**COMP232 – Data Structures & Problem Solving**

**Fall 2020**

**Homework #5**

**Binary Tree Applications**

1. Describe an algorithm for using a Priority Queue to sort a list of integers. You do not have to write any code, just describe the algorithm in sufficient detail to convince me that the code could be written fairly easily.

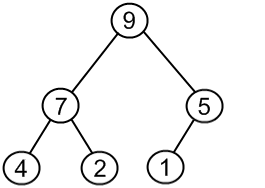
To sort a list of integers using a Priority Queue, we can construct a binary tree based on the list of integers. Each node in the binary tree contains the key-value pair. Then, we can iterate through all of the integers and assign each integer a key indicating the number’s priority where 0 being the highest priority and len(list)-1 being the lowest priority. Finally, we can use those keys as indicators to sort the list.

2. Give and briefly justify the asymptotic bounds for the running time of the algorithm you described in #1, assuming that the backing store for the priority queue is:

a. an unsorted array-based list. 🡪 O(n). We will have to sort the unsorted list of integers to achieve the order of the unsorted array of the backing store, which can cause rearrangement of all the numbers, which can lead to O(n). We can then use a loop that loops through the unsorted array to continuously find the smallest key and add that corresponding value to the new array. The runtime for this process is O(n). Since these two O(n) are independent, the ultimate runtime is O(n).

b. a heap. 🡪 O(nlgn). With a heap, we can iterate through the heap, which has O(n) runtime. We can use the values of the integers as indicators of priority. Then, we need to percolate up/trickle down as appropriate, which can take O(lgn). Together, the total time it takes is O(nlgn).

3. Consider the heap shown below:



Apply each of the following operations, one after the other, to the above max heap. Redraw the heap after each operation has been performed.

A paper with writing on it

Description automatically generated

4. The GenericBox class in the linear.generic package of the COMP232-SampleCode project is a generic class with the type parameter T. How would you change the definition of the GenericBox class such that it can only store types that are numeric (i.e. of type Integer, Float, Short, Byte, Double, Float, etc.). You need only give the first line of the class definition (e.g. public class GenericBox…) as your solution for this question. Hint: These classes share a common super class.

Public class GenericBox<T extends Number> {}

5. The CS232ArrayHeap class is designed to maintain a max heap with respect to the compareTo method implemented by the key type. The compareTo method in the String class places items into alphabetical order, thus strings later in a dictionary would come out of the heap before words earlier in a dictionary. This may be backwards for many applications. This can be addressed by defining a new type for the key that provides a compareTo method that orders the keys appropriately. This is (almost) done in the StringMinHeapKey class in the hw05 package.

a. Run the main method in the StringMinHeapKey class in the hw05 package. What happens? Why? Hint: Look at the constructor being used, the keys being passed in and the compareTo method implementation.

Heap is not valid error. This is because our letters are arranged alphabetically. With A being the root node and B and C being its left and right children respectively, “B”.compareTo(“A”) or “C”.compareTo(“A”) will both return 1 as they come later in the alphabets. This breaks our integrity of the heap as the method understands “B” has higher priority than “A”, which we do not want.

b. Modify the compareTo method so that the given keys form a valid heap with “A” having higher priority than “B”, and “B” higher priority than “C”, etc. Just give the body of your compareTo method as your solution for this exercise.

return o.key.compareTo(this.key)

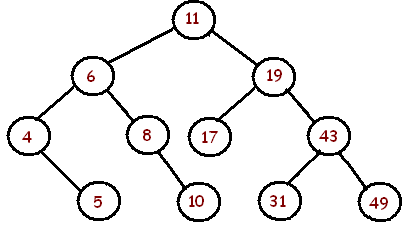
c. Explain why your solution in part b works?

This reverses the result of the compareTo method. Meaning if “A”.compareTo(“B”) = -1, the result for “A”.compareTo(“B”) is now “B”.compareTo("A”), which is 1.

6. Complete the add method in the CS232ArrayHeap class in your hw05 package. The No6Tests class contains tests that you can use to check your implementation of this method.

7. Complete the adjustPriority method in the CS232ArrayHeap class in your hw05 package. The No7Tests class contains tests that you can use to check your implementation of this method.

8. Consider the binary search tree shown below:



Apply each of the following operations, one after another, to the above binary search tree. Redraw the binary search tree after each operation has been performed.

A whiteboard with black text and circles

Description automatically generated

9. Complete the get method in the CS232LinkedBinarySearchTree class in your hw05 package. The No9Tests class contains tests that you can use to check your implementation of this method. Suggestion: Writing a helper method here like we did with get in LinkedBinaryTree will help you with the next two questions.

10. Complete the set method in the CS232LinkedBinarySearchTree class in your hw05 package. The No10Tests class contains tests that you can use to check your implementation of this method.

11. Complete the remove method in the CS232LinkedBinarySearchTree class in your hw05 package. The No11Tests class contains tests that you can use to check your implementation of this method. Suggestion: Handle each of the 3 cases for remove in turn. Implement one, when that passes the appropriate tests; add the next case and so on. Suggestion 2: When you write the 3rd case, add a helper method to find the node with the smallest key in a subtree. Hint: The smallest key will always be in the leftmost node in the subtree.