Double-Ended Queues

CSE 2011

Fall 2009

Double-Ended Queue ADT

- Deque (pronounced “deck”)
- Allows insertion and deletion at both the front and the rear of the queue
- Deque ADT: operations

addFirst(*e*): insert *e* at the beginning of the deque

addLast(e): insert *e* at the end of the deque

removeFirst(): remove and return the first element

removeLast(): remove and return the last element

getFirst(): return the first element

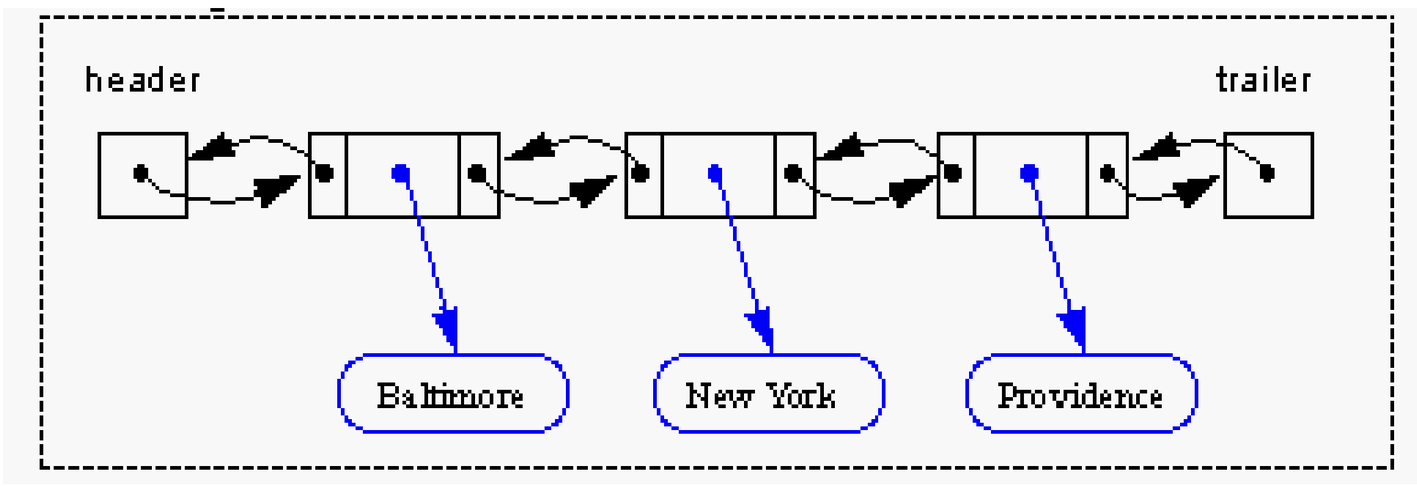
getLast(): return the last element

isEmpty(): return true if deque is empty; false otherwise

size(): return the number of objects in the deque

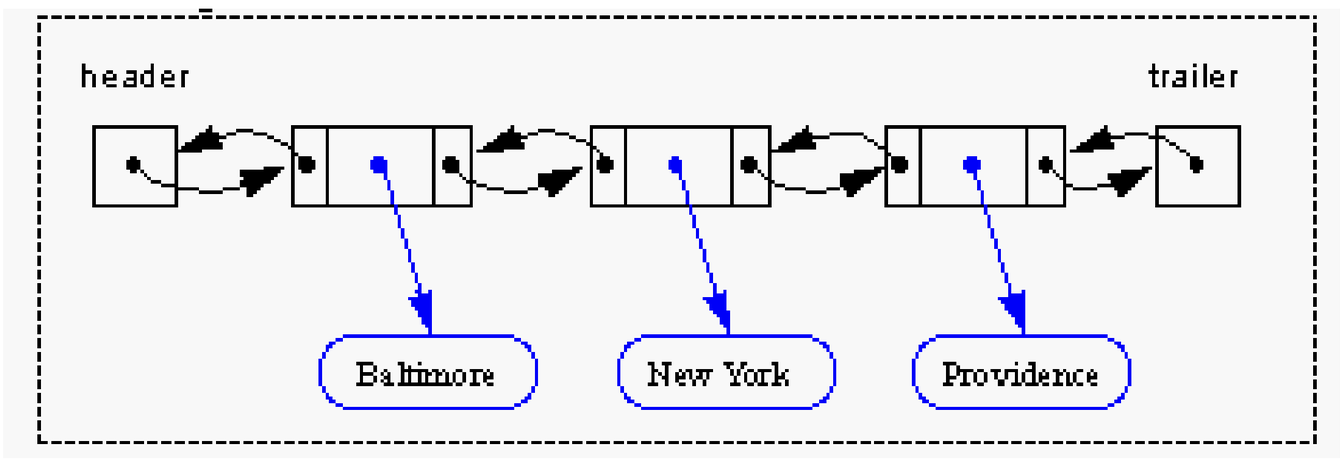
Implementation Choices

- Array (homework)
- Singly linked list: removing at the tail costs $\theta(n)$
- Doubly linked list

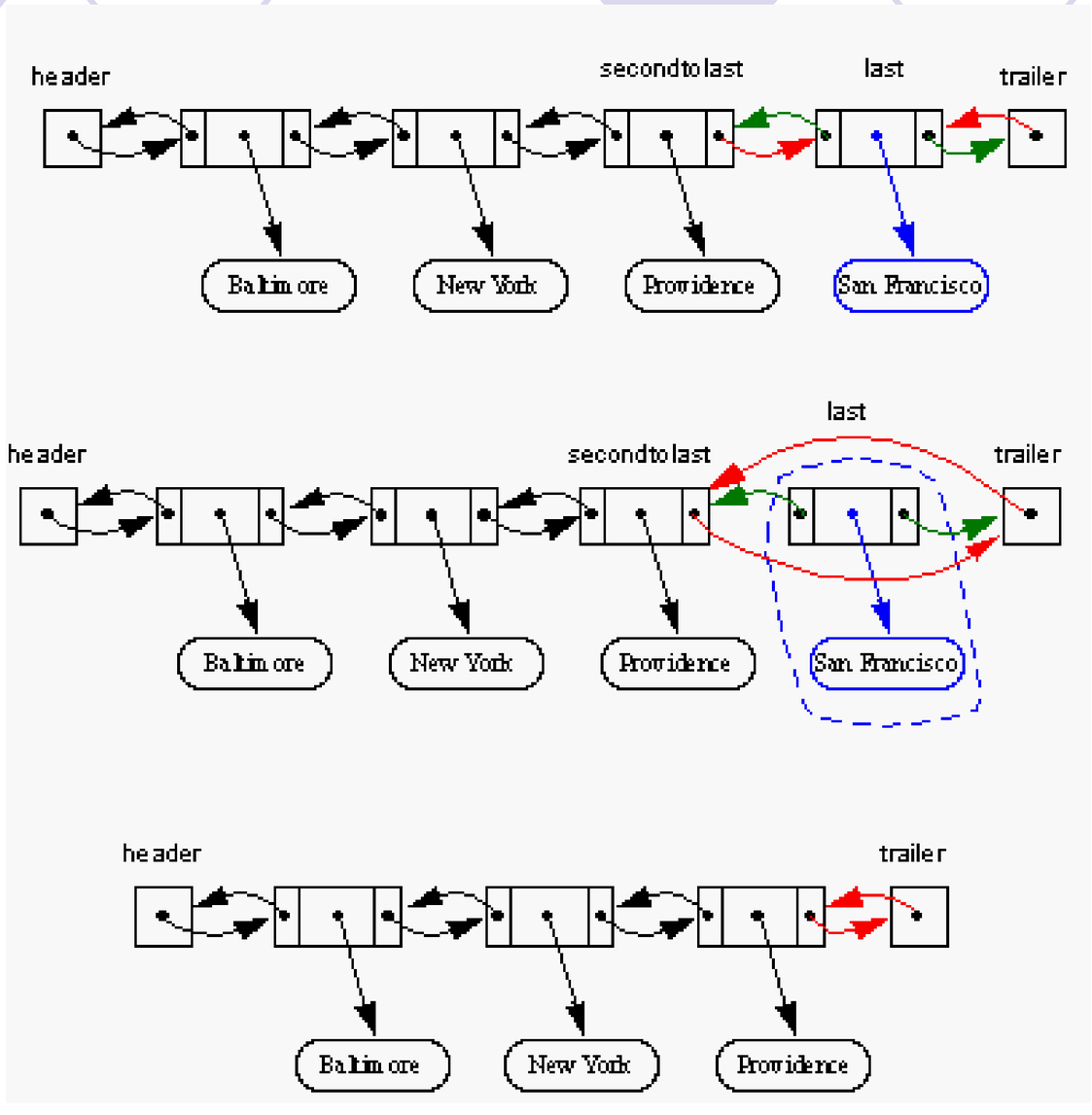


Implementing a Deque with a Doubly Linked List

- Recall the special cases of *dequeue*() when Q becomes empty, and *enqueue*() when Q is empty
- To avoid these special cases, use 2 dummy nodes:
  - header sentinel (*prev* pointer is null)
  - trailer sentinel (*next* pointer is null)



removeLast() and *addLast*()



Implementing Stacks and Queues with Deques

Stack Method	Deque Implementation
size()	size()
isEmpty()	isEmpty()
top()	last()
push(e)	insertLast(e)
pop()	removeLast()

Queue Method	Deque Implementation
size()	size()
isEmpty()	isEmpty()
front()	first()
enqueue()	insertLast(e)
dequeue()	removeFirst()

The Adapter Pattern

- Using methods of one class to implement methods of another class
- Example: using Deque to implement Stack and Queue

Next time ...

- Quick sort: analysis
- Linear-time sorting
- When to use which sorting algorithm?
- Extendable arrays

- Reminder: Labtest Oct. 8, 17:30-19:00