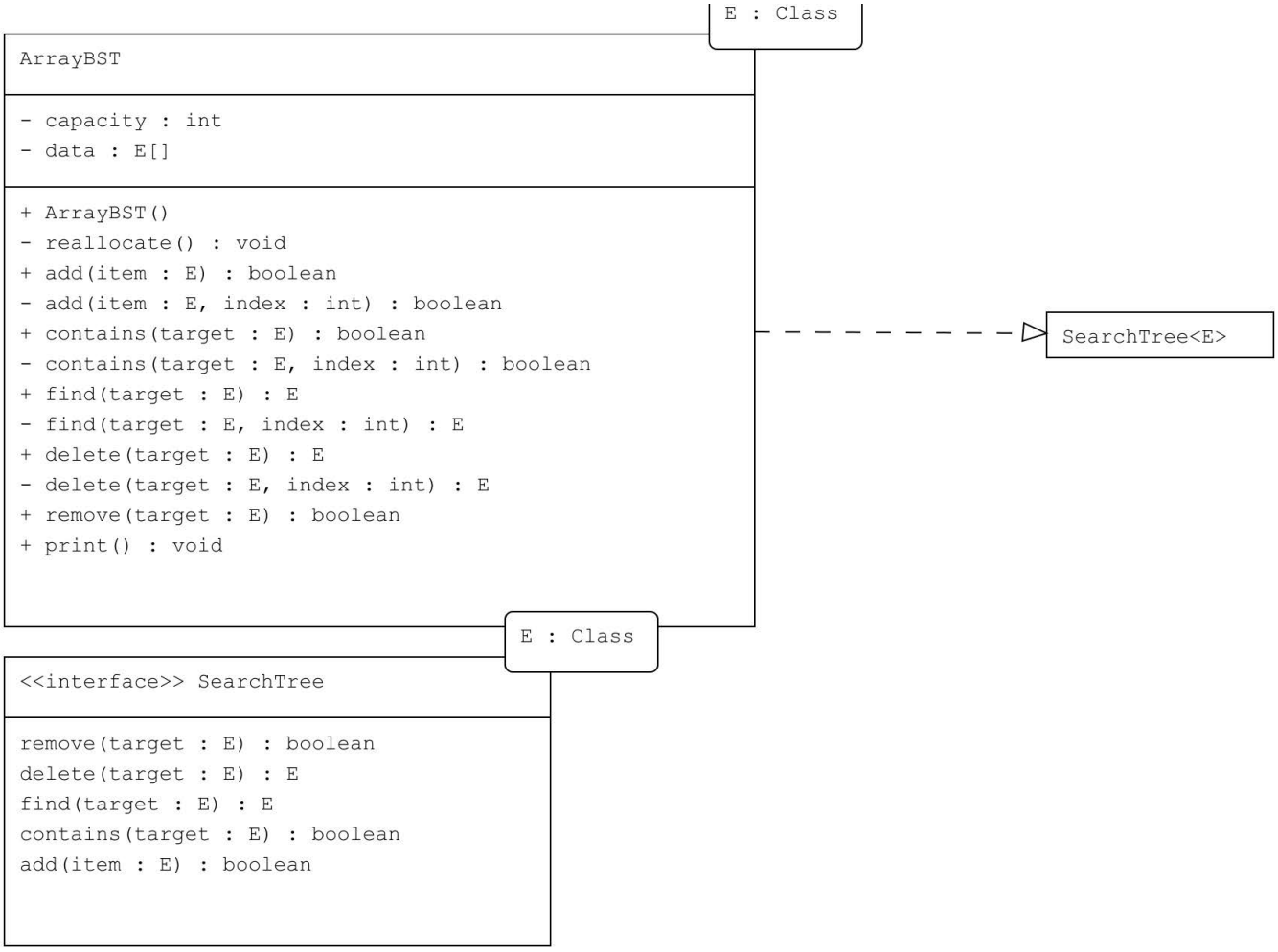
GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 5 Report

Atacan Başaran
200104004008008

1. SYSTEM REQUIREMENTS

Question 4 -> Since it stores tree data in array, it should be designed in a dynamic way and reallocate when necessary. Of course, the methods in the SearchTree interface should be implemented. Helper private methods are required to implement these methods recursively(example for index parameter). Other requirements are things that include binary search tree algorithm
Since the other questions are verbal, I explained the necessary explanations in the answers to the questions below

2. CLASS DIAGRAM



3. PROBLEM SOLUTION APPROACH

Question 4 -> I initialize tree array with certain capacity and if it needed increase capacity and reallocate it.I wrote helper methods for every recursions because i need extra parameters to recursion.All recursion methods looking root and its left/right child for operation which will be executed(add,delete,find...).I paid attention to array space while doing recursion because left/right child(index*2 +1 or +2) can be out of bound.Remove method has extra cases .I shifted to replace the deleted element and its children.

Since the other questions are verbal, I explained the necessary explanations in the answers to the questions below.

4. TEST CASES

Creating a tree based on array

```
ArrayBST<Integer> myobject = new ArrayBST<Integer>();
```

Printing empty tree

```
System.out.print("Tree: ");
myobject.print();
```

Adding elements to tree

```
myobject.add(8);
myobject.add(3);
myobject.add(10);
myobject.add(1);
myobject.add(6);
myobject.add(7);
myobject.add(4);
myobject.add(14);
myobject.add(13);
```

Printing the tree

```
System.out.print("\nTree: ");
myobject.print();
```

Finding some elements on tree

```
System.out.println("\nTree has element of "+ myobject.find(10));
System.out.println("Tree does not have -1: "+ myobject.find(-1));

System.out.println("Is tree has -1: "+ myobject.contains(13));
System.out.println("Is tree has 9: "+ myobject.contains(9));
```

Deleting elements from tree

```
System.out.println("Deleting the element "+myobject.delete(1)+"...");
System.out.println("Deleting the element "+myobject.delete(-1)+"...");
System.out.println("Has element 7 been deleted from the tree: "+myobject.delete(7));
System.out.println("Has element 9 been deleted from the tree: "+myobject.delete(9));
```

Print the latest version of tree

```
System.out.print("Tree: ");
myobject.print();
```

5. RUNNING AND RESULTS

```
Tree: - - - - - - - - - - -
Tree: 8 3 10 1 6 - 14 - - 4 7 - - 13 - - - - - - -
Tree has element of 10
Tree does not have -1: null
Is tree has -1: true
Is tree has 9: false
Deleting the element 1...
Deleting the element null...
Has element 7 been deleted from the tree: 7
Has element 9 been deleted from the tree: null
Tree: 8 3 10 - 6 - 14 - - 4 - - - 13 - - - - - -
```

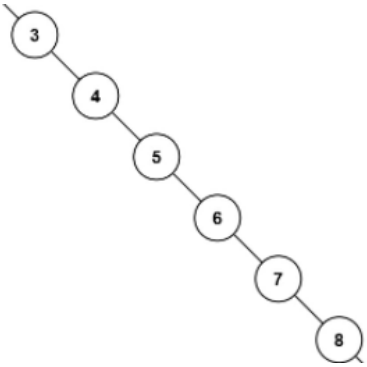
6. TIME COMPLEXITY ANALYSIS

Question 4

reallocate() method -> T(n) = θ(n) because its creating new object and copying old one into the new one

real add() method which called recursively -> Tbest(n) = θ(1)
Tworst(n) = θ(n) (sorted array tree in image)
in general case T(n) = O(logn)

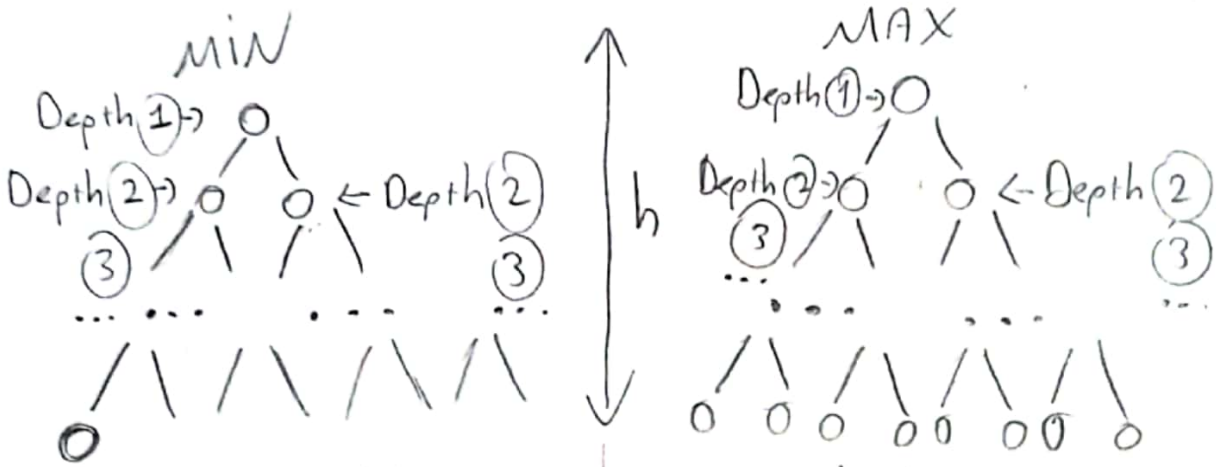
real contains() and find() method which called recursively ->
Tbest(n) = θ(1)
Tworst(n) = θ(n) (sorted array tree in image)
in general case T(n) = O(logn)



real delete() and remove() method which called recursively ->
Tworst(n) = θ(n) (sorted array tree in image)
in general case T(n) = O(logn)

print() method -> T(n) = θ(n) because it traverse all array

1) a) A complete binary tree can be several different form so i calculate min, max depth of tree which has height h.



Total depth of MIN tree:

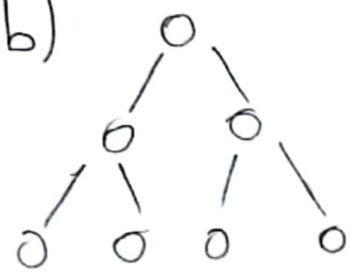
$$\left(\sum_{i=1}^{h} i \cdot 2^{i-1}\right) - h(2^{h-1} - 1)$$

Total depth of MAX tree:

$$\sum_{i=1}^{h} i \cdot 2^{i-1}$$

Difference of them just last level

b)



Searching:

Best case: $O(1)$ → Target is root

Waist case: $O(\log n)$ → Target is leaf

Average case: Target can be in every node. So we should calculate all possible amount of comparison which equals total depth of tree. After that divide it to node amount for seeing possibility (average comparison)

Average comparison: Total depth / node amount

n(node amount) = 7 : $\sum_{i=1}^{3} i \cdot 2^{i-1} / 7$

height = 3

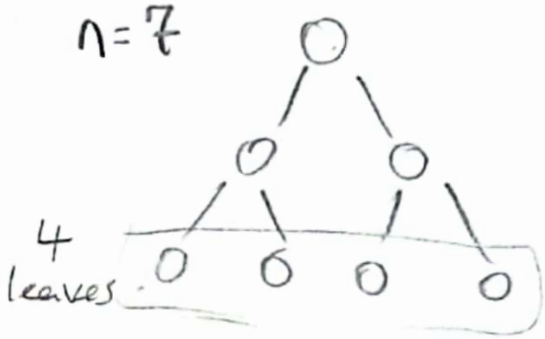
$= 17/7 = 2.43 \cong \log 7 \Rightarrow$ Average: $O(\log n)$

c) Full binary tree is a tree where each child has 2 childre or 0 children (leaf).

Number of leaves: $(n+1)/2$

$= (7+1)/2$

$= 4$



Number of internal nodes:

Total nodes - leaves

$n - \frac{(n+1)}{2} = \frac{n-1}{2}$

$= (7-1)/2$

$= 3$ internal nodes

2) A=(30,30) , B=(20,15) , C=(50,40) , D=(10,12) , E(40,20)
F=(25,60) , G=(15,25)