Implementing and Using a Binary Heap

Out: 2/4
Due: 2/17 by 11:59 PM

Learning Objectives

- ⊙ Implementing a Binary Heap Data Structure.
- ⊙ Manipulating an Extensible Array
- ⊙ Using a Binary Heap

In this project you will complete the implementation of a parametric heap data structure.

Definition 1. A **heap** is a complete binary tree in which each entry has a key that is less (greater) than or equal to the key of its parent. When the parent key is greater (less) than or equal to the keys in its sub-trees, it is a *max-heap* (*min-heap*).

The heap property enables the data structure to be used in extracting the item with the highest *priority* value. If a series of insertions are performed on an initially empty heap, the top-most item on the heap is guaranteed to be the item with the highest *priority* value. Deletion always removes the top-most item from a non-empty heap; that is, deletion removes the item with the highest priority. After deletion, the top-most item is replaced with the item with the highest priority of all the items remaining in the heap. We can implement a user-defined comparator so that the entries in the heap are inserted and removed in any desired order whether the underlying implementation of the data structure is a max-heap or min-heap. In this project you will write a parser for a simple heap *language*. Your program will parse any valid program written in the heap *language* and execute all its statements.
I have provided starter code for an extensible parametric*Heap* implementation. You will complete the *Heap* implementation. See the starter code on Moodle for additional details. Do not modify the code except where indicated.

The PQParser program

You will write a program called *PQParser* consisting of three methods (functions): the *height*, *diameter* and *main*. The *height* method/function takes a binary *heap* instance as a parameter and returns the height of the underlying complete binary tree representation of the heap. The *diameter* method/function takes a binary *heap* instance as a parameter and returns the diameter of the underlying complete binary tree representation of the heap.

Definition 2. The **diameter** of a tree is the number of nodes on its longest path. Observe that the longest path in a binary heap (complete tree) goes through the root.

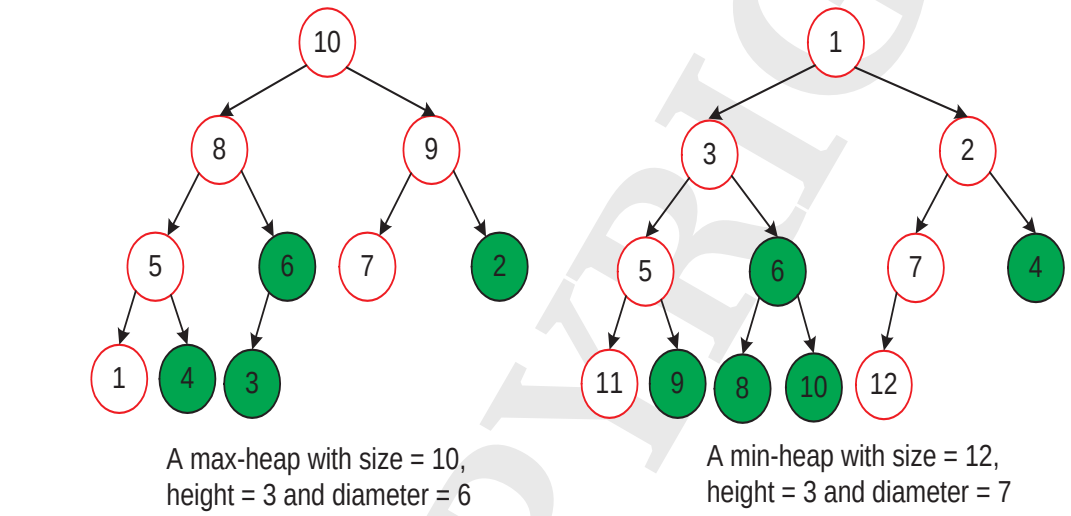

Figure 1: The Size, Height and Diameter of a Heap

Your program will execute a series of statements in a command file from the grammar shown in Listing 1. The program will be called *PQParser*. It will take the name of a *command-file* as a command line argument and execute the instructions in the file while performing a trace of the instructions as they are executed.

The command file consists of instructions in one of these formats:

Listing 1: A Simple Heap ADT Grammar

```
insert < word > | < integer >
delete
peek
stats
```

The *insert* command is followed by an item - word, in the case of a heap of strings, or an integer, in the case of a heap of integers. When your program reads this instruction, it inserts the item in the heap. Whenever an item is inserted, the following message is displayed.

Listing 2: Trace of the insert Command

```
Inserted: item
```

The *delete* command removes the maximum (minimum) item from the max-heap (min-heap). Whenever an item is removed, the following message is displayed, where the ellipses denotes the top-most item in the heap.

Listing 3: Trace of the delete Command

```
Deleted: ...
```

The *peek* command retrieves the maximum (minimum) item from the max-heap (min-heap). Whenever the peek command is executed, the following message is displayed, where the ellipses denotes the top-most item in the heap.

Listing 4: Trace of the delete Command

```
Top: ...
```

The *stats* command displays the size, height and diameter of the heap.

Listing 5: Trace of the stats Command

```
Stats: size = ... height = ... diameter = ...
```

To run the parser:

Listing 6: Running PQParser

```
PQParser <data-type> <order-code> <command-file>

<data-type>: -s or -S for strings
   <order-code>:
      -1 min-heap based on lexicographical order
      1 max-heap based on lexicographical order
      -2 min-heap based on string length
      2 max-heap based string length
<data-type>: -i or -I for integers
   <order-code>
      -1 min-heap based on numerical order
      1 max-heap based on numerical order
<command-file>: name of the command file name
```

Your program throws an exception and terminates if the correct number of command line arguments are not entered: *invalid_argument* for C++ and IllegalArgumentException for Java™programmers. Your program also throws an exception and terminates if an invalid *data-type* or *order-code* flag is provided. Your program should display the usage information shown in Listing 6 prior to throwing an exception. Also, if the *command-file* does not exist, your program should gracefully exit.

I have also provided two command files *strings.hpq* and *integers.hpq* that you may use to test your program. The *strings.hpq* command file performs operations on a heap of strings and *integers.hpq* performs operations on a heap of integers. Be sure to test your program using various order codes. You may create additional command files that include test cases for scenarios not handled in these command files.

Submitting Your Work

1. Complete the preamble documentation in the starter code. Do not delete the GNU GPL v2 license agreement in the documentation, where applicable.

```
/**
 * Describe what the purpose of this file
 * @author Programmer(s)
 * @see list of files that this file references
 * <pre>
 * Course: CS3102.01
 * Programming Project #: 1
 * Instructor: Dr. Duncan
 * Date: 9999-99-99
 * </pre>
 */
```

2. Verify that your code has no syntax error and that it is ISO C++ 11 or JDK 8 compliant prior to uploading it to the drop box on Moodle. Be sure to provide missing documentation, where applicable. Also, add your name after the @author tag when you augment the starter code that I have provided. Put the last date modified in the header comments for each source code file.

3. Locate your source files -

   (a) for Java programmers:
   HeapAPI.java, **Heap.java** and **PQParser.java**

   (b) for C++ programmers:
   Heap.h, **Heap.cpp** and **PQParser.cpp**

   - and enclose them along with the sample command files, *strings.hpq* and *integers.hpq*, in a zip file, YOURPAWSID_proj01.zip, and submit your programming project for grading using the digital drop box on the course Moodle. YOURPAWSID denotes the part of your LSU email address that is left of the @ sign.