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

TreeQueue

Task: Use a BST to implement the signature of PRIOQUEUE .

Step 1 Load Files

Load Files

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

Step 2: Complete the `TreeQueue` functor

Complete the types

```
type elt = Elt.t
type queue = T.collection
```

Complete empty

```
let empty = T.empty
```

Complete is_empty

```
let is_empty (q : queue) = (q = T.empty)
```

Complete add

```
let add (e : elt) (q : queue) = T.insert e q
```

Step 2: Complete the `TreeQueue` functor

Complete take

```
let take (q : queue) =
    let highest_pri = T.getmin q in
    (highest_pri, (T.delete highest_pri q))
```

Complete to_string

```
let to_string (q: queue) : string =
    T.to_string q
```

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

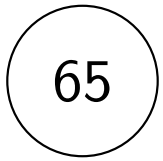
First create the IntTreeQueue module

```
# module IntTreeQueue = (TreeQueue(IntCompare) :
        PRIOQUEUE with type elt = IntCompare.t);;
```

Test add function

Example

1. First create an empty TreeQueue called myQ
   # **let myQ = IntTreeQueue.empty;;**
2. Insert 65
   # **let myQ = IntTreeQueue.add 65 myQ**;;



Example

Print the queue:

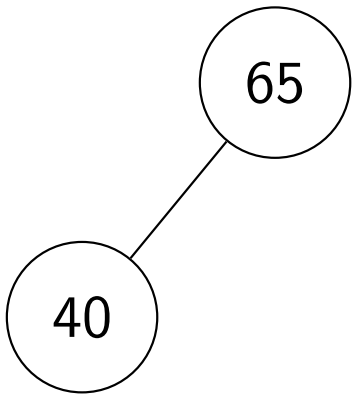
IntTreeQueue.to_string myQ;;

Output: - : string = "Branch (Leaf, [65], Leaf)"

Test add function

Example

1. Now insert 40
   # **let myQ = IntTreeQueue.add 40 myQ;;**



Example

Print the queue:

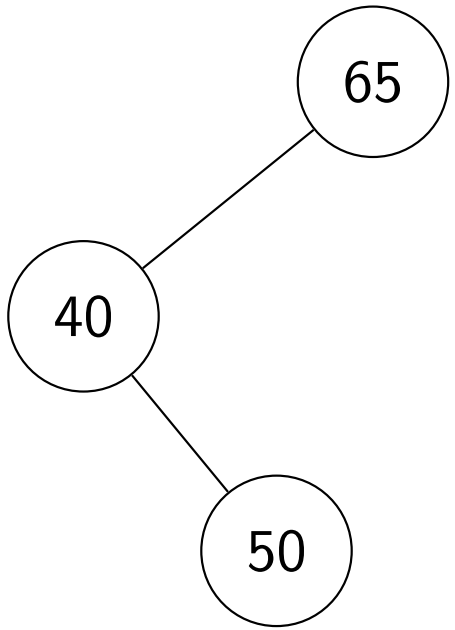
IntTreeQueue.to_string myQ;;

Output: - : string = "Branch (Branch (Leaf, [40], Leaf), [65], Leaf)"

Test add function

Example

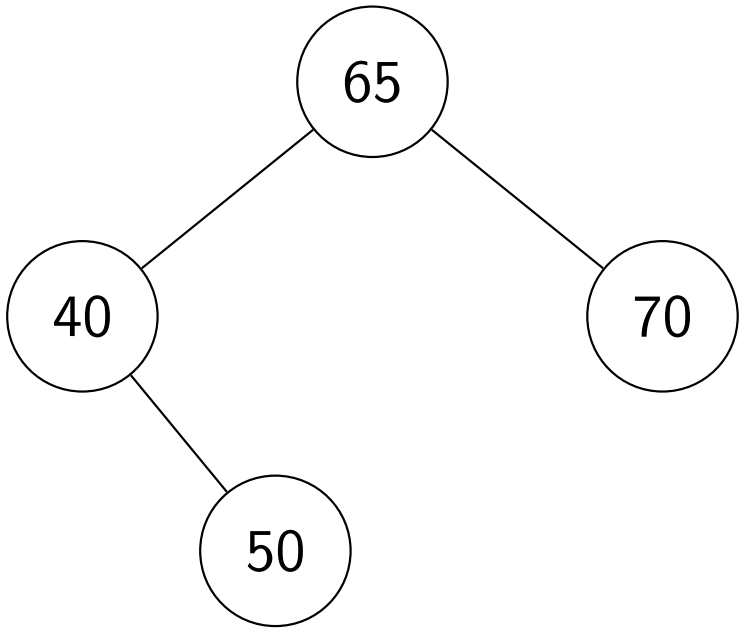
1. Now insert 50
   # **let myQ = IntTreeQueue.add 50 myQ**;;



Test add function

Example

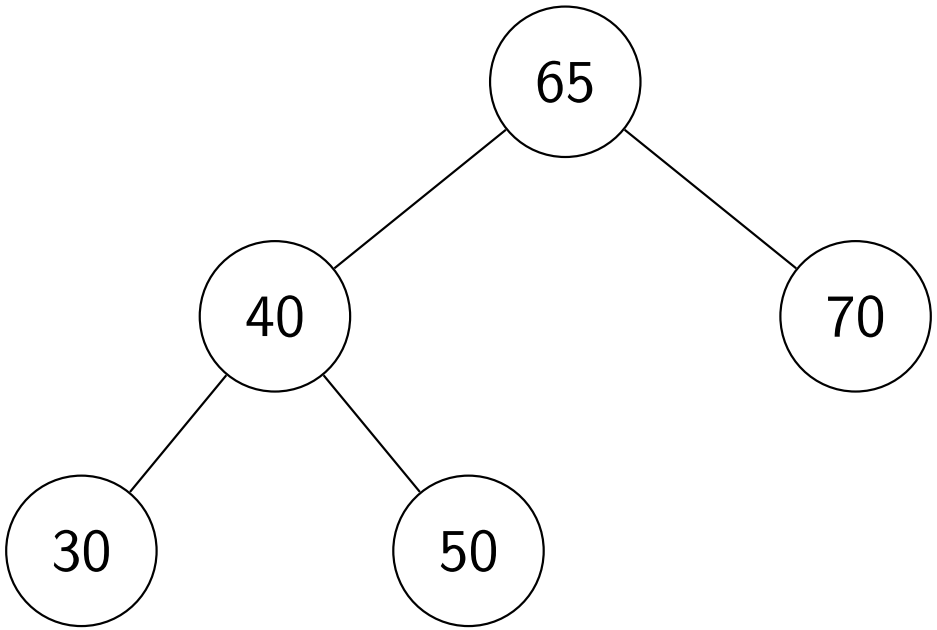
1. Now insert 70
   # **let myQ = IntTreeQueue.add 70 myQ**;;



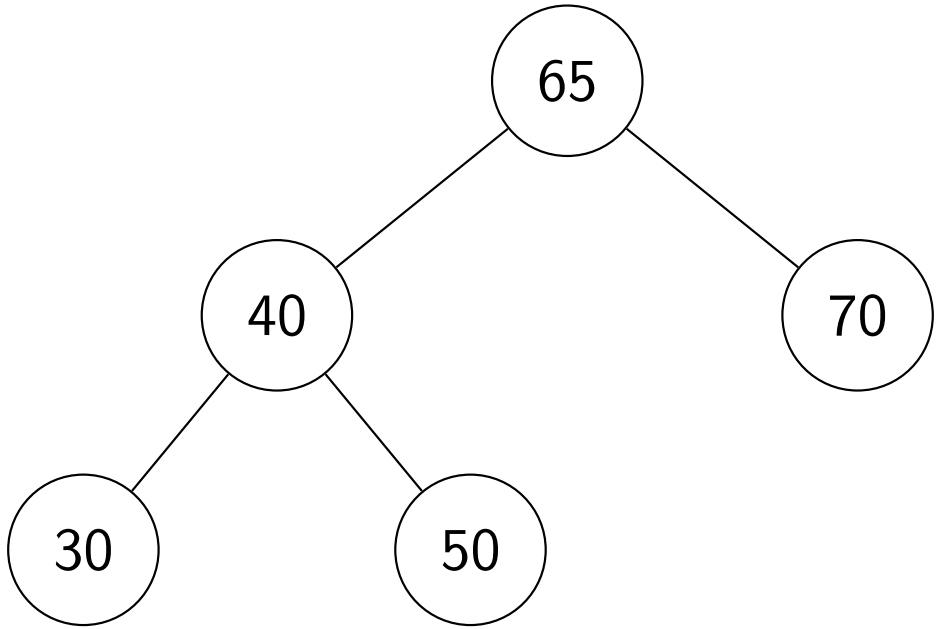
Test add function

Example

1. Now insert 30
   # **let myQ = IntTreeQueue.add 30 myQ**;;



Test add function



Print myQ using the provided "to_string" function

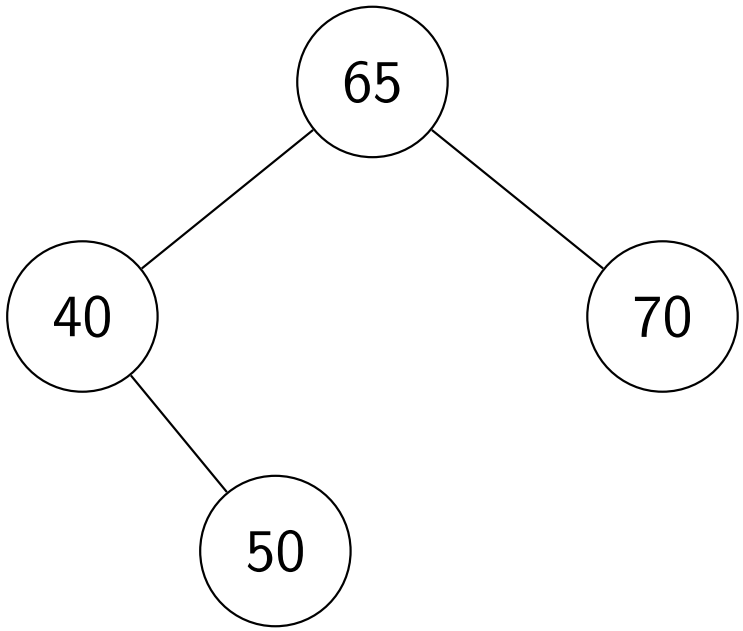
 # IntTreeQueue.to_string myQ;;

"Branch (Branch (Branch (Leaf, [30], Leaf), [40], Branch (Leaf, [50], Leaf)), [65], Branch (Leaf, [70], Leaf))"

Test take function

Example

1. The take function returns the element with the highest priority (i.e: smallest value) and the remaining queue
   # **let (hiPri, myQ) = IntTreeQueue.take myQ**;;
2. The value of **hiPri = 30** and the new **myQ** is shown below



Important comments on TreeQueue

Average and worst-case time complexity

1. On the average, the BST is nearly balanced. So, the height of the tree is $\log n$. Hence the average time complexity to search an element is $O(\log n)$
2. In the worst case, the elements are inserted in sorted order. This will result in a completely right-skewed or left-skewed tree. So, the worst-case time complexity to search an element is $O(n)$