PSET5: Ordered Collections Priority Queues

Anusha Murali

March 17, 2023

Tasks to do

Part II: Implement Ordered Collections with Priority Queues

1. Complete ListQueue: elements are stored a list
2. Complete TreeQueue: elements are stored in a BST
3. Complete BinaryHeap: elements are stored in a balanced binary tree

ListQueue

Task: Use a a simple list to store the queue elements in sorted order.

Step 1 Load Files

Load Files

1. # #use "order.ml"
2. # #use "orderedcoll.ml"
3. # #use "prioqueue.ml"

Step 2: Complete the `ListQueue` functor

Complete empty

```
let empty : queue =
    failwith "ListQueue empty not implemented"
```

empty

```
let empty : queue = []
```

Complete is_empty

```
let is_empty (q : queue) : bool =
    failwith "ListQueue is_empty not implemented"
```

is_empty

```
let is_empty (q : queue) : bool = (q = empty)
```

Step 2: Complete the `ListQueue` functor

Complete add

$$\text{let add (e : elt) (q : queue) : queue} =$$
$$\text{failwith "ListQueue add not implemented"}$$

add

```
let rec add (e : elt) (q : queue) : queue =
      match q with
      | [] -> e :: []
      | hd :: tl -> (match Elt.compare e hd with
                     | Less -> e :: q
                     | Greater -> hd :: (add e tl)
                     | Equal -> hd :: (add e tl)
                     )
```

Step 2: Complete the `ListQueue` functor

Complete take

let take (q : queue) : elt * queue =
failwith "ListQueue take not implemented"

take

```
let take (q : queue) : elt * queue =
      match q with
      | [] -> raise QueueEmpty
      | hd :: tl -> (hd, tl)
```

Step 2: Complete the `ListQueue` functor

Re-load prioqueue.ml (with your completed functions)

1. # #use "prioqueue.ml"

Example

Now the examples in the next slides should work

Test all the new functions

Example

1. First create an empty queue called myQ
   # **let myQ = IntListQueue.empty;;**
2. Check if myQ is empty or not
   # **IntListQueue.is_empty myQ**
3. Insert 65
   # **let myQ = IntListQueue.add 65 myQ**;;
4. Print myQ
   # **IntListQueue.to_string myQ;;**
5. Insert 7
   # **let myQ = IntListQueue.add 7 myQ**;;
6. Print myQ
   # **IntListQueue.to_string myQ;;**

Test all the new functions

Example

1. Check if myQ is empty or not
   # **IntListQueue.is_empty myQ**
2. Insert 29. It should be inserted between 7 and 65
   # **let myQ = IntListQueue.add 29 myQ**;;
3. Print myQ
   # **IntListQueue.to_string myQ;;**
4. Try the take function. It should return the first element (which is 7 in this case, and it has the highest priority) and the rest of the list.
   # **let (x, myQ) = IntListQueue.take myQ**;;
5. Print myQ. It should only have [29;65] now.
   # **IntListQueue.to_string myQ;;**

Test using IntString

Test all the new functions

First create the IntStringListQueue module

```
# module IntStringListQueue = (ListQueue(IntStringCompare) :
      PRIOQUEUE with type elt = IntStringCompare.t);;
```

Test all the new functions

Example

1. First create an empty queue called myQ
   # **let myQ = IntStringListQueue.empty;;**
2. Check if myQ is empty or not
   # **IntStringListQueue.is_empty myQ**
3. Insert (65, "Anusha")
   # **let myQ = IntStringListQueue.add (65, "Anusha") myQ;;**
4. Print myQ
   # **IntStringListQueue.to_string myQ;;**
5. Insert (7, "CS51")
   # **let myQ = IntStringListQueue.add (7, "CS51") myQ**;;
6. Print myQ
   # **IntStringListQueue.to_string myQ;;**

Test all the new functions

Example

1. Check if myQ is empty or not
   # **IntStringListQueue.is_empty myQ**
2. Insert (29, "PHYS143"). It should be inserted between (7, "CS51") and (65, "Anusha")
   # **let myQ = IntStringListQueue.add (29, "PHYS143") myQ**;;
3. Print myQ
   # **IntStringListQueue.to_string myQ;;**
4. Try the take function. It should return the first element (which is (7, "CS51"), and it has the highest priority) and the rest of the list.
   # **let (x, myQ) = IntStringListQueue.take myQ**;;
5. Print myQ. It should only have [(29, "PHYS143"); (65, "Anusha")] now.
   # **IntStringListQueue.to_string myQ;;**

Important comments on ListQueue

Average and worst-case time complexity

1. On the average, the element is found in the middle of the list. So, we need to search $n/2$ items. So the average time complexity is $O(n)$
2. In the worst case, the element is found at the end of the list. So, the worst-case time complexity is also $O(n)$