GTU Department of Computer Engineering

CSE 222/505 - Spring 2022

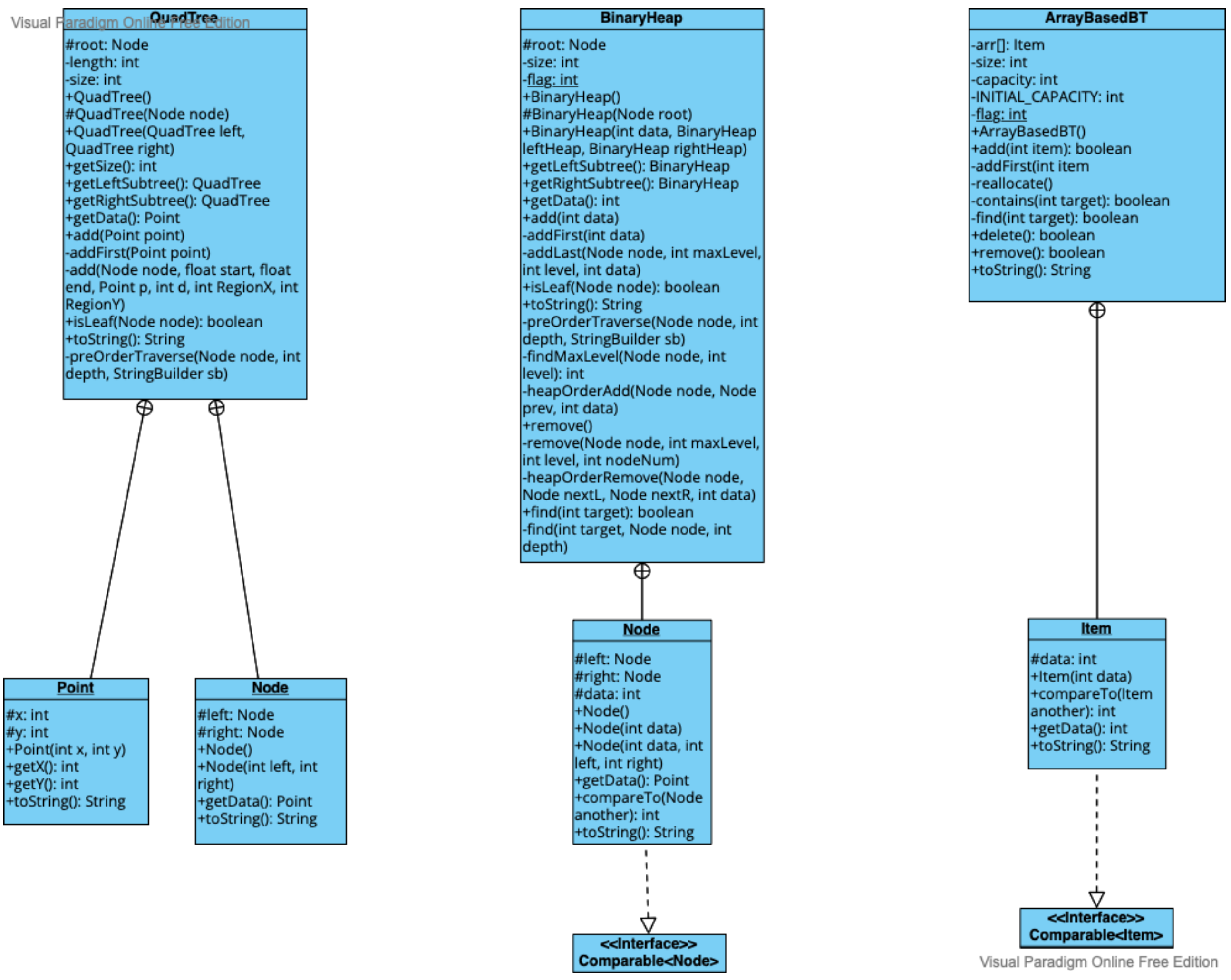
Homework 5 Report

Sena Özbelen
1901042601

1. System Requirements

- Quad tree should be given to right parameters.
- Points in quad tree should have positive coordinates.
- Binary heap should be given to a complete binary tree.
- Binary heap should be given to integer data.
- Array based binary search tree should be given to a complete binary tree in order.
- Only the last item in the array can be removed.
- Items can be added only to the array on the basis of the BST array order.
- Array based binary search tree should be given to integer data

2. Class Diagram


Visual Paradigm Online Free Edition
QuadTree
#root: Node
-length: int
-size: int
+QuadTree()
#QuadTree(Node node)
+QuadTree(QuadTree left, QuadTree right)
+getSize(): int
+getLeftSubtree(): QuadTree
+getRightSubtree(): QuadTree
+getData(): Point
+add(Point point)
-addFirst(Point point)
-add(Node node, float start, float end, Point p, int d, int RegionX, int RegionY)
+isLeaf(Node node): boolean
+toString(): String
-preOrderTraverse(Node node, int depth, StringBuilder sb)
Point
#x: int
#y: int
+Point(int x, int y)
+getX(): int
+getY(): int
+toString(): String
Node
#left: Node
#right: Node
+Node()
+Node(int left, int right)
+getData(): Point
+toString(): String
BinaryHeap
#root: Node
-size: int
-flag: int
+BinaryHeap()
#BinaryHeap(Node root)
+BinaryHeap(int data, BinaryHeap leftHeap, BinaryHeap rightHeap)
+getLeftSubtree(): BinaryHeap
+getRightSubtree(): BinaryHeap
+getData(): int
+add(int data)
-addFirst(int data)
-addLast(Node node, int maxLevel, int level, int data)
+isLeaf(Node node): boolean
+toString(): String
-preOrderTraverse(Node node, int depth, StringBuilder sb)
-findMaxLevel(Node node, int level): int
-heapOrderAdd(Node node, Node prev, int data)
+remove()
-remove(Node node, int maxLevel, int level, int nodeNum)
-heapOrderRemove(Node node, Node nextL, Node nextR, int data)
+find(int target): boolean
-find(int target, Node node, int depth)
Node
#left: Node
#right: Node
#data: int
+Node()
+Node(int data)
+Node(int data, int left, int right)
+getData(): Point
+compareTo(Node another): int
+toString(): String
<<Interface>>
Comparable<Node>
ArrayBasedBT
-arr[]: Item
-size: int
-capacity: int
-INITIAL_CAPACITY: int
-flag: int
+ArrayBasedBT()
+add(int item): boolean
-addFirst(int item
-reallocate()
-contains(int target): boolean
-find(int target): boolean
+delete(): boolean
+remove(): boolean
+toString(): String
Item
#data: int
+Item(int data)
+compareTo(Item another): int
+getData(): int
+toString(): String
<<Interface>>
Comparable<Item>
Visual Paradigm Online Free Edition

3. Problem Solution Approach

1-) a-) $h=1 \rightarrow 1$

$h=2 \rightarrow 1+2+2=5$

$h=3 \rightarrow 1+2+2+3+3+3+3=17$

$h=k \rightarrow 1.2^0+2.2^1+3.2^2+\dots+k.2^{k-1} = \sum_{k=1}^{n} k.2^{k-1}$

$$S = \sum_{k=0}^{n} x^k = \frac{1-x^{n+1}}{1-x}$$

$$S = \frac{d}{dx}\left(\sum_{k=0}^{n} x^k\right) = \frac{d}{dx}\left(\frac{1-x^{n+1}}{1-x}\right) = \frac{-nx^n+nx^{n+1}-x^n+1}{(1-x)^2}$$

$$S = \sum_{k=0}^{n} kx^{k-1} = \frac{-n.x^n+nx^{n+1}-x^n+1}{(1-x)^2}$$

$$x=2 \longrightarrow S = \sum_{k=1}^{n} k.2^{k-1} = \frac{-n.2^n+n.2^{n+1}-2^n+1}{(-1)^2} = 2^n(n-1)+1$$

• When the height is h, the total depth is $2^h(h-1)+1$ for a full binary tree. For a complete binary tree,

$2^{(h-1)}(h-2)+1 < S <= 2^h(h-1)+1$ is the interval of the sum

b-) $n=2^h-1$ for a full binary tree. Therefore, a complete binary tree may have,

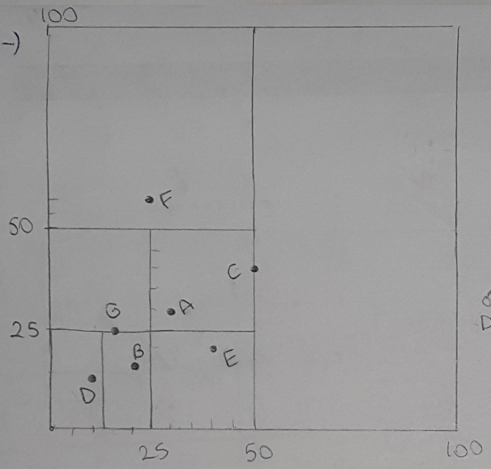
$2^{h-1}-1 < n <= 2^h-1$ $\quad$ $2^{h-1}-1=n$, $h=\log(n+1)+1$ $\quad|\quad$ $2^h-1=n$, $h=\log(n+1)$

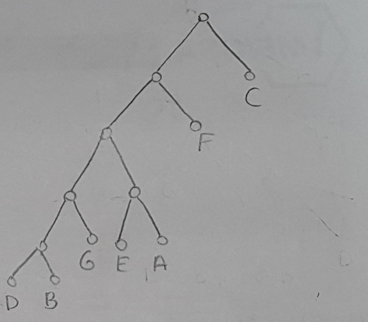
$T(n)=O(h) \sim O(\log n)$

c-) $h=\log(n+1)$ so the number of leaves is 2^{h-1} and the number of internal nodes is $2^h-1-2^{h-1}=2^{h-1}-1$

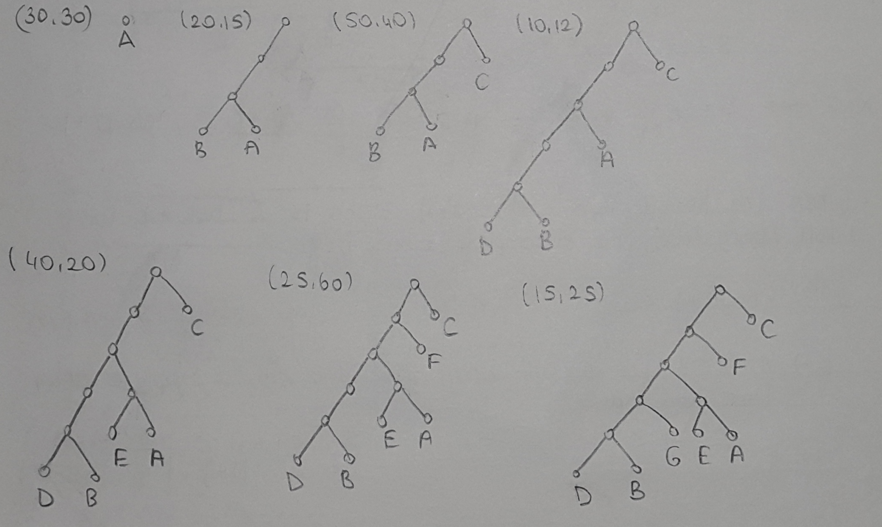
Restrictions depend on memory allocation.

2-)





· binary tree representation of quadtree



For the third question, we are asked to implement a binary heap using nodes. I used Node inner class to store the data. To add a new element, add method finds the very last node of the heap and adds the element to the end to preserve the structure. Since binary heap's order should be preserved as well, heapOrderAdd method compares the new item to the parents of it one by one and swaps them. To remove the top element, remove method finds the very last node and store the data in the very last node in the root. Then, to preserve the heap order, heapOrderRemove method compares elements and chooses the smallest element and swaps them.

Analysis: Since binary heap always has a complete binary tree structure, using nodes in the implementation is less efficient than using array because every child and parent has an accessible relation so we can traverse among items fast in the array version but in the node version, recursive calls with left, right nodes are required. Also, to find the end of the heap, there should be checking conditions. However, we can easily add the element to the end in the array. Also, while comparing node to another node requires to store the another node every time whilst all items are accessible in the array version without storing.

For the fourth question, we are asked to implement an array-based binary search tree. I used a regular array to keep the data. Since there should be a relation between parents and children, the order should be preserved in the array. That is why only addition to the end and removing from the end are allowed (it is similar to binary heap).

Analysis: Binary search tree cannot always be a complete binary tree since the structure depends on the elements but using arrays requires a complete binary tree structure because there should be a relation between parents and children. We cannot observe such a structure in the BST. Thus, using arrays limits the addition and removing operations.

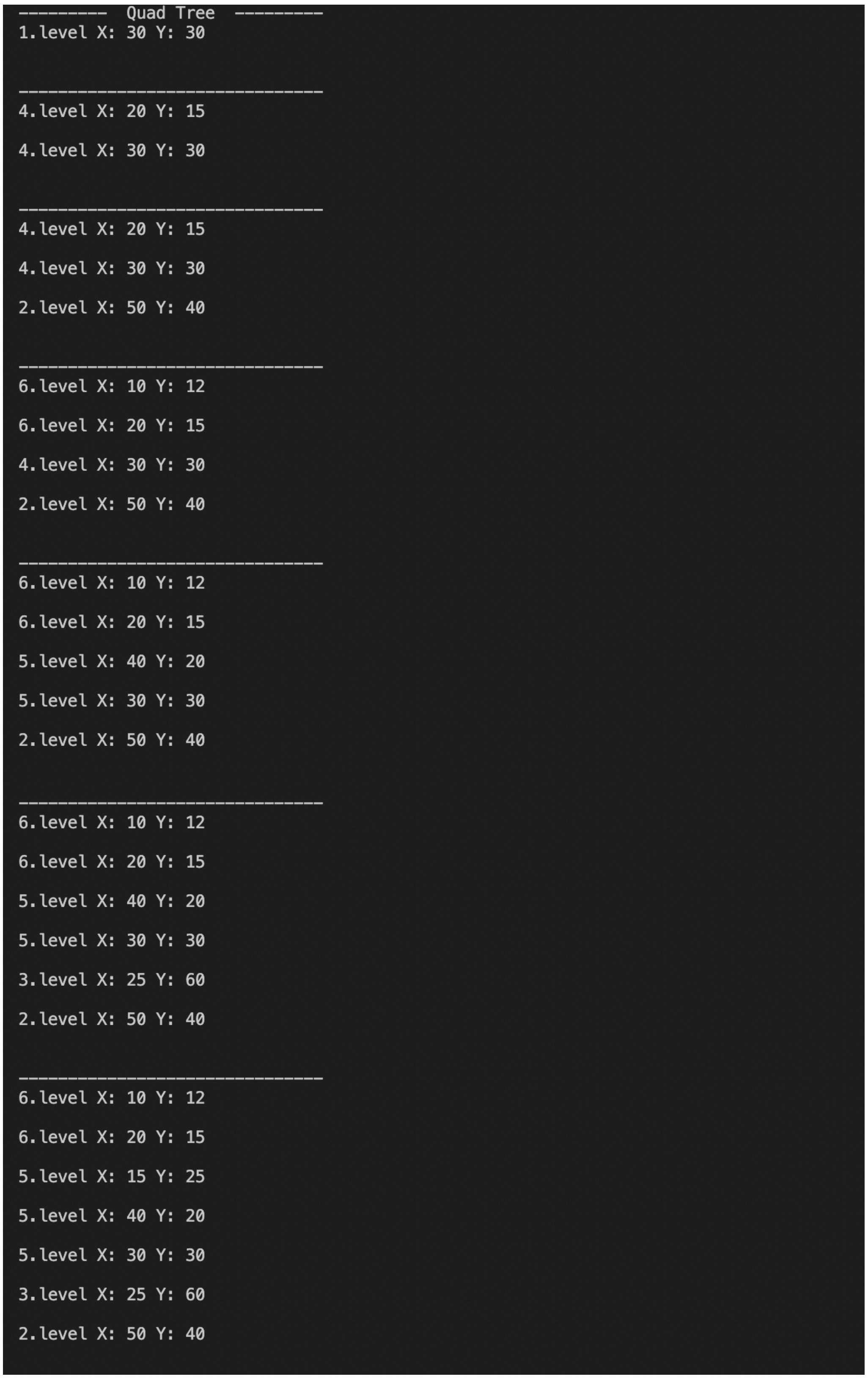
4. Test Cases

For the second question,
-Add the given items in the pdf respectively
-Show the current quad tree after each addition

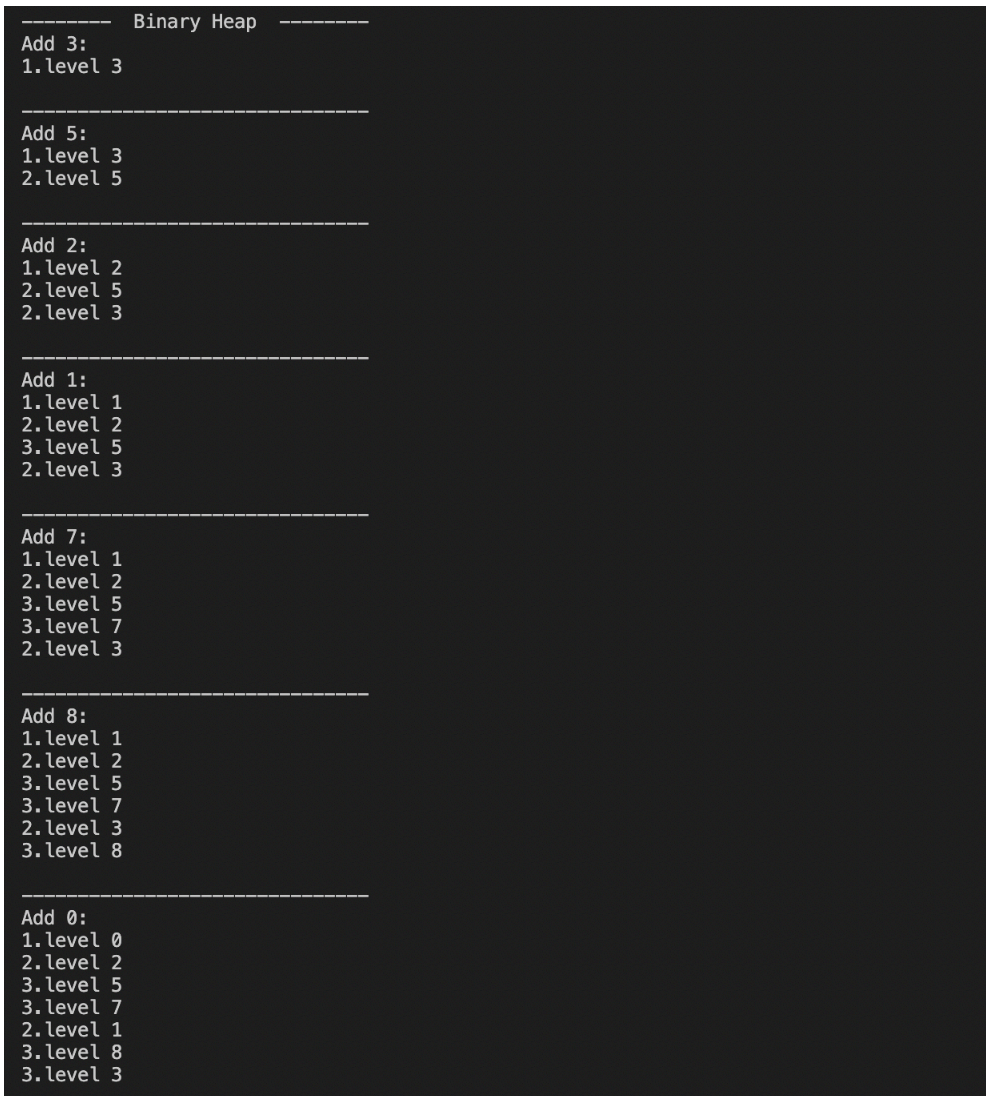
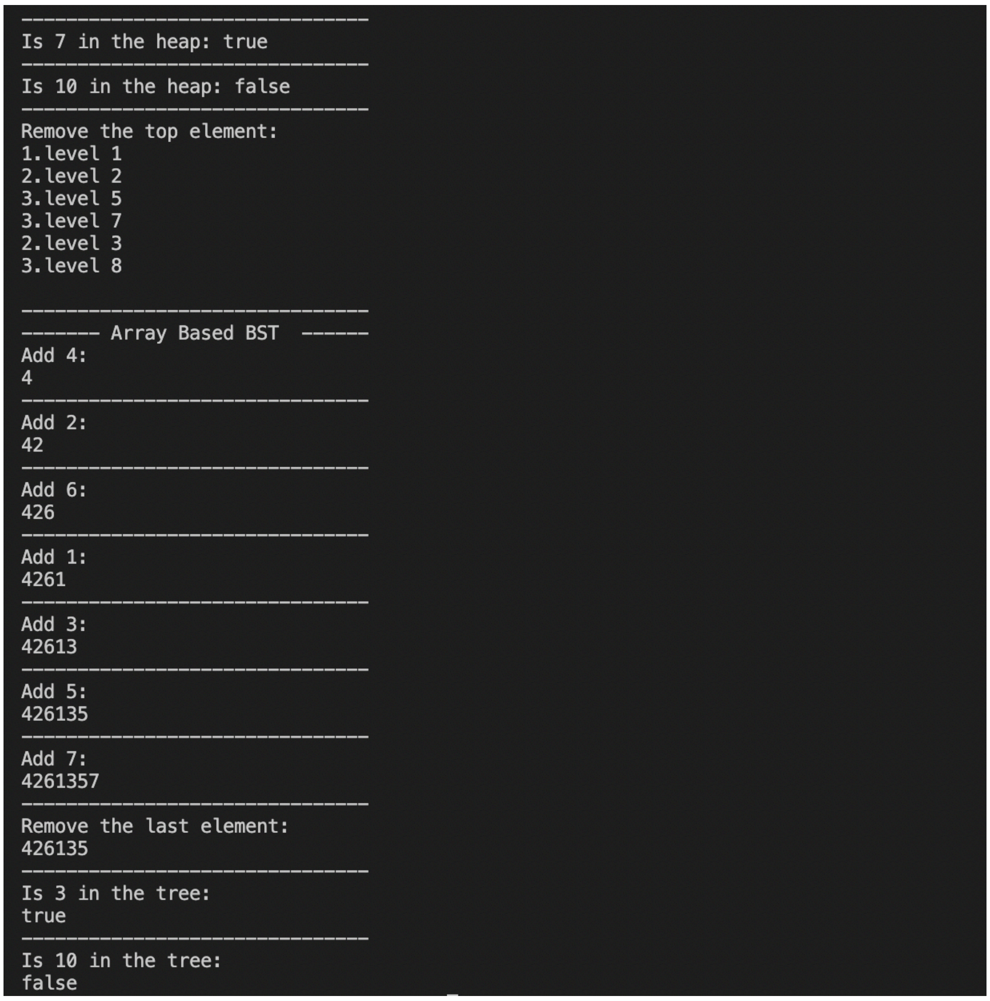
For the third question,
-Items = 3, 5, 2, 1, 7, 8, 0
-Add the items respectively
-Find the number 7
-Find the number 10
-Remove the top element
-Show the current binary heap after each addition

For the fourth question,
-Array= {4,2,6,1,3,5,7}
-Add each element above. This order preserves the relation in the BST.
-Remove the last element in the tree which is 7
-Find the number 3
-Find the number 10

5. Running Command and Results



```
----------  Quad Tree  ----------
1.level X: 30 Y: 30


--------------------------------
4.level X: 20 Y: 15

4.level X: 30 Y: 30


--------------------------------
4.level X: 20 Y: 15

4.level X: 30 Y: 30

2.level X: 50 Y: 40


--------------------------------
6.level X: 10 Y: 12

6.level X: 20 Y: 15

4.level X: 30 Y: 30

2.level X: 50 Y: 40


--------------------------------
6.level X: 10 Y: 12

6.level X: 20 Y: 15

5.level X: 40 Y: 20

5.level X: 30 Y: 30

2.level X: 50 Y: 40


--------------------------------
6.level X: 10 Y: 12

6.level X: 20 Y: 15

5.level X: 40 Y: 20

5.level X: 30 Y: 30

3.level X: 25 Y: 60

2.level X: 50 Y: 40


--------------------------------
6.level X: 10 Y: 12

6.level X: 20 Y: 15

5.level X: 15 Y: 25

5.level X: 40 Y: 20

5.level X: 30 Y: 30

3.level X: 25 Y: 60

2.level X: 50 Y: 40
```


```
---------  Binary Heap  ---------
Add 3:
1.level 3

--------------------------------
Add 5:
1.level 3
2.level 5

--------------------------------
Add 2:
1.level 2
2.level 5
2.level 3

--------------------------------
Add 1:
1.level 1
2.level 2
3.level 5
2.level 3

--------------------------------
Add 7:
1.level 1
2.level 2
3.level 5
3.level 7
2.level 3

--------------------------------
Add 8:
1.level 1
2.level 2
3.level 5
3.level 7
2.level 3
3.level 8

--------------------------------
Add 0:
1.level 0
2.level 2
3.level 5
3.level 7
2.level 1
3.level 8
3.level 3
```

```
--------------------------------
Is 7 in the heap: true
--------------------------------
Is 10 in the heap: false
--------------------------------
Remove the top element:
1.level 1
2.level 2
3.level 5
3.level 7
2.level 3
3.level 8

--------------------------------
-------- Array Based BST  -------
Add 4:
4
--------------------------------
Add 2:
42
--------------------------------
Add 6:
426
--------------------------------
Add 1:
4261
--------------------------------
Add 3:
42613
--------------------------------
Add 5:
426135
--------------------------------
Add 7:
4261357
--------------------------------
Remove the last element:
426135
--------------------------------
Is 3 in the tree:
true
--------------------------------
Is 10 in the tree:
false
```