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

ÇAĞRI ÇAYCI
1901042629

1. SYSTEM REQUIREMENTS

BinaryHeap

- An inner class named Node is needed to keep links of children and parent of a node and keep data of node.

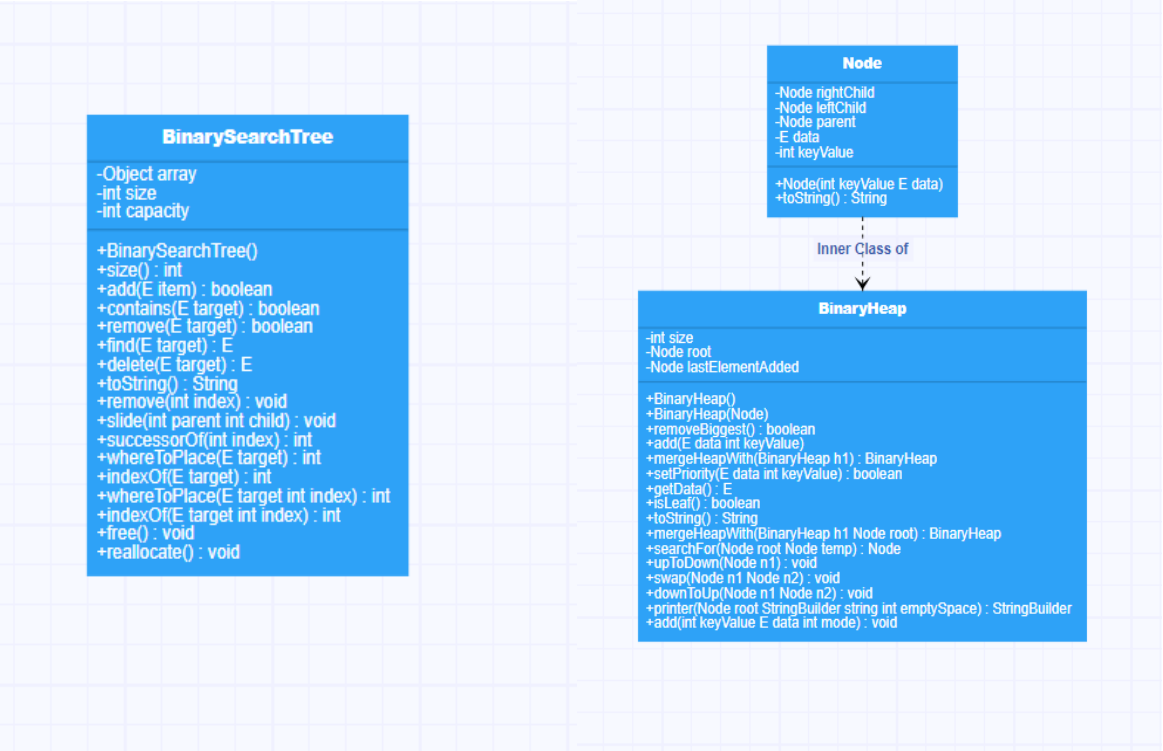
  Node:

  - 3 link is needed for class. One for right child, one for left child and one for parent. 2 data fields are also needed to keep data of node and key value(priority) of node.
  - A constructor is needed for Node class to create instances.
  - An overriding toString method is needed to print node.
- Three data fields are needed. One integer to keep size of the Heap, two nodes to keep root of the Heap and to keep parents of the element which is added last.
- Two constructors are needed. One of constructors is no parameter constructor (creates an empty BinaryHeap). The other one is one parameter constructor (creates an BinaryHeap with root given parameter).
- A remove method is needed to remove the element which has highest priority.
- An add method is needed to add element to Heap.
- A merge method is needed to merge two Heap.
- A set method is needed to change priority of an element.
- A get method is needed to get data of root node.
- A method is needed to check whether root is leaf or not.

BinarySearchTree

- Three data fields are needed. One Object array is needed to keep elements of tree as array. Two integers to keep size of tree and capacity of array.
- A size method is needed to get size of the tree.
- An add method is needed to add an element to tree.
- Contains and find methods are needed to check whether tree contains an element or not.
- Remove and delete methods are needed to remove an element from tree.
- A toString method is needed to print tree.
- A free and reallocate method is needed to set capacity of array when it is needed.

2. USE CASE AND CLASS DIAGRAMS



3. PROBLEM SOLUTION APPROACH

a. BinarySearchTree

- **Adding an element to binary search tree**

  To add an element to binary search tree, finding correct place is needed. For example, right child of the root must be bigger both root and left child, also root must be bigger than left child. To find the correct place new element must be compared with root first, if it is equal to root, the element should not be added to tree because the element is already in tree.

- **Removing an element from binary search tree**

  To remove an element from binary search tree, there are three situations encountered. First the element has no child: if the element has no child, it is deleted from tree. Second the element has one child: if the element has one child, it is deleted from tree and slides child of it up. Three the element has two children, if the element has two child, the element must be replaced with successor of it.

- **Setting capacity of array**

  To set capacity of array, instead of increasing or decreasing array capacity every element added or deleted, when a depth is filled capacity must be increased. Decreasing capacity is applied when if capacity is bigger than the capacity of tree which number of nodes equals to height.

b. BinaryHeap

- **Adding an element to BinaryHeap**

  To preserve the complete binary tree property, it is needed to know where the next element added. lastElementAdded data field keeps track of parent node of the node where the next element added. When an element is wanted to be added, the left child of the lastElementAdded checked first, if it is full, the right child of it is checked, if it is also filled, the same procedure is applied for the right child of parent of lastElementAdded. If all children are full, the element is added to left child of left child of lastElementAdded node. Left child of lastElementAdded is assigned to lastElementAdded node.

- **Removing the element which has highest priority**

  The element which has highest priority of BinaryHeap is root. To preserve the complete binary tree property, while removing root from the BinaryHeap, it must be changed with last element which is added to BinaryHeap last. And last element must be deleted. After this change, their order must be arranged.

- **Changing the priority of an element**

  Changing the priority of an element may damage the order of elements. So, after changing priority of the element, the element must be compared with parents of it and children of it.

- **Merging two BinaryHeap**

  To merge two BinaryHeap, all elements of one BinaryHeap must be added to other BinaryHeap one by one.

4. RUNNING AND RESULTS

The results of BinarySearchTree and BinaryHeap class are placed to their folders.