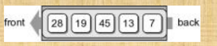
CSCI 204: Data Structures & Algorithms

Queue ADT

Revised based on textbook author’s notes.

Queue

- A restricted access container that stores a linear collection.
  - Very common for solving problems in computer science that require data to be processed in the order in which it was received.
  - Provides a **first-in first-out** (FIFO) protocol.
- New items are added at the **back** while existing items are removed from the **front** of the queue.



2

The Queue ADT

- A *queue* stores a linear collection of items with access limited to a first-in first-out order.
  - New items are added to the back.
  - Existing items are removed from the front.

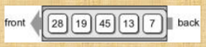
- Queue()
- is_empty()
- len()
- enqueue(item)
- dequeue()

3

Queue Example

- The following code creates the queue from the earlier slide.

```
Q = Queue()
Q.enqueue( 28 )
Q.enqueue( 19 )
Q.enqueue( 45 )
Q.enqueue( 13 )
Q.enqueue( 7 )
```

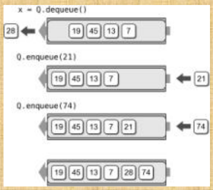


4

Queue Example

- We can remove items from the queue and add more items.

```
x = Q.dequeue()
Q.enqueue( 21 )
Q.enqueue( 74 )
```



5

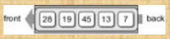
Queue Implementation

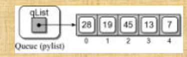
- Several common ways to implement a queue:
  - Python list
    - easiest to implement
  - Linked list
    - reduces memory wastes by eliminating the extra capacity created with an array.
  - Circular array
    - fast operations with a fixed size queue.

6

Queue: Python List

- How is the data organized within the Python list?
  - Add new items to the end of the list.
  - Remove items from the front of the list.





7

Queue: Python List

pylistqueue.py

```
# Implementation of the Queue ADT using a Python list.
class Queue :
  def __init__( self ):
    self._qlist = list()

  def is_empty( self ):
    return len( self ) == 0

  def __len__( self ):
    return len( self._qlist )

  def enqueue( self, item ):
    self._qlist.append( item )

  def dequeue( self ):
    assert not self.is_empty(), "Cannot dequeue from an empty queue."
    return self._qlist.pop( 0 )
```

8

Queue Analysis: Python List

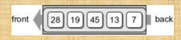
Queue Operation	Worst Case
q = Queue()	O(1)
len(q)	O(1)
q.is_empty()	O(1)
q.enqueue(x)	O(n)*
x = q.dequeue()	O(n)

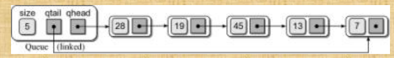
* While the **enqueue()** operation itself is O(1), the queue potentially needs to be expanded, which is O(n).

9

Queue: Linked List

- How should the data be organized?
  - Use both head and tail references.
  - Let the head of the list represent the front of the queue and the tail the back.





10

Your in-class work

- Implement a linked list queue
- Test your implementation with test_linkedlist_queue.py