Queues

Christophe Rhodes

Motivation

- first-in first-out collection
- useful for mediating access to resources
  - wait your turn
  - (but see priority-queues)

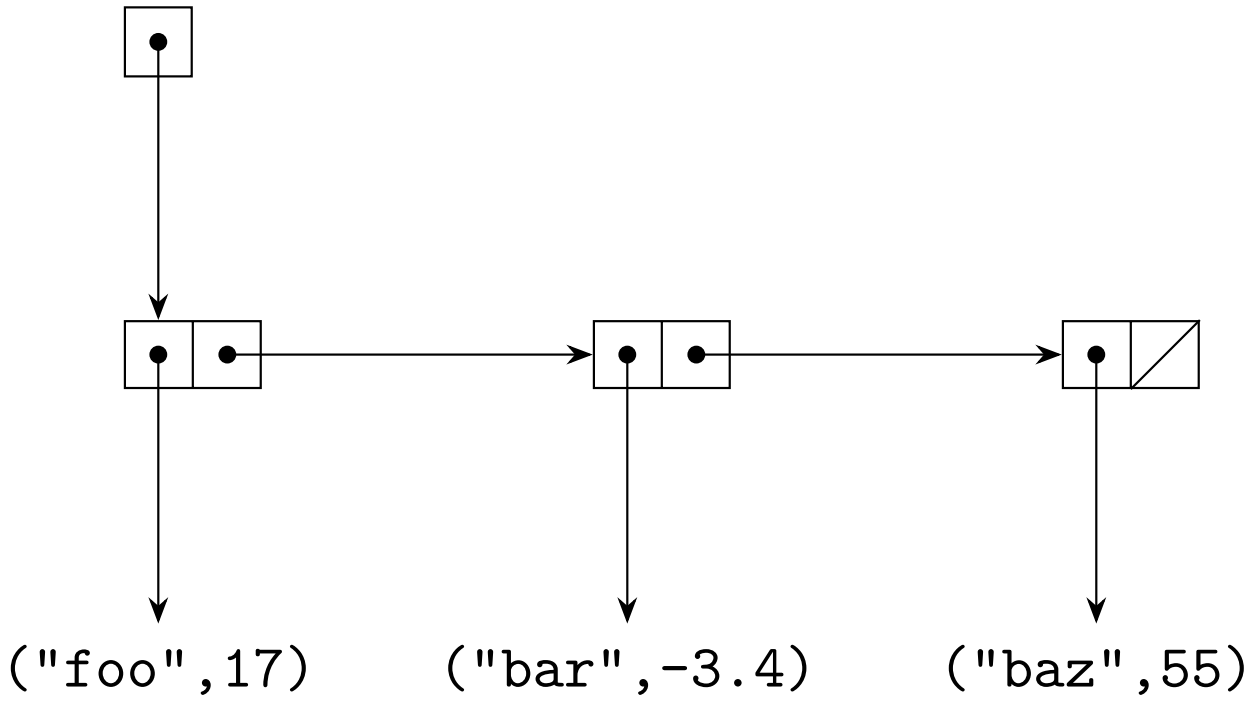
Definition

A queue is an extensible collection of data where removal of data happens at the head of the queue (maintaining the order of remaining elements), and addition of data happens at the tail of the queue (again maintaining order of other elements)

Operations

head	return the element at the head of the queue
dequeue!	return and remove the element at the head of the queue
enqueue![o]	add o to the tail of the queue
empty?	return *true* if the queue has no elements.

Implementation



Complexity analysis

head, dequeue!, empty?

Exactly the same as stack operations top, pop!, empty?

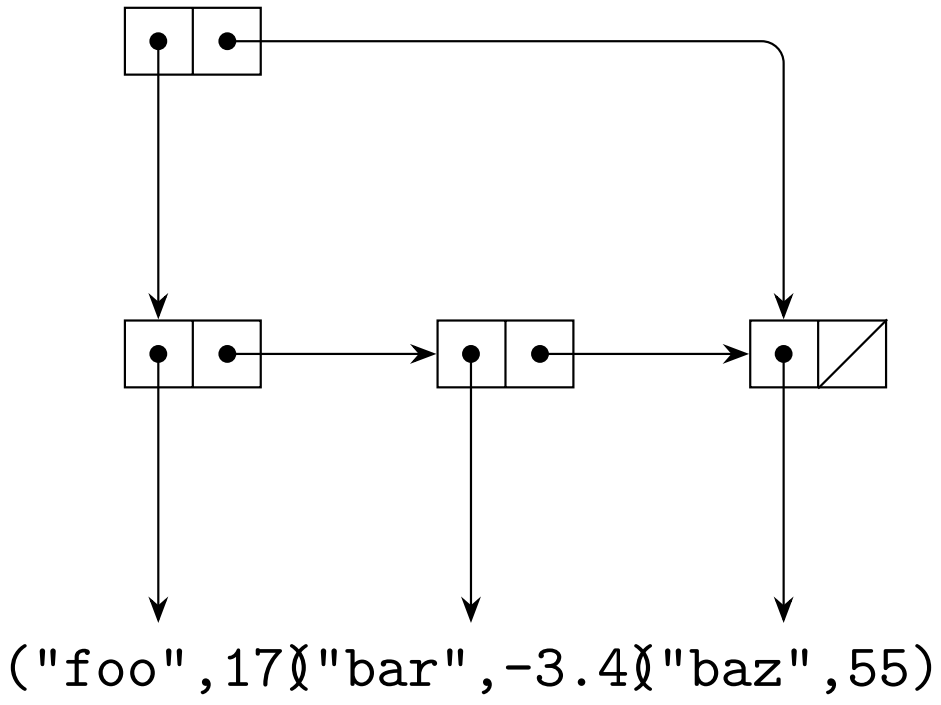
$$\Longrightarrow \Theta(1)$$

enqueue!

Add to the end of the underlying list: N pointer reads

$$\Longrightarrow \Theta(N)$$

Better implementation



Complexity analysis

head, dequeue!, empty?

As before, exactly the same as stack operations top, pop!, empty?

$$\Longrightarrow \Theta(1)$$

enqueue!

One pair allocation, two pointer writes

$$\Longrightarrow \Theta(1)$$

Work

1. implement queues using both variants described above (one with a tail pointer and one without). Count the operations it takes to create queues by successive enqueueing with 10, 100, 1000, 10000 and 100000 elements. Does this support the lecture discussion about complexity analysis?
2. do the stacks and queues quiz on the VLE
   - open from 22nd October 2018
   - closes 2nd November 2018