Stacks and Queues

25 Nov 2009
CMPT140
Dr. Sean Ho
Trinity Western University

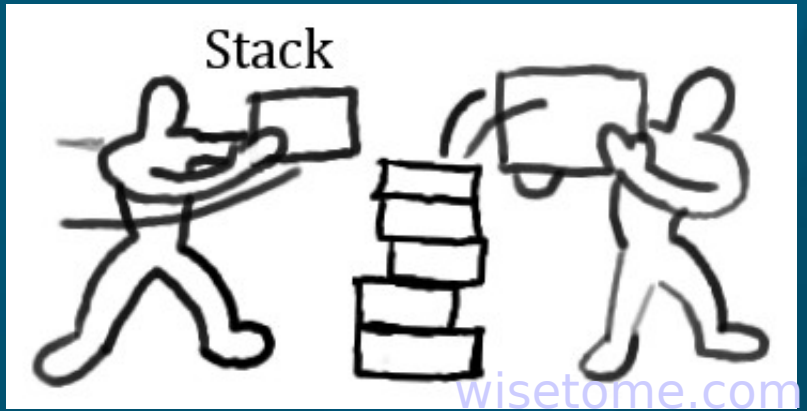
- ***Midterms back: avg 51/70 ≈ 74%***

TRINITY WESTERN UNIVERSITY

Abstract Data Structures

- An abstract data structure provides:
  - A way of storing data
  - Functions to access/operate on that data
- And ADT can be implemented using a class:
  - Attributes and methods
- Actually, dictionaries, lists, etc. are ADTs
  - Python has built-in implementations
- Today we'll talk about two more: stacks, queues
  - More in CMPT 231 (Data Structures) in Spr!

Stacks: theory



- In a stack, the last item added to the stack is the first item off:
  - LIFO: last in, first out
  - Analogy: stack of boxes or papers on desk
- Operations/methods:
  - push(x): add item x to top of stack
  - pop(): remove top item from stack and return it
  - (also: peek(): get top item without removing it from the stack)

Queues: theory

- With a queue, the first item added to the queue
  is the first item out of the queue:
  - FIFO: first in, first out
  - Analogy: waiting in line at bank, or tubes/pipes for water
- Operations/methods:
  - enqueue(x): add x to back of queue
  - dequeue(): remove item from front of queue and return it

Class design: Stack

- Design an interface for a Stack class:
  - Public methods: name, parameters, pre/post-conditions
  - Worry about attributes later

```
class Stack:
    def push( self, item ):
        """pre: check if stack full?
        post: stack has grown by 1; item is on top """
    def pop( self ):
        """pre: stack not empty
        post: return top item from stack;
        stack is smaller by one"""
```

Class design: Queue

- Design an interface for a Queue class:
  - Public methods: name, parameters, pre/post-conditions

```
class Queue:
    def enqueue( self, item ):
        """pre: queue not full
        post: queue has grown by 1; item is at tail"""
    def dequeue( self ):
        """pre: queue not empty
        post: return item from head of queue,
        queue gets smaller"""
```

Implementing stacks

- What attributes for our Stack class?
- The ADT can be implemented using various different existing data structures:
  - Plain C array; Python list; linked-list, etc.
- Using plain C array:
  - Fixed length → upper limit on size of stack
  - Keep an index to track current top of stack
  - push(x) stores item x in next entry of array
  - pop() returns top entry and decrements idx

Implement Stack class

- **class Stack:**

- Constructor sets up our empty list and index:
  - **def __init__(self, size=10):**
    - **self.__list = range(size) # set size**
    - **self.__top = -1**
- Use double-underscore ('__list') to hide private internal attributes
- We have Python lists here, but if we were using plain C arrays, we'd need to declare the maximum size of the array and element type:
  - **int* __list = new int[size]**

Implement push() and pop()

- push() takes an item to add to the stack:
  - **def push(self, item):**
    - Store item in array, and increment top idx:
      - **self.__top += 1**
      - **self.__list[self.__top] = item**
    - What if array is already full?
- pop() takes no arguments but returns an item:
  - **def pop(self):**
    - **self.__top -= 1**
    - **return self.__list[self.__top+1]**

Stacks and queues in Python

- Queues can also be implemented with C arrays.
- It's much easier to implement stacks/queues using Python lists: (but not all langs have this!)
- Stack:
  - Push: use .append(): myList.append(item)
  - Pop: use .pop(): myList.pop() (pops frm end)
- Queue:
  - Enqueue: use .append() (adds to end)
  - Dequeue: use .popleft() (pops from start)