# GTU Department of Computer Engineering
# CSE 222/505 - Spring 2021
# Homework 4
# Due date: April 30 2021– 23:55 PM

## PART 1: 40 pts.

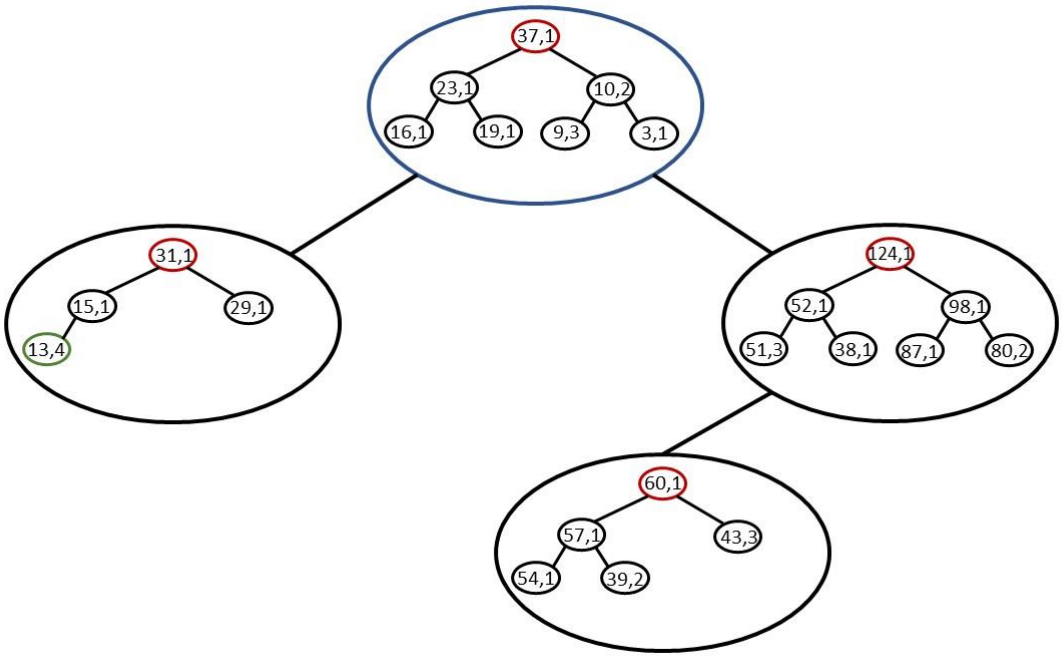
You should implement the following features for the Heap structure.

i. Search for an element
ii. Merge with another heap
iii. Removing $i^{th}$ largest element from the Heap
iv. Extend the Iterator class by adding a method to set the value (value passed as parameter) of the last element returned by the next methods.

## PART 2: 60 pts.

You should implement a BSTHeapTree class that keeps the elements in a Binary Search Tree where the nodes store max-Heap with a maximum depth of 2 (maximum number of elements included in a node is 7).

An example of this class is below.




When implementing the class, you must adhere to the following features:

1. Each node in the heap holds two data: a value (an integer in the example above) and the number of occurrences of the value (how many that number is added to the tree). In the example, the notation 37.1 means that the number 37 is there is only one occurrence of the value 37 is available in the container.
2. Movement on BST is based on values at the root nodes of the Heap. In the example, the root nodes of Heaps are shown in red. If you want to add the number 50 to the tree, the movement will be done as follows:

   Since the Heap with root 37 is full, since 50 is not an element of this Heap and 50 > 37, it moves to the right node in BST.

   Since the Heap with root 124 is full, since 50 is not the element of this Heap and 50 < 124, it moves to the left node in BST.

   Since the Heap with root 60 does not have the number 50 and there is space, it is inserted into the heap at this node as (50.1).
3. If the Heap at a node of BST is full and a new number still needs to be added, a new BST node should be created as the left or the right child of the BST node. After the Heap in the root node of the BST in the example is filled, when a number smaller than 37 is inserted, the left child is created, and the number is inserted into the new Heap in that node. When we examine the example, there are 7 numbers smaller than 37 are inserted into the left child (one 31, one 15, one 29 and four 13's).
4. Remove operation removes only one occurrence of the value. If the number of occurrences becomes zero than the value should be removed.
5. During a remove operation, if the heap at a node becomes empty, the corresponding BST node should be removed as well.
6. The mode is the value in the BSTHeapTree that occurs most frequently. In the example the mode is 13 (node marked in green).

Your class should include the following methods:

- int add (E item) – returns the number of occurrences of the item after insertion
- int remove (E item) – returns the number of occurrences of the item after removal
- int find (E) – returns the number of occurrences of the item in the BSTHeapTree
- find_mode ()

Test your implementation as follows:

1. Insert the 3000 numbers that are randomly generated in the range 0-5000 into the BSTHeapTree. Store these numbers in an array as well. Sort the numbers to find the number occurrences of all the numbers.
2. Search for 100 numbers in the array and 10 numbers not in the array and make sure that the number of occurrences is correct.
3. Find the mode of the BSTHeapTree. Check whether the mode value is correct.
4. Remove 100 numbers in the array and 10 numbers not in the array and make sure that the number of occurrences after removal is correct.

## PART 3: 30 pts.

Analyze the time complexity (in most appropriate asymptotic notation) of all methods in your implementation. Attach the code just before its analysis.

**RESTRICTIONS:**

- Can be only one main class in project
- Don't use any other third part library

**GENERAL RULES:**

- For any question firstly use course news forum in Moodle, and then the contact TA.
- You can submit assignment one day late and will be evaluated over sixty percent (%60).

**TECHNICAL RULES:**

- You must write a driver function that demonstrates all possible actions in your homework. For example, if you are asked to implement an array list and perform an iterative search on the list then, you must at least provide the following in the driver function:
    - Create an array list and add items to the list. Append items to head, tail, and $k^{th}$ index of the list.
    - Perform at least two different searches by using two items in the list and print the index of the items.
    - Perform another search with an item that isn't in the array list and inform the user that the item doesn't exist in the array list.
    - Delete an existing item from the list and repeat the searches.
    - Try to delete an item that is not on the array list and throw an exception for this situation.

  The driver function should run when the code file is executed.

- Implement clean code standards in your code;
    - Classes, methods and variables names must be meaningful and related with the functionality.
    - Your functions and classes must be simple, general, reusable and focus on one topic.
    - Use standard java code name conventions.

**REPORT RULES:**

- Add all javadoc documentations for classes, methods, variables …etc. All explanation must be meaningful and understandable.
- You should submit your homework code, Javadoc and report to Moodle in a "studentid_hw1.tar.gz" file.
- Use the given homework format including selected parts from the table below:

| | |
|---|---|
| Detailed system requirements | X |

| | |
|---|---|
| The Project use case diagrams (extra points) | |
| Class diagrams | X |
| Other diagrams | |
| Problem solutions approach | X |
| Test cases | X |
| Running command and results | X |

**GRADING :**

- **No OOP design:** **-100**
- **No error handling:** **-50**
- **No inheritance:** **-95**
- No javadoc documentation: -50
- No report: -90
- Disobey restrictions: -100
- **Cheating:** **-200**
- Your solution is evaluated over 100 as your performance.

**CONTACT :**


- Teaching Assistant : Başak Karakaş
- bkarakas2018@gtu.edu.tr