CSCE 221 Cover Page
Homework Assignment

First Name: Justice Last Name: Ugochukwu UIN:828007313

User Name: justiceu E-mail address: justiceu@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources					
People					
Web pages (provide URL)					
Printed material					
Other Sources					

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name (signature) Justice Ugochukwu Date 11/05/2020 The Programming Assignment Report Instructions
CSCE 221

1. The description of an assignment problem.

   The objective was to implement the Minimum Priority Queue (MPQ) ADT based on three different data structures: vector, linked list, and binary heap.

2. The description of data structures and algorithms used to solve the problem.

   (a) Unsorted MPQ (using vectors):

      i. to insert, we push back data to the end of the vector, but to find and remove minimum, we have to traverse througheach value.

   (b) Sorted MPW (using linked list):

      i. to insert, we check the value across the list, and once we find a value greater than the value we want to insert, we insert our value prior to the one we found, but to find and remove minimum, we deal with the first value in the list.

(c) BinaryHeap: This data structure has the root to be the smallest in the list always, and. preserves that a parent is always smaller than a child. to insert, we insert at the back of the heap, and perform an upheap continuosly. Upheap checks if the child is less than its parent, and switches positions if it is. To find minimum we deal with the root, and to remove, we swap the root and last nodes, delete the last element, then perform down heap until its is greater than all nodes before it. Down heap is switchinh the parent node with its child node until the child is greater than than the parent

3. Show the time complexity analysis for each of the MPQ functions (remove min(), min(), and insert()).

(a)

mpq	insert	min	remove min
sorted	O(n)	O(n)	O(1)
unsorted	O(1)	O(1)	O(n)
binary heap	O(1)	O(logn)	O(logn)

4. Give the best, worst, and average case input examples and runtime (bigO) for each implementation on sorting a given array.

   (a) Sorted:

      i. best case, worst case, and average case is O(1) for min and remove min, because we return the first element in the list.

      ii. best case for insert is O(1) happens either when the value to be inserted is the min, or the list is empty. Worst case is O(n), ahppens when inserted value is the max of the list. Average case, O(n), happens when inserted value is somewhere in the middle of the list.

   (b) unsorted:

      i. best case for min and remove min is O(1), happens when list has size=1. Worst and avergae case is O(n), happens when min is anywhere else in the list, so you have to traverse through to comapre all values.

      ii. insert best, worse, and average case is O(1), her eyou just push to back of the vector.

   (c) Binaryheap:

      i. min best, worst, and average case is O(1), here you return root value.

      ii. Remove best case is O(1), hapens when the heap is the same, so we just switch the values. avergae case is O(logn) happens when we switch root and last element, and the last element was not the max. worst case is O(logn), happens when we switch root and last element, and the last element was the max

      iii. insert is O(1) happens when we insert a value larger than root value, so no upheap. average is O(logn), happens when inserted value is not min, so there is a couple upheaping. Worst case is O(logn), happens when inserted value is min, so max number of upheaping.

5. A user guide description how to navigate your program with the instructions how to:

   (a) compile the program: specify the directory and file names, etc.

   (b) run the program: specify the name of an executable file.

6. Specifications and description of input and output formats and files

   (a) The type of files: keyboard, text files, etc (if applicable).

   there was use of input txt files for testing the code.

   (b) Discuss possible cases when your program could crash because of incorrect input (a wrong file name, strings instead of a number, or such cases when the program expects 10 items to read and it finds only 9.)

   for testing purposes, incorrect input would cause the program to not run, but it would not crash.

7. Provide types of exceptions and their purpose in your program.

   (a) logical exceptions (such as deletion of an item from an empty container, etc.).

   deletion of an item from an empty minimum priority queue would cause a logical error.

   (b) runtime exception (such as division by 0, etc.)

   there are no runtime errors

8. Test your program for correctness using valid, invalid, and random inputs (e.g., insertion of an item at the beginning, at the end, or at a random place into a sorted vector). Include evidence of your testing, such as an output file or screen shots with an input and the corresponding output.