GIT Department of Computer Engineering

CSE 222/505 - Spring 2021
Homework 4 # Report

Okan Torun

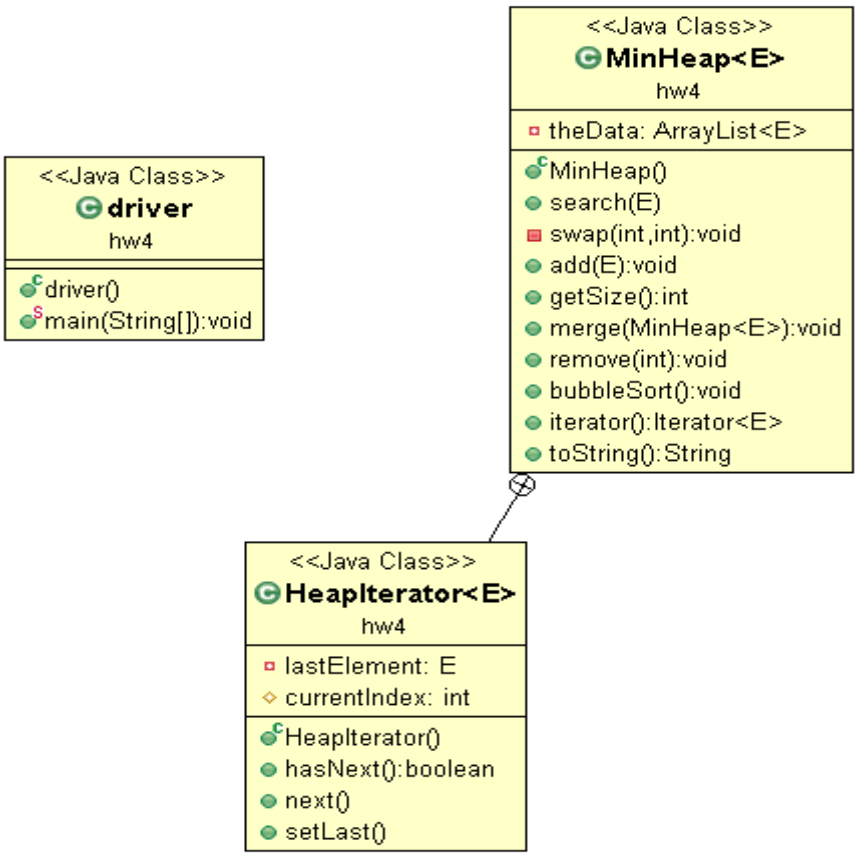
1801042662

PART 1

1) SYSTEM REQUIREMENTS

Searching, adding and removing operations can be performed on the Min Heap structure in the system. The Add method adds the element you want to add to the Heap with the MinHeap principles. This can be the same element. The Remove method is the largest indexth element in the heap with the index parameter.If the requested element is not found in the heap, it throws an exception. Search method notifies that the element you check is found in the heap.

2) CLASS DIAGRAMS



3)PROBLEM SOLUTION APPROACH

We were asked to apply operations such as search, remove and merge in the MinHeap class. I tried to apply the Completed Binary Tree principle on Arraylist. The indexes where the elements will be placed was important. For this, I applied the algorithm leftchild = (2 * parent) +1. I preserved the heap structure with the swap method using the algorithm, I created a new array to delete the ith largest element from the heap, after storing and sorting the elements in this array, I found the element that should be deleted and deleted this element from the original heap. I added an old heap to the end and fix it.

The most important issue was not to break the stack principle after doing these operations, I did the add and delete operations with the help of the last index and I kept the heap structure with the swap method after each operation.

4)TEST CASES

```
MinHeap<Integer>tempHeap=new MinHeap<>();
MinHeap<Integer>tempHeap2=new MinHeap<>();
System.out.print("*******Adding Heap1*******\n");
tempHeap.add(8);
tempHeap.add(3);
tempHeap.add(5);
tempHeap.add(10);
tempHeap.add(15);
tempHeap.add(9);
System.out.println(tempHeap.toString());

System.out.println("******REMOVİNG******\n");
tempHeap.remove(3);
System.out.print("\n\nHeap1 Again\n"+tempHeap.toString());

tempHeap2.add(2);
tempHeap2.add(4);
tempHeap2.add(6);
tempHeap.add(35);
tempHeap.add(40);
tempHeap.add(1);
System.out.print("\n\nAdding Heap2\n"+tempHeap.toString());



tempHeap.remove(6);
System.out.print("\n\nHeap2 Again\n"+tempHeap.toString());



System.out.println("\nSearched Element 10 : "+tempHeap.search(10)+" found");
System.out.println("\nSearched Element 4 : "+tempHeap.search(4));



tempHeap.merge(tempHeap2);
System.out.println("\n**********MERGİNG***************");
System.out.println("\nMerged Completed");
System.out.print("Merge Heap\n"+tempHeap.toString());

tempHeap.remove(6);
System.out.print("\n\nMerged Heap print again \n"+tempHeap.toString());
```

5)RUNNİNG AND RESULTS

```
*******Adding Heap1*******
3
8
5
10
15
9

******REMOVİNG******

Removed->9


Heap1 Again
3
8
5
10
15



Adding Heap2
1
3
5
8
15
35
40
10
Removed->5


Heap2 Again
1
3
8
15
35
40
10
```

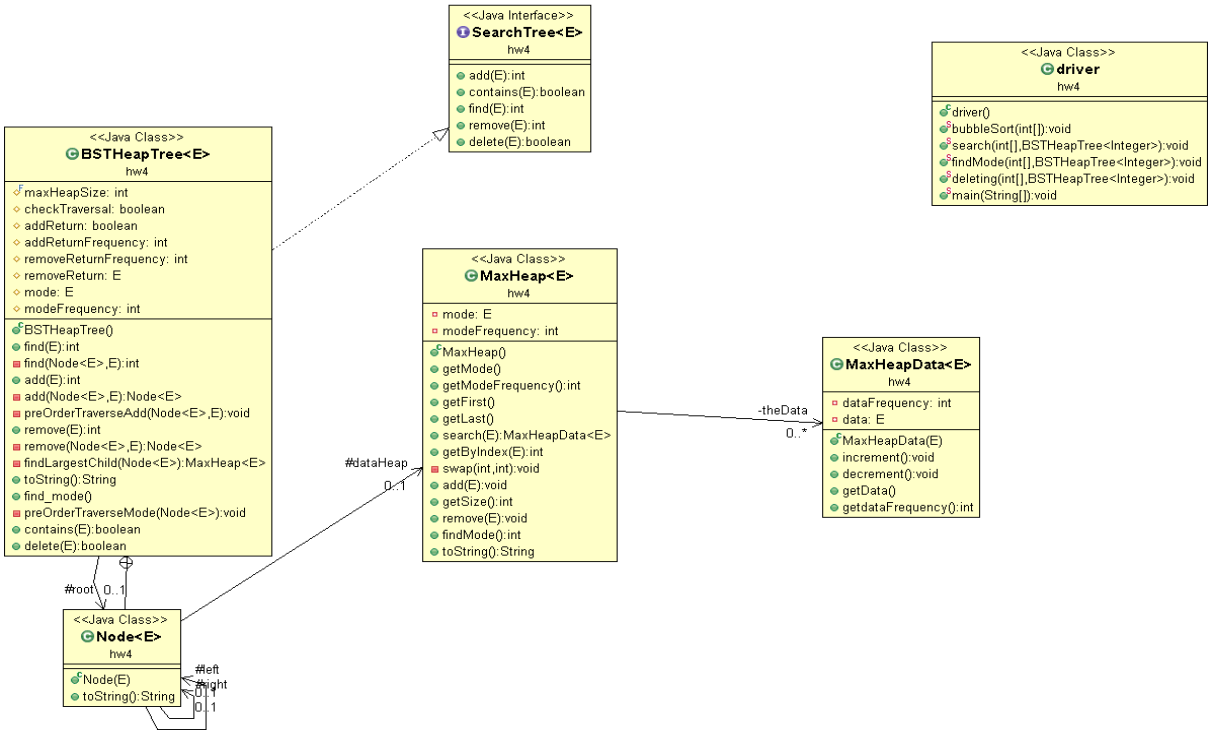
```
Searched Element 10 : 10 found

Searched Element 4 : null



**********MERGİNG***************

Merged Completed
Merge Heap
1
2
8
3
6
40
10
15
4
35
Removed->6


Merged Heap print again
1
2
8
3
40
10
15
4
35
```

PART 2

1) SYSTEM REQUIREMENTS

BSTHeapTree is built from the system that holds one heap for each node in the system. The general architecture of the structure is BST, but the elements held by each node work on the Heap principle. For example, if the number of elements of the heap we are in is less than 7, the element is added to the heap we are in and converted back to MaxHeap format. If the heap we are in is full, the target value is compared with the root of each heap with the BST principle and divided into nodes accordingly. If the element we want to add exists in the tree, a new place is not opened again and the frequency of the value that was added before is increased. Likewise, if the number of the element we want to delete is more than one, the element is not removed from the heap and its number is reduced. Find_mode () method is available to find the mode of the tree. The add, remove, and find methods return the frequency of the desired item in the tree.

2) CLASS DIAGRAMS



3)PROBLEM SOLUTION APPROACH

The problem was to keep a heap in each node of BST with the BST principle and also to store the number information of the element in the heap. I created the MaxHeapData class to store the elements in the heap. In this class I saved the data and frequency of the heap. The Node class was holding the MaxHeap object as an element. In this way, I could easily access the elements I wanted to find in the heap while traversing on the node.

When I wanted to delete an element from Tree, I reached the node where the element is located and reduced the frequency if the element is more than one, if it is one, I removed it from the heap and fixed the heap If the node is completely emptied, I deleted the empty node.

In the mode finding process, I traverse the whole tree, compare each heap's mode with other heap modes and keep the data of the mod.

In addition, I checked whether the same element was added before using the find method. If it was added before, I reached the heap where the element is located using the preTraverseAdd method and completed the adding process.

4)TEST CASES

```
BSTHeapTree<Integer>temp=new BSTHeapTree<>();

System.out.println("*****Adding Element******");
temp.add(5);

System.out.println("*****Searching Element******");
temp.find(5);

System.out.println("*****Find Mode******");
temp.find_mode();

System.out.println("*****Deleting Element******");
temp.remove(5);
```

5)RUNNİNG AND RESULTS

****The frequency of the items in the heap******

```
1) Occurrence of 2 in Array :1--- Occurrence in HeapTree 1
2) Occurrence of 3 in Array :5--- Occurrence in HeapTree 5
3) Occurrence of 4 in Array :1--- Occurrence in HeapTree 1
4) Occurrence of 5 in Array :1--- Occurrence in HeapTree 1
5) Occurrence of 6 in Array :3--- Occurrence in HeapTree 3
6) Occurrence of 7 in Array :1--- Occurrence in HeapTree 1
7) Occurrence of 8 in Array :1--- Occurrence in HeapTree 1
8) Occurrence of 9 in Array :0--- Occurrence in HeapTree -1
9) Occurrence of 12 in Array :1--- Occurrence in HeapTree 1
10) Occurrence of 13 in Array :0--- Occurrence in HeapTree -1
11) Occurrence of 15 in Array :1--- Occurrence in HeapTree 1
12) Occurrence of 16 in Array :0--- Occurrence in HeapTree -1
13) Occurrence of 18 in Array :1--- Occurrence in HeapTree 1
14) Occurrence of 19 in Array :1--- Occurrence in HeapTree 1
15) Occurrence of 20 in Array :0--- Occurrence in HeapTree -1
16) Occurrence of 21 in Array :1--- Occurrence in HeapTree 1
17) Occurrence of 22 in Array :0--- Occurrence in HeapTree -1
18) Occurrence of 24 in Array :1--- Occurrence in HeapTree 1
19) Occurrence of 25 in Array :0--- Occurrence in HeapTree -1
20) Occurrence of 27 in Array :1--- Occurrence in HeapTree 1
21) Occurrence of 28 in Array :1--- Occurrence in HeapTree 1
22) Occurrence of 29 in Array :0--- Occurrence in HeapTree -1
23) Occurrence of 31 in Array :1--- Occurrence in HeapTree 1
24) Occurrence of 32 in Array :3--- Occurrence in HeapTree 3
25) Occurrence of 33 in Array :1--- Occurrence in HeapTree 1
26) Occurrence of 34 in Array :1--- Occurrence in HeapTree 1
27) Occurrence of 35 in Array :0--- Occurrence in HeapTree -1
28) Occurrence of 36 in Array :1--- Occurrence in HeapTree 1
29) Occurrence of 37 in Array :0--- Occurrence in HeapTree -1
30) Occurrence of 38 in Array :1--- Occurrence in HeapTree 1
```

********Mode of Heap**********

```
Mode in Array is : 2751--- Mode in HeapTree 2751
```

********Delete Element From Heap*********

```
Before removing 3 Occurrence in HeapTree 5
1)After removing 3 Occurrence in HeapTree 4

Before removing 3 Occurrence in HeapTree 4
2)After removing 3 Occurrence in HeapTree 3

Before removing 4 Occurrence in HeapTree 1
3)After removing 4 Occurrence in HeapTree -1

Before removing 6 Occurrence in HeapTree 3
4)After removing 6 Occurrence in HeapTree 2

Before removing 6 Occurrence in HeapTree 2
5)After removing 6 Occurrence in HeapTree 1

Before removing 8 Occurrence in HeapTree 1
6)After removing 8 Occurrence in HeapTree -1

Non Existing: Before removing 13 Occurrence in HeapTree -1
After removing 13 Occurrence in HeapTree -1

Before removing 15 Occurrence in HeapTree 1
7)After removing 15 Occurrence in HeapTree -1

Before removing 19 Occurrence in HeapTree 1
8)After removing 19 Occurrence in HeapTree -1

Non Existing: Before removing 22 Occurrence in HeapTree -1
After removing 22 Occurrence in HeapTree -1
```