i206: Lecture 10:
Stacks, Queues

Tapan Parikh

Spring 2013

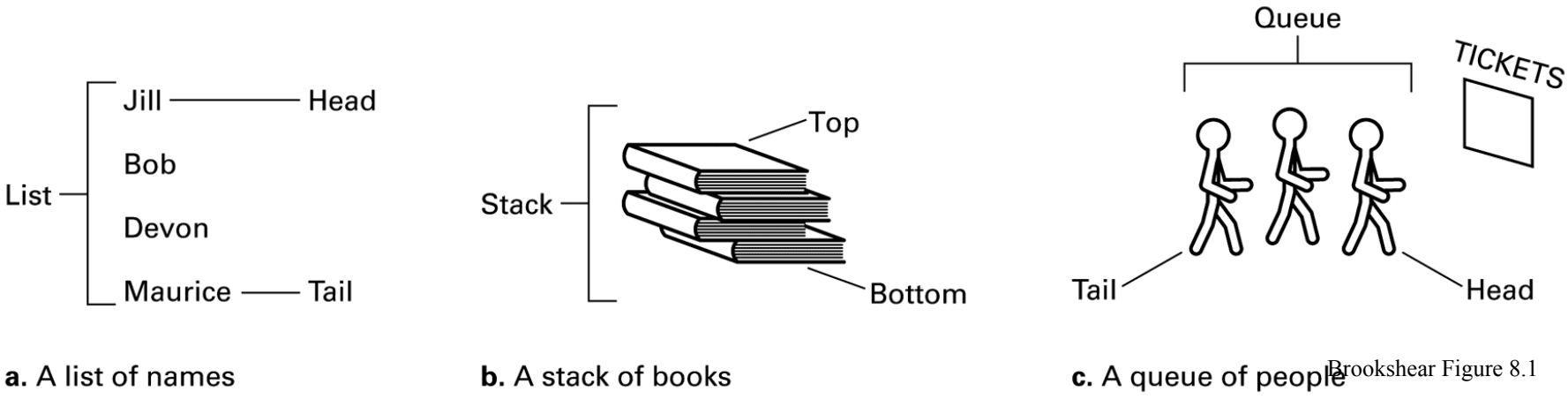
Some slides courtesy Marti Hearst, John Chuang and others

Outline

- What is a data structure
- Basic building blocks: arrays and linked lists
- Data structures (uses, methods, performance):
  - List, stack, queue
  - Dictionary
  - Tree
  - Graph

List

- An ordered collection of objects



a. A list of names **b.** A stack of books **c.** A queue of people

Brookshear Figure 8.1

- Stacks and queues are special cases of lists

Stacks

Stack

- Container of objects that are inserted and removed according to the principle of
  - Last-in-first-out
  - LIFO
- Objects can be inserted at any time, but only the most recently inserted can be removed at any time.
- Operations:
  - Pop: remove item from stack
  - Push: enter item onto stack

Why Stacks?

- The Undo facility in software applications
  - Is LIFO a good model for this? Why or why not?
- The Back button facility in web browsers
  - Is LIFO a good model? Why or why not
- Certain mathematical calculators (operand stack)
  - Makes it easy to do parenthesized expressions
  - Example:
    - 10 Enter (pushes 10)
    - 30 Enter (pushes 30)
    - 15 Enter (pushes 15)
    - Plus (pops and adds the most recent pair; then pushes the result onto the top of the stack)
    - Plus (same as above; end up with 55 as only entry)

Stack Methods

- **push (o)** – Insert an item into/onto the stack
  - Input: an object. Output: none.
  - If the stack has a fixed size and the stack cannot accept the push, an stack-overflow exception/error is thrown (or returned)
- **pop()** – Returns the most recently inserted object from the stack and removes the object from the stack (an object is removed in last-in-first-out order)
  - Input: none. Output: an object.
  - If the stack is empty, a stack-empty exception/error is thrown (or returned)

Stack Methods

- Auxiliary/Support Methods
  - **size()** – Returns the number of objects in the stack
    - Input: none. Output: non-negative integer.
  - **isEmpty()** (or empty()) – Returns true if there are no objects in the stack
    - Input: none. Output: true or false
  - **peek()** (or top()) – Returns a reference to (alternatively, a copy of) the most recent item put into the stack
    - Input: none. Output: reference to an object (or an object if a copy)

Stack Running Times

- What is the running time of each operation?
- Push

  O(1)
- Pop

  O(1)
- isEmpty()

  O(1)

Stack Implementation

- In Python, the list can be used as a stack:
  - `list.append(x)`
  - `list.pop()`

- Let's try to implement our own stack as an exercise in understanding what it takes to implement a data structure

Stack Implementation

- First review a bit about python lists.
- What do these mean if we have an array `a = [1,5,7]`:

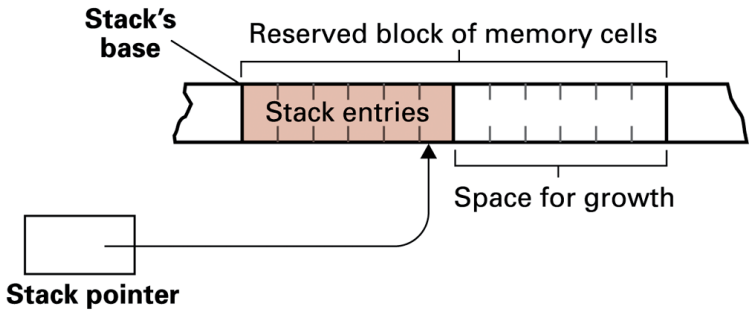
```
len(a)
a[:]
a[2:]
a[:2]
a[0:]
a[-1]
a[-1:]
a[:-1]
a[len(a):]
```

Python Activity

- Create a new Python class *Stack* that serves as a wrapper around a List object, and implements the following functions:
  - Push
  - Pop
  - Peek

Stack Implementation in Python

```
class Stack(list):
    def push(self, item):
        self[len(self):] = [item]        # add item to end of list

    def pop(self):
        if not self: return None         # stack is empty
        else:
            top = self[-1]               # get last element of list
            del self[-1]                 # delete last element
            return top

    def top(self):
        if not self: return None
        else: return self[-1]

    def isEmpty(self):
        return not self
```



Brookshear Figure 8.10

More list fun

Same effect:

```
a[len(a):] = [item]
a.append(item)
```

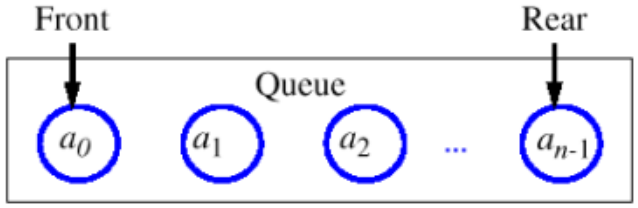
Different ways to build the same list of numbers:

```
s = []
for i in range(0,9):
    s.append(i)

s = Stack()
for i in range(0,9):
    s.push(i)

s = [i for i in range(0,9)]
```

Queues

http://www.lib.uconn.edu/DoddCenter/ASC/Wilcox/Students.htm

Queue



- Container of objects that are inserted and removed according to the principle of
  - First-in-first-out
  - FIFO
- Objects can be inserted at any time, but only the least recently inserted can be removed at any time.
- Operations:
  - Enqueue: put item onto queue
  - Dequeue: remove item from queue

Queue Running Times

- What is the running time of each operation?
- Enqueue

  O(1)
- Dequeue

  O(1)
- isEmpty()

  O(1)

Queue Methods

- enqueue(item) – Insert the item into the queue.
  - If the queue has a fixed capacity, an exception/error will be returned if attempting to enqueue an item into a filled queue.
  - Input: item. Output: none.
- dequeue() – Returns a reference to and removes the item that was least recently put into the queue (first-in-first-out)
  - If the queue is empty, an exception/error will be returned if attempting to dequeue.
  - Input: none. Output: item

Queue Methods

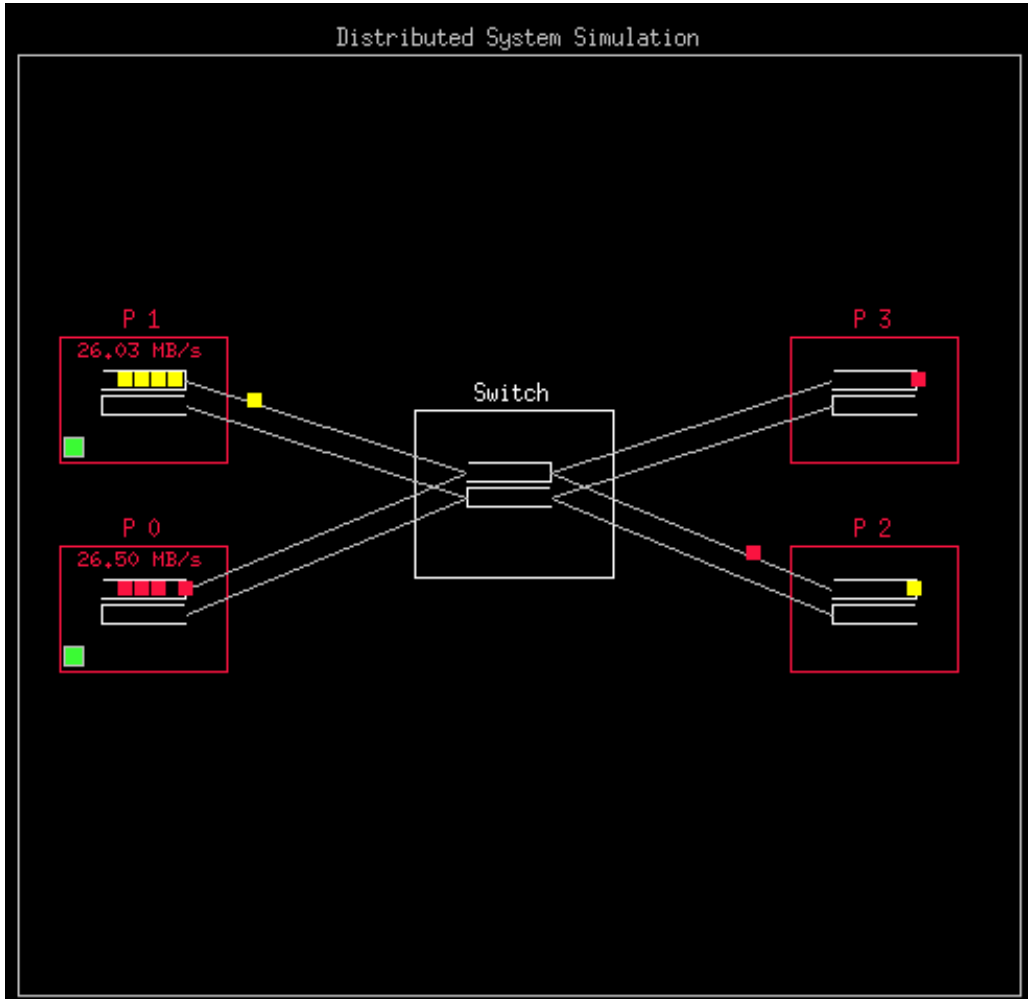
- size() (or getSize()) – Returns the number of items in the queue.
  - Input: none. Output: integer.
- isEmpty() – Checks if the queue has no items in it. Returns true if the queue is empty.
  - Input: none. Output: true or false.
- front() – Returns a reference to the "first" item in the queue (the least recent item). If the queue is empty, an exception/error will be returned if attempting to dequeue.
  - Input: none. Output: item.

How are Queues Used?

- Queues are used extensively in
  - The OS
    - For scheduling processes to receive resources
  - Computer networking
    - For storing and sending network packets

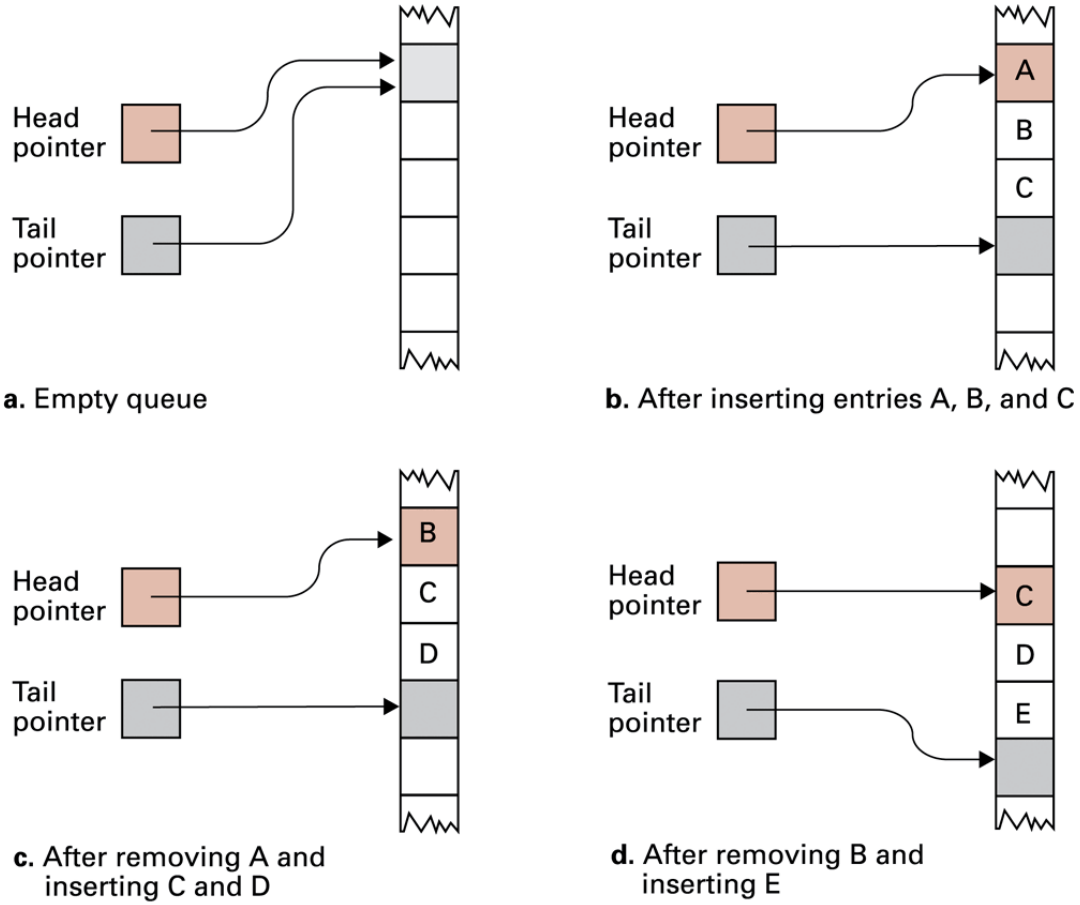
Use of Queues in Distributed Systems

Figure by
Remzi Arpaci-Dusseau



Queue Implementation

- In Python, the list can be used as a queue:
  - `list.append(x)`
  - `list.pop(0)`



a. Empty queue

b. After inserting entries A, B, and C

c. After removing A and inserting C and D

d. After removing B and inserting E

Brookshear Figure 8.13

Python Activity

- Use your Stack class to read in a string, and print out the letters in reverse, with a semicolon in between each letter