CS 3353: Data Structures and Algorithm Analysis I   Fall 2021

Homework 4: Heaps and Priority Queues
Due Date: 2021/10/25 (Monday) - 11:59 pm

Notes:

- Read Course Information: Section 8 (Miscellaneous) and Section 10 (Academic Dishonesty or Misconduct).
- When you are giving a construction, example, etc., provide a justification with your argument. Your solutions to numerical problems must contain the derivation of your answers. In all of your presentations, strive for correctness, completeness, and clarity. When in doubt about the assumptions of problems, the interpretations of wording, etc., consult the instructor.
- You should strive to complete all problems assigned, and a subset of them will be graded.

1. Read the notes above carefully.
2. **You may need to review the prerequisite materials in discrete mathematics to have sufficient working knowledge, and then do the following exercises.**
3. [Programming Problem]
   Define a new abstract data type *MMHeap*, which includes the operations supporting the abstract data type (data structures) *Heap*: *BuildHeap*, both operations *Min* and *Max*, *Insert*, and two deletion operations *Extract_Min* and *Extract_Max* that delete only keys with minimum and maximum priority values/keys, together with other necessary helper operations such as *Heapify*.

   We can abstract *MMHeap* with modified "almost-complete" binary trees in which the priority ordering alternates level by level; thus, levels $0, 2, 4, \ldots$, are "min" level, and $1, 3, 5, \ldots$, are "max" levels. The priority of a vertex in a min level (max level) is the minimum (maximum, respectively) in its subtree. We call such a tree, an mm-priority tree.

   Implement the *MMHeap* operations with a single array structure - a heap in which the priority ordering alternates level by level, such that they run in $O(\log n)$ time, when the heap has size n.

   Your main program encapsulating the implementations of the necessary packages/subprograms and testing code should incorporate adequate documentation and good programming styles. (When in doubt, please ask.) Also, subprograms should be properly documented with objectives, pre- and post-conditions.

   Develop a simple plan for implementing, testing, and debugging your program. Use bottom-up testing to complement the top-down approach.

   Hand-in
   Since our grader will compile and test run your programs on our departmental machine, you should compile your source programs and test run them on it before your submission.

   Your submission is a zip-file (file-extension is ".zip") that consists of the required source files for compiling your program(s), additional necessary documentation/inputs/remarks for running your programs, and a legible PDF-file for the detailed derivations of the worst-case running time analysis of the recursive multiplication algorithm.

   Submit the zip-file on the campus-wide online learning system Canvas, (which can be accessed through "https://my.oksta
   by 11:59 pm of the due date. Do not use any other type of archive (such as ".tar" or WinRAR). Check the contents of the zip-file before submitting, and check that the submission was successful.