

CSCI 204: Data Structures & Algorithms

Priority Queue ADT

Revised based on textbook author's notes.

Priority Queues

- Some applications require the use of a queue in which items are assigned a priority.
 - higher priority items are dequeued first.
 - items with equal priority still follow FIFO.

Some Applications

- Operating systems such as Linux use priority queues to manage their jobs (try e.g., the **top** command)
- Simulations use priority queues to manage events to be simulated
- All other FIFO queues, e.g., online shopping queues are special cases of priority queue, that is, time of arrival is the priority

The Priority Queue ADT

- A *priority queue* is a queue in which each item is assigned a priority and items with a higher priority are removed before those with lower priority.
 - Integer values are used for the priorities.
 - Smaller integers have a higher priority. Other arrangements, such as larger values represent higher priority are possible.

The Operations

- PriorityQueue()
- is_empty()
- len()
- enqueue(item, priority)
- dequeue()
- peek()

Priority Queue Example

- Consider the following code segment:

```
Q = PriorityQueue( 6 )
Q.enqueue( "purple", 5 )
Q.enqueue( "black", 1 )
Q.enqueue( "orange", 3 )
Q.enqueue( "white", 0 )
Q.enqueue( "green", 1 )
Q.enqueue( "yellow", 5 )
```



7

Priority Queue Implementation

- How should the ADT be implemented. We must consider:
 - A priority must be associated with each item in the queue.
 - The next item to be dequeued is the item with the highest priority.
 - If multiple items have the same priority, those must be dequeued in a FIFO order.

8

Priority Queue Implementation

- There can be many different implementations, we'll consider three here
 - Textbook approach
 - Linked list
 - Bounded array with linked lists

1. Textbook approach

- The priority queue is implemented as a Python list
- The enqueue operation puts the item at the end of the queue (as in our FIFO queue)
- The dequeue operation takes the item with the highest priority off the queue (note: the item could be anywhere in the queue!)

10

Queue operations

```
def enqueue( self, item, priority ):
    """Adds the given item to the queue"""
    # Create a new instance of the Storage class and append it to the list.
    entry = _PriorityEntry( item, priority )
    self._qlist.append( entry )

def dequeue( self ):
    """Removes and returns the first item in the queue"""
    assert not self.is_empty(), "Cannot dequeue from an empty queue."
    # Find the entry with the highest priority
    top = self.find_top_priority()
    # Remove the entry with the highest priority and return the item.
    entry = self._qlist.pop( top )
    return entry

def peek( self ):
    """Return the value of the top priority without removing it"""
    assert not self.is_empty(), "Cannot peek from an empty queue."
    top = self.find_top_priority()
    return self._qlist[top].item

def find_top_priority( self ):
    """Find the entry with the highest priority"""
    highest = self._qlist[0].priority
    high_index = 0
    for i in range( len( self ) ):
        # See if the ith entry contains a higher priority (smaller integer).
        if self._qlist[i].priority < highest:
            highest = self._qlist[i].priority
            high_index = i
    return high_index
```

priorityq.py

Details of find_top_priority()

Basically it is the same process of finding a minimum in a list.

```
def find_top_priority( self ):
    highest_index = 0
    highest = self._qlist[highest_index].priority
    for i in range( len( self ) ):
        if highest > self._qlist[i].priority: # smaller value has higher priority
            highest_index = i
            highest = self._qlist[i].priority
    return highest_index
```

Try the program testpriorityqueue.py

Complexity of operations

What is the complexity for dequeue?

$O(n)$

What is the complexity for enqueue?

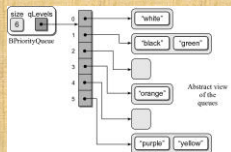
$O(1)$

2. Bounded Priority Queue

- A bounded priority queue has a fixed set of priorities
- We use an array to represent the set of priorities, each array element maintains a queue of the items with the same priority

Bounded Priority Queue

- The following example shows a bounded priority queue with six levels



Bounded Priority Q Implementation

```
from array204 import Array
from listqueue import Queue

class BPriorityQueue:
    def __init__(self, num_levels = 6):
        self._qsize = 0
        self._qllevels = Array(num_levels)
        for i in range(num_levels):
            self._qllevels[i] = Queue()

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

    def __len__(self):
        return len(self._qsize)
# ...
```

Bounded Priority Q Implementation

```
class BPriorityQueue:
# ...
    def enqueue(self, item, priority):
        assert priority >= 0 and priority < len(self._qllevels), \
            "Invalid priority level."
        self._qllevels[priority].enqueue(item)

    def dequeue(self):
        # Make sure the queue is not empty.
        assert not self.is_empty(), "Cannot dequeue from an empty queue."

        # Find the first non-empty queue.
        top_index = self.find_top_priority_queue()

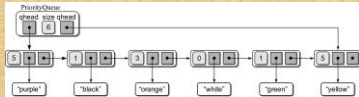
        # We know the queue is not empty, so dequeue from the ith queue.
        return self._qllevels[top_index].dequeue()
```

Bounded Priority Q Implementation

```
class BPriorityQueue:
# ...
    def find_top_priority_queue(self):
        # find the first non-empty queue,
        # a.k.a. highest priority
        i = 0
        p = len(self._qllevels)
        while i < p and self._qllevels[i].is_empty():
            i += 1
        return i
```

3. Unbounded Priority Q: Linked List

- We can use a singly linked list:
 - Head and tail references.
 - Append new entries to the end.



19

Priority Queue Analysis

- The worst case analysis for the two implementations.

Queue Operation	Python List	Linked List
<code>q = PriorityQueue()</code>	$O(1)$	$O(1)$
<code>len(q)</code>	$O(1)$	$O(1)$
<code>q.is_empty()</code>	$O(1)$	$O(1)$
<code>q.enqueue(x)</code>	$O(n)$	$O(1)$
<code>x = q.dequeue()</code>	$O(n)$	$O(n)$

20

3a. Unbounded Priority Q: Linked List

- We can use a singly linked list:
 - Head and tail references.
 - Insert (enqueue) new item at the correct place
 - Remove (dequeue) at the beginning of the queue.

21

Implement enqueue()

- Your task is to implement the enqueue() method for a linked list based queue as in 3a in which other necessary methods have been implemented