PSET5: Ordered Collections

Anusha Murali

March 17, 2023

Tasks to do in this PSET

Part I: Implement Ordered Collections with Binary Search Trees

1. Complete insert, which inserts an arbitrary integer in a binary tree
2. Complete search, which returns true if the element is found in the binary tree, else returns false
3. Complete getmin, returns the minimum integer in a binary tree
4. Complete getmax, returns the maximum integer in a binary tree

Step 1 Load Files

Load Files

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

Step 2: Complete insert in `orderedcoll.ml`

Complete insert

```
let rec insert (x : elt) (t : tree) : tree =
  failwith "insert not implemented"
```

Re-load orderedcoll.ml (with your new insert function)

1. `# #use "orderedcoll.ml"`

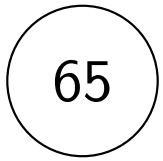
Example

Now the examples in the next slides should work

Test insert function

Example

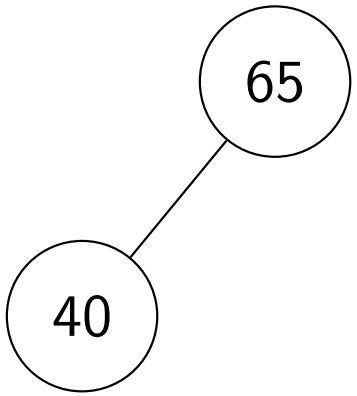
1. First create an empty binary tree called myTree
   # **let myTree = IntTree.empty;;**
2. Insert 65
   # **let myTree = IntTree.insert 65 myTree**;;



Test insert function

Example

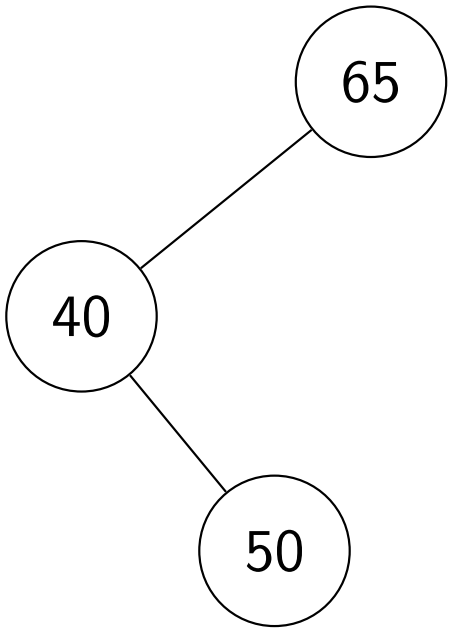
1. Now insert 40
   # **let myTree = IntTree.insert 40 myTree**;;



Test insert function

Example

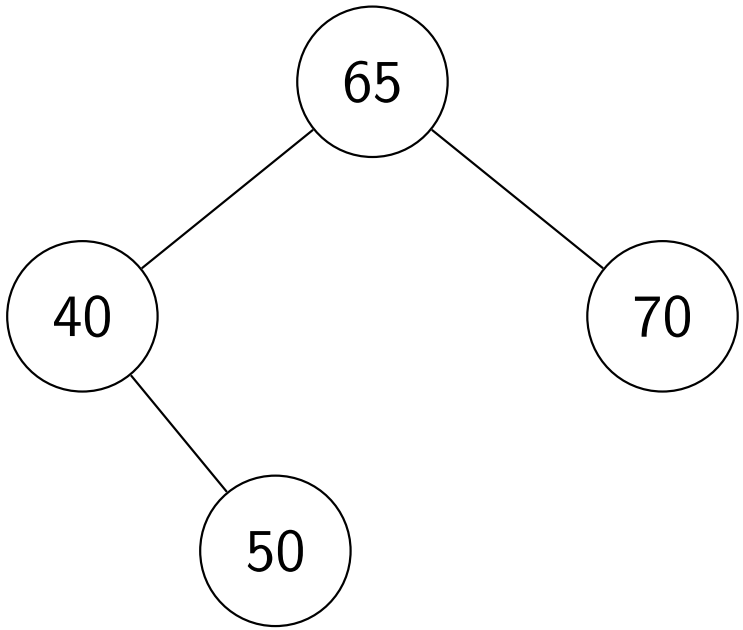
1. Now insert 50
   # **let myTree = IntTree.insert 50 myTree**;;



Test insert function

Example

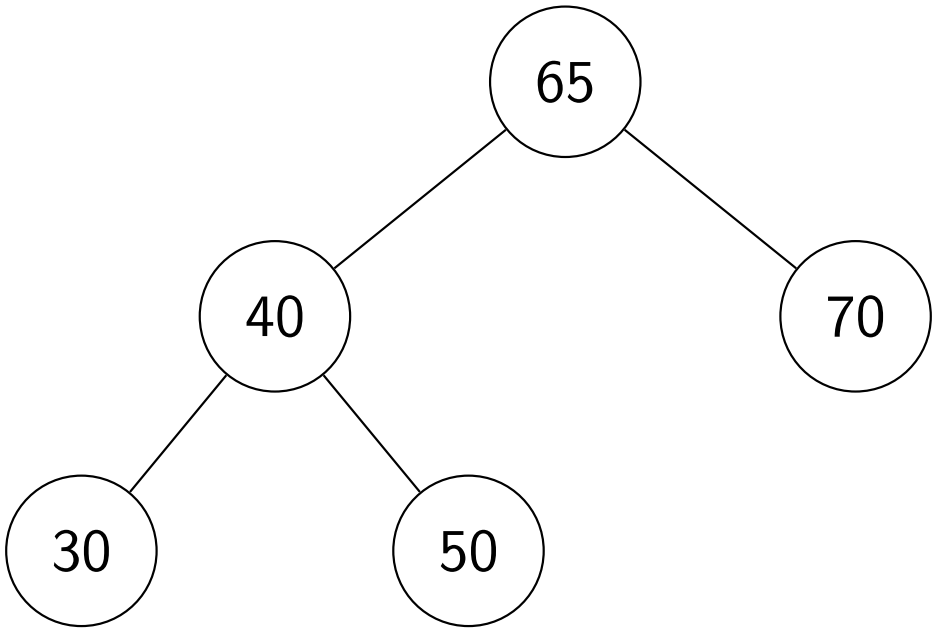
1. Now insert 70
   # **let myTree = IntTree.insert 70 myTree**;;



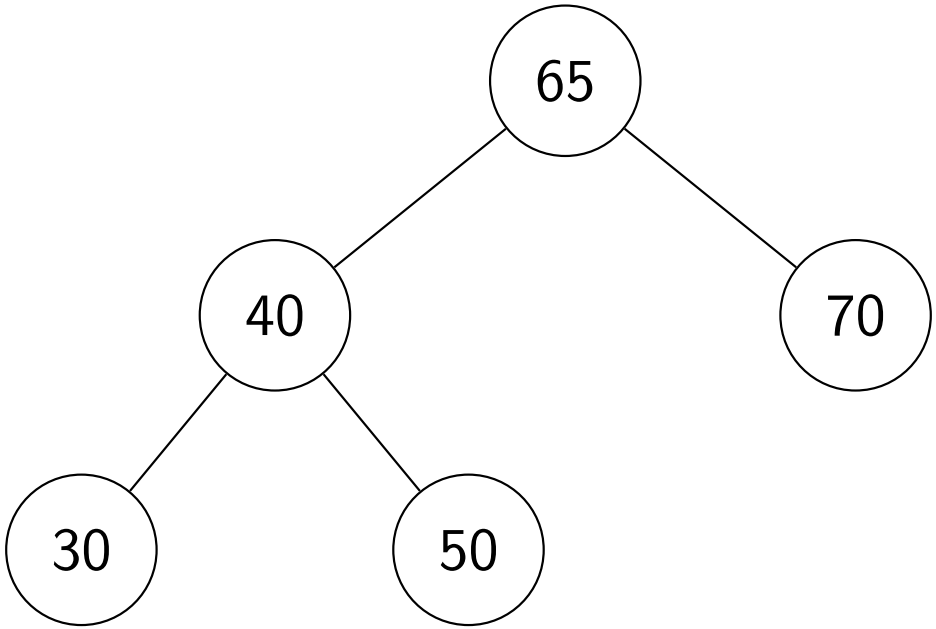
Test insert function

Example

1. Now insert 30
   # **let myTree = IntTree.insert 30 myTree;;**



Test insert function



Print myTree using the provided "to_string" function

 # IntTree.to_string myTree;;

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

Example insert function

Example insert function

```
1. let rec insert (x : elt) (t : tree) : tree =
2.  match t with
3.  | Leaf -> Branch (Leaf, [x], Leaf)
4.  | Branch (left, lst, right) ->
5.     (match Elt.compare x (List.hd lst) with
6.       | Less -> Branch (insert x left, lst, right)
7.       | Greater -> Branch (left, lst, insert x right)
8.       | Equal -> Branch (left, x :: lst, right))
```

Example insert function

Explanation of the insert function

```
1. let rec insert (x : elt) (t : tree) : tree =
2.  match t with
3.  | Leaf -> Branch (Leaf, [x], Leaf)
4.  | Branch (left, lst, right) ->
5.     (match Elt.compare x (List.hd lst) with
6.       | Less -> Branch (insert x left, lst, right)
7.       | Greater -> Branch (left, lst, insert x right)
8.       | Equal -> Branch (left, x :: lst, right))
```

Explanation

Line #1 Insert is a recursive function that takes an element of Elt type and a binary search tree as input arguments and returns the binary tree with the Elt inserted at the correct position

Example insert function

Explanation of the insert function

```
1. let rec insert (x : elt) (t : tree) : tree =
2.  match t with
3.  | Leaf -> Branch (Leaf, [x], Leaf)
4.  | Branch (left, lst, right) ->
5.     (match Elt.compare x (List.hd lst) with
6.       | Less -> Branch (insert x left, lst, right)
7.       | Greater -> Branch (left, lst, insert x right)
8.       | Equal -> Branch (left, x :: lst, right))
```

Explanation

Line #3 If t is a Leaf, just insert x at the Leaf and exit

Line #5 If t is a branch, compare x with the first element of the node (remember - it is a list)

Example insert function

Explanation of the insert function

```
1. let rec insert (x : elt) (t : tree) : tree =
2.  match t with
3.  | Leaf -> Branch (Leaf, [x], Leaf)
4.  | Branch (left, lst, right) ->
5.     (match Elt.compare x (List.hd lst) with
6.       | Less -> Branch (insert x left, lst, right)
7.       | Greater -> Branch (left, lst, insert x right)
8.       | Equal -> Branch (left, x :: lst, right))
```

Explanation

Line #6 If $x < hd$ then insert it on the left branch of this node

Line #7 If $x > hd$ then insert it on the right branch of this node

Line #8 If $x = hd$, then add it to the list at this node