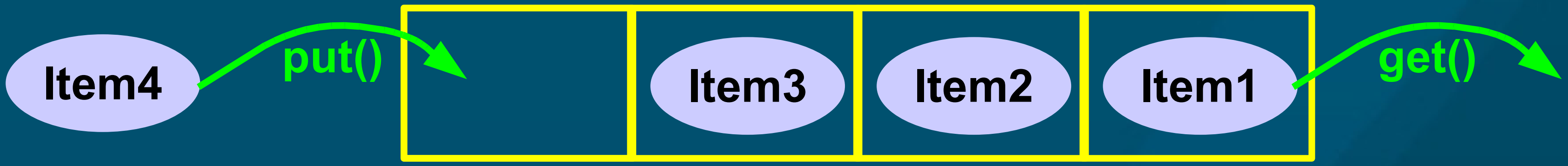
30 Nov 2007
CMPT14x
Dr. Sean Ho
Trinity Western University

Review of last time: §14.7-14.8

- Trees:
  - Definition of terms:
    - Parent, children, root, leaves, degree, depth, level, forest
  - Depth-first vs. breadth-first search
  - Binary trees: pre/in/post-order traversal
  - Binary search trees (BST):
    - Type definition
    - Search, Insert, Delete
    - Algorithmic efficiency of BST Search

Queues

- A queue is a list-like data structure where items added first to the queue are withdrawn first



- First-in / first-out: FIFO
- e.g., waiting in line for a bank teller
- Operations:
  - put(): add an item to the end of the queue
  - get(): withdraw item at the head of the queue
  - empty(), full(), size(): check number of items

Implementing queues

- Use a subclass of linked-lists (inheritance)

```
class Queue(LinkedList):
```

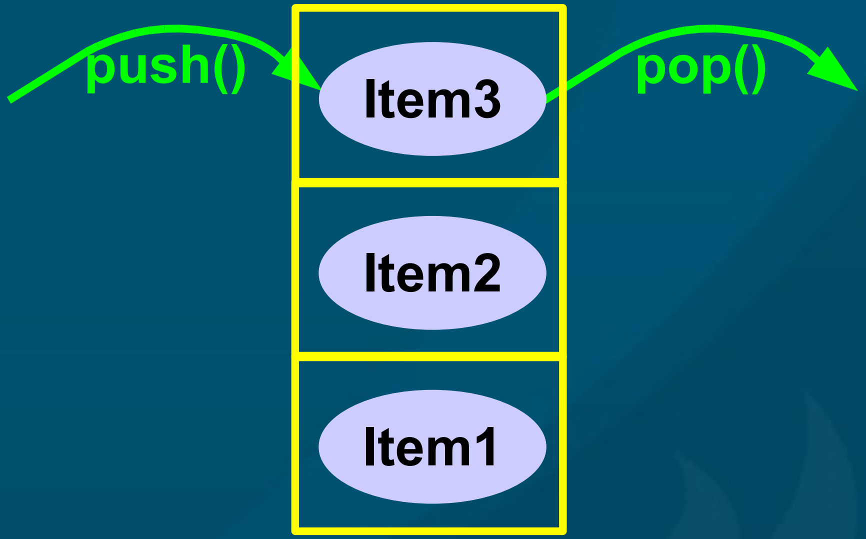
- Implement put()/get() using linked-list operations:

```
def put(self, data):
    self.insert(self.size, data)        # insert at tail
def get(self):
    data = self.head.data               # save the payload
    self.delete(0)                      # delete first node
    return data
```

- M2 book gives a different implementation using dynamic arrays

Stacks

- A stack is like a queue, but items added last to the stack are withdrawn first



- Last-in / first-out: LIFO
- e.g., RPN calculator
- Operations:
  - push(): add an item to the top of the stack
  - pop(): withdraw item from the top of the stack
  - empty(), full(), size(): check number of items

Implementing stacks

- Could use either linked-lists or arrays

```
class Stack:
    def __init__( self, maxsize=1 ):
        self.stack = range( maxsize )     # allocate new array
        self.top = -1                     # index of top of stack
```

- push()/pop() from the array:

```
    def push( self, data ):               # what if array is full?
        self.top += 1
        self.stack[ self.top ] = data     # push onto top
    def pop( self ):
        self.top -= 1
        return self.stack[ top+1 ]
```

Using Python lists for queues/stacks

- Most languages will only have arrays and pointers
  - Use pointers to build a linked-list ADT
  - Use either arrays or linked-lists to make queue or stack ADT
- Python lists are special
  - Provide many of the advantages of linked-lists
  - Can use Python lists naturally as queues/stacks
  - Stack: .append(), .pop() (pops from tail)
  - Queue: .append(), .pop(0) (pops from head)
    - See Py tut 5.1

TODO

- Paper due next Mon 3Dec
- Lab10 due next Wed 5Dec:
  - Implement one of your old Lab04-07 in M2
  - Full lab-writeup (may reuse parts of old writeup)
- Final exam next Sat 8Dec: 9-11am Neu37