Queues

What is a Queue?

A queue is a linear data structure that allows two primary operations:

- Enqueue: Adds an element to the back (or rear) of the queue.
- Dequeue: Removes and returns the element from the front (or head) of the queue.

Insertions and deletions follow the first-in first-out scheme

The Queue ADT

Auxiliary queue operations:

- object first(): returns the element at the front without removing it
- integer len(): returns the number of elements stored
- boolean is_empty(): indicates whether no elements are stored

Few important characteristics:

- FIFO Ordering: Elements are processed in the same order they were added.
- Dynamic size: The size of the queue can vary as elements are added and removed.
- Restricted access: Only the front element can be directly accessed without iterating through the entire queue.

Example

Operation	Return Value	first $\leftarrow Q \leftarrow$ last
Q.enqueue(5)	–	[5]
Q.enqueue(3)	–	[5, 3]
len(Q)	2	[5, 3]
Q.dequeue( )	5	[3]
Q.is_empty( )	False	[3]
Q.dequeue( )	3	[ ]
Q.is_empty( )	True	[ ]
Q.dequeue( )	"error"	[ ]
Q.enqueue(7)	–	[7]
Q.enqueue(9)	–	[7, 9]
Q.first( )	7	[7, 9]
Q.enqueue(4)	–	[7, 9, 4]
len(Q)	3	[7, 9, 4]
Q.dequeue( )	7	[9, 4]

Applications of Queues

- In operating systems, it is used for Process scheduling, Disk IO operations, Buffering, Access to shared resources (e.g., printer)
- In networking used for routing, data transmission etc
- Web servers, audio & video playback
- Event processing
- Simulation and Modelling

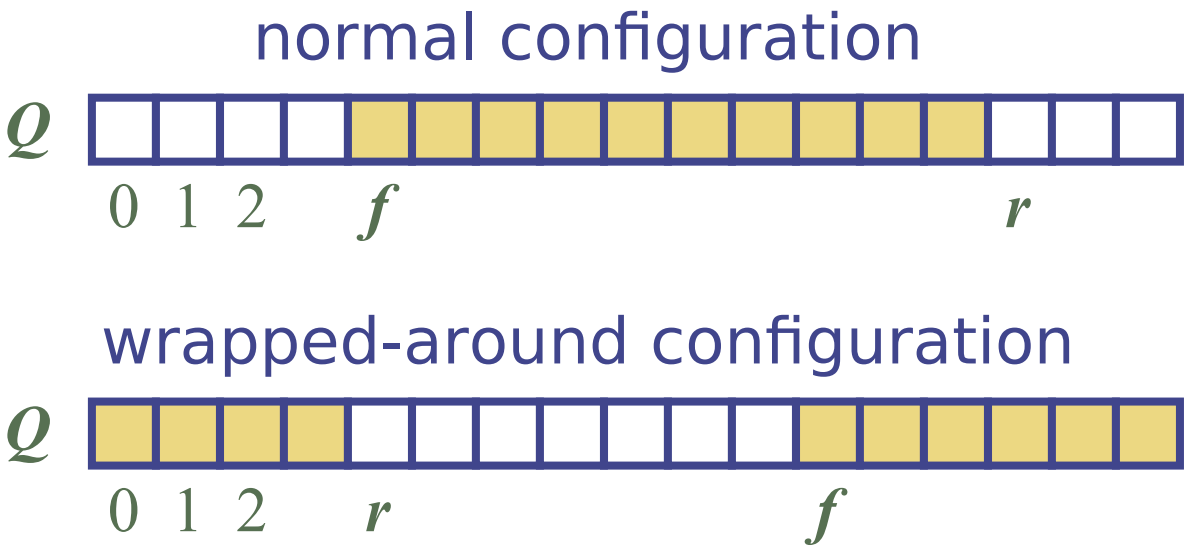
Fixed Array-based Queue

Use an array of **fixed size** N in a circular fashion

Two variables keep track of the front and rear

- f index of the front element
- r index immediately past the rear element

Array location r is kept empty

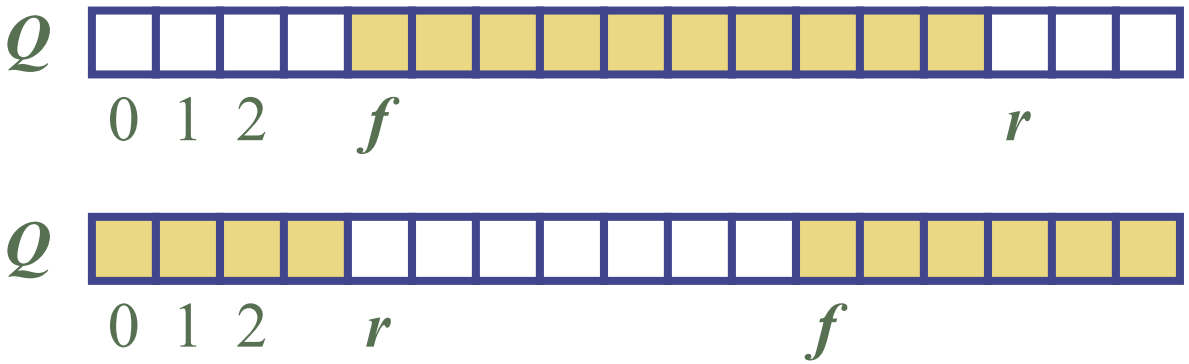


Queue Operations

We use the modulo operator (remainder of division)

Algorithm ***size***()
  return $(N - f + r) \bmod N$

Algorithm ***isEmpty***()
  return $(f = r)$

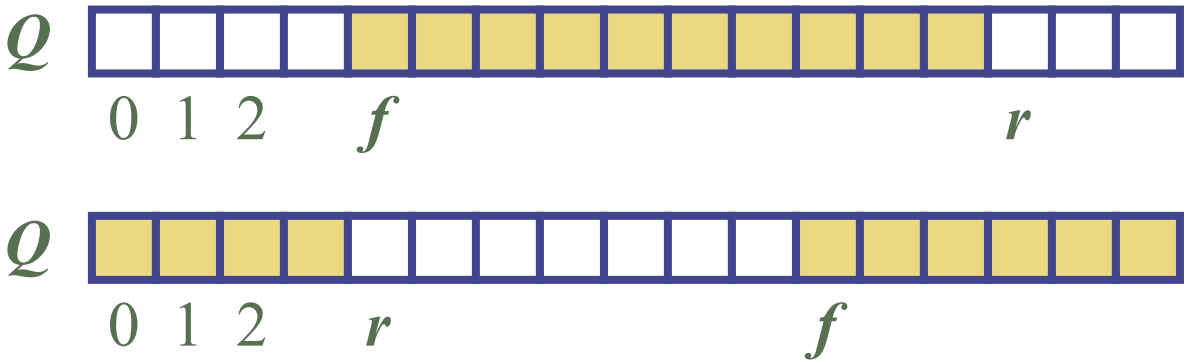


Queue Operations (cont.)

Operation enqueue throws an exception if the array is full

This exception is implementation-dependent

Algorithm *enqueue*(*o*)
  if *size*() = *N* - 1 **then**
    throw *FullQueueException*
  else
    Q[*r*] ← *o*
    r ← (*r* + 1) mod *N*

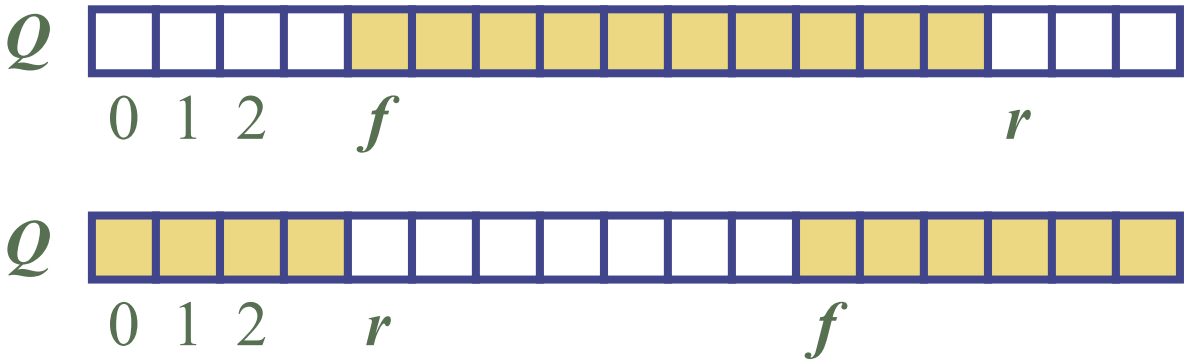


Queue Operations (cont.)

Operation dequeue throws an exception if the queue is empty

This exception is specified in the queue ADT

```
Algorithm dequeue()
  if isEmpty() then
    throw EmptyQueueException
  else
    o ← Q[f]
    f ← (f + 1) mod N
    return o
```



Queue in Python (Dynamic Arrays)

Use the following three instance variables:

_data: is a reference to a list instance with a fixed capacity.

_size: is an integer representing the current number of elements stored in the queue (as opposed to the length of the data list).

_front: is an integer that represents the index within data of the first element of the queue (assuming the queue is not empty).

Queue in Python, Beginning

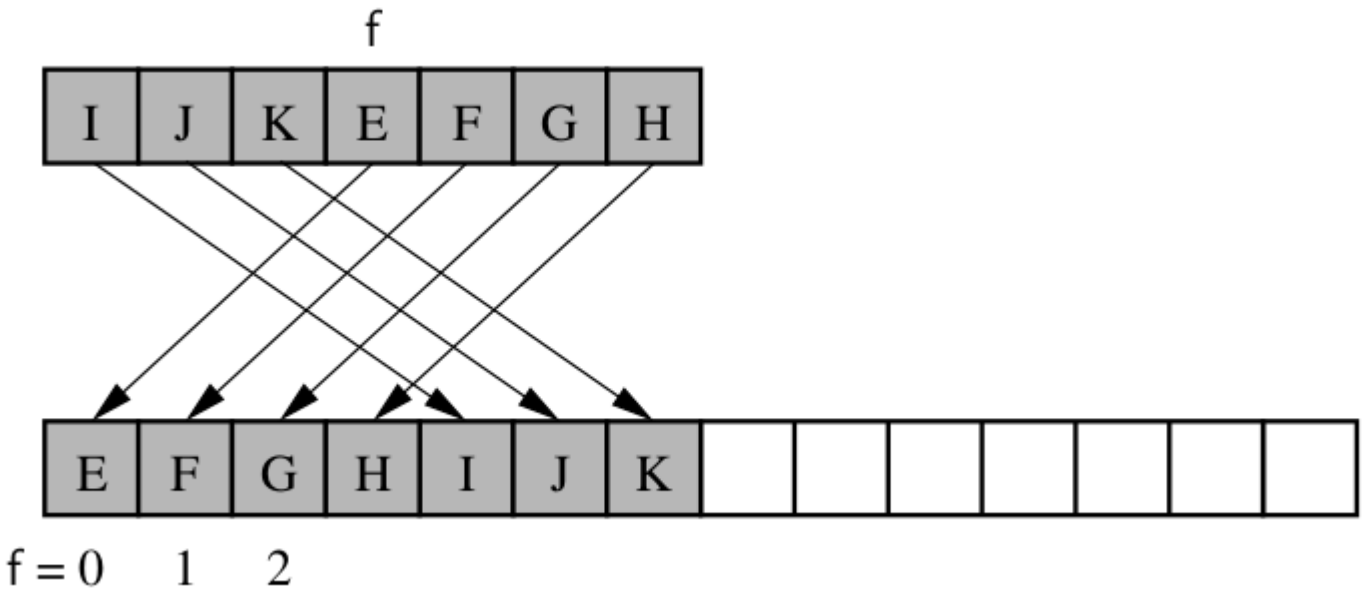
```
1   class ArrayQueue:
2     """FIFO queue implementation using a Python list as underlying storage."""
3     DEFAULT_CAPACITY = 10          # moderate capacity for all new queues
4
5     def __init__(self):
6       """Create an empty queue."""
7       self._data = [None] * ArrayQueue.DEFAULT_CAPACITY
8       self._size = 0
9       self._front = 0
10
11    def __len__(self):
12      """Return the number of elements in the queue."""
13      return self._size
14
15    def is_empty(self):
16      """Return True if the queue is empty."""
17      return self._size == 0
```

```
19    def first(self):
20      """Return (but do not remove) the element at the front of the queue.
21
22      Raise Empty exception if the queue is empty.
23      """
24      if self.is_empty():
25        raise Empty('Queue is empty')
26      return self._data[self._front]
27
28    def dequeue(self):
29      """Remove and return the first element of the queue (i.e., FIFO).
30
31      Raise Empty exception if the queue is empty.
32      """
33      if self.is_empty():
34        raise Empty('Queue is empty')
35      answer = self._data[self._front]
36      self._data[self._front] = None                    # help garbage collection
37      self._front = (self._front + 1) % len(self._data)
38      self._size -= 1
39      return answer
```

Queue in Python, Continued

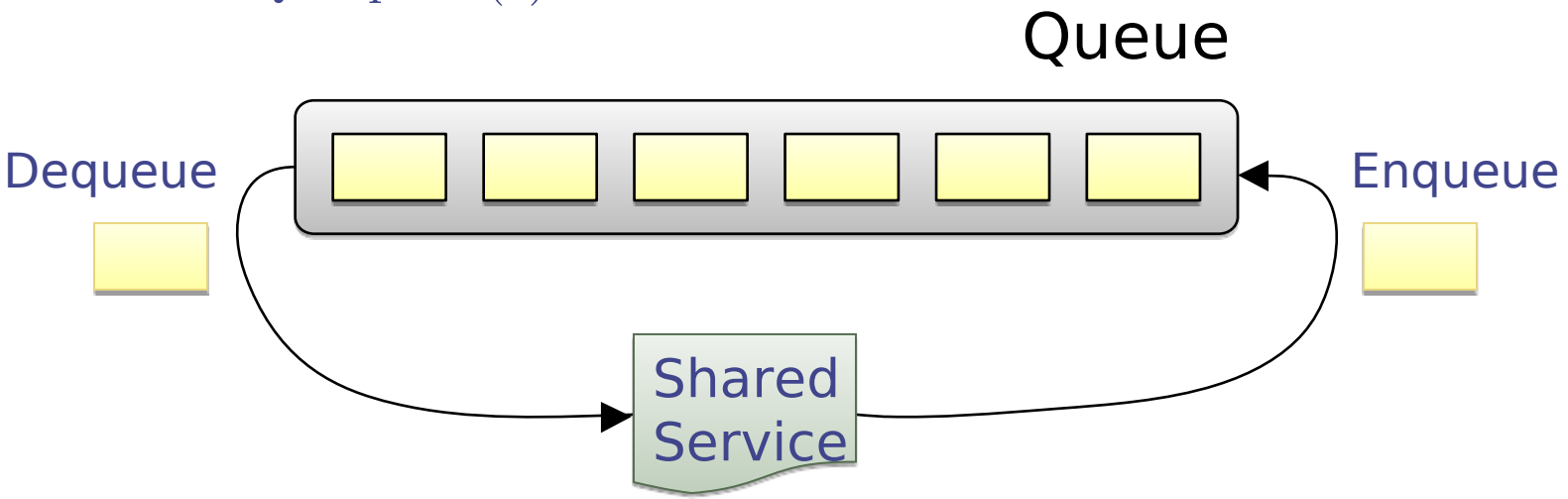
```
40    def enqueue(self, e):
41      """Add an element to the back of queue."""
42      if self._size == len(self._data):
43        self._resize(2 * len(self.data))          # double the array size
44      avail = (self._front + self._size) % len(self._data)
45      self._data[avail] = e
46      self._size += 1
47
48    def _resize(self, cap):                       # we assume cap >= len(self)
49      """Resize to a new list of capacity >= len(self)."""
50      old = self._data                            # keep track of existing list
51      self._data = [None] * cap                   # allocate list with new capacity
52      walk = self._front
53      for k in range(self._size):                 # only consider existing elements
54        self._data[k] = old[walk]                 # intentionally shift indices
55        walk = (1 + walk) % len(old)              # use old size as modulus
56      self._front = 0                             # front has been realigned
```

Resizing the queue, while realigning the front element with index 0



Application: Round Robin Schedulers

- We can implement a round robin scheduler using a queue Q by repeatedly performing the following steps:
  1. e = Q.dequeue()
  2. Service element e
  3. Q.enqueue(e)



Performance of an array-based implementation of a queue

Operation	Running Time
Q.enqueue(e)	$O(1)^*$
Q.dequeue( )	$O(1)^*$
Q.first( )	$O(1)$
Q.is_empty( )	$O(1)$
len(Q)	$O(1)$

*amortized

Questions

1)What values are returned during the following sequence of queue operations, if executed on an initially empty queue?

enqueue(5), enqueue(3), dequeue(), enqueue(2), enqueue(8), dequeue(), dequeue(), enqueue(9), enqueue(1), dequeue(), enqueue(7), enqueue(6), dequeue(), dequeue(), enqueue(4), dequeue(), dequeue().

Questions

2) Suppose an initially empty queue Q has executed a total of 32 enqueue operations, 10 first operations, and 15 dequeue operations, 5 of which raised Empty errors that were caught and ignored. What is the current size of Q?

3) Had the queue of the previous problem been an instance of ArrayQueue that used an initial array of capacity 30, and had its size never been greater than 30, what would be the final value of the front instance variable?

Questions

4) Consider what happens if the loop in the ArrayQueue. resize method at lines 53–55 of Code Fragment shown in a previous slide:

```
for k in range(self. size):
    self. data[k] = old[k]          # rather than old[walk]
```

Give a clear explanation of what could go wrong.

Double-Ended Queues (deque)

D.add_first(e):	Add element e to the front of deque D.
D.add_last(e):	Add element e to the back of deque D.
D.delete_first( ):	Remove and return the first element from deque D; an error occurs if the deque is empty.
D.delete_last( ):	Remove and return the last element from deque D; an error occurs if the deque is empty.

Additionally, the deque ADT will include the following accessors:

D.first( ):	Return (but do not remove) the first element of deque D; an error occurs if the deque is empty.
D.last( ):	Return (but do not remove) the last element of deque D; an error occurs if the deque is empty.
D.is_empty( ):	Return True if deque D does not contain any elements.
len(D):	Return the number of elements in deque D; in Python, we implement this with the special method __len__.

Operation	Return Value	Deque
D.add_last(5)	–	[5]
D.add_first(3)	–	[3, 5]
D.add_first(7)	–	[7, 3, 5]
D.first( )	7	[7, 3, 5]
D.delete_last( )	5	[7, 3]
len(D)	2	[7, 3]
D.delete_last( )	3	[7]
D.delete_last( )	7	[ ]
D.add_first(6)	–	[6]
D.last( )	6	[6]
D.add_first(8)	–	[8, 6]
D.is_empty( )	False	[8, 6]
D.last( )	6	[8, 6]

Comparison of our deque ADT and the Python's built-in collections.deque class

Our Deque ADT	collections.deque	Description
len(D)	len(D)	number of elements
D.add_first( )	D.appendleft( )	add to beginning
D.add_last( )	D.append( )	add to end
D.delete_first( )	D.popleft( )	remove from beginning
D.delete_last( )	D.pop( )	remove from end
D.first( )	D[0]	access first element
D.last( )	D[−1]	access last element
	D[j]	access arbitrary entry by index
	D[j] = val	modify arbitrary entry by index
	D.clear( )	clear all contents
	D.rotate(k)	circularly shift rightward k steps
	D.remove(e)	remove first matching element
	D.count(e)	count number of matches for e

Exercise

- What values are returned during the following sequence of deque ADT operations, on initially empty deque? add first(4), add last(8), add last(9), add first(5), back( ), delete first( ), delete last( ), add last(7), first( ), last( ), add last(6), delete first( ), delete first( ).

- R-6.13 Suppose you have a deque D containing the numbers (1, 2, 3, 4, 5, 6, 7, 8), in this order. Suppose further that you have an initially empty queue Q. Give a code fragment that uses only D and Q (and no other variables) and results in D storing the elements in the order (1, 2, 3, 5, 4, 6, 7, 8).

- Repeat the previous problem using the deque D and an initially empty stack S.

Exercise

- Write a method to reverse a queue
- Describe how to implement the queue ADT using two stacks.
- Describe how to implement the stack ADT using two queues
- Describe how to implement the double-ended queue ADT using two stacks as instance variables. What are the running times of the methods?

Exercise

- Suppose you have a stack S containing n elements and a queue Q that is initially empty. Describe how you can use Q to scan S to see if it contains a certain element x, with the additional constraint that your algorithm must return the elements back to S in their original order. You may only use S, Q, and a constant number of other variables.

References

Data Structures and Algorithms in Python
Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser

Introduction to Algorithms
Leiserson,Stein,Rivest,Cormen

Algorithms, 4th Edition
Robert Sedgewick and Kevin Wayne

Few Images from the internet