CS 340 - Assignment 2
Total : 100pts

1 Written Part [18pts]

Note: pages refer to the 2nd edition of the textbook.

Exercise 1 : *exercise 4.6 page 171* [4pts]

A *full node* is a node with two children. Prove that the number of full nodes plus one is equal to the number of leaves in a nonempty binary tree.

Exercise 2 : *exercise 4.48 page 175* [6pts]

Suppose we want to add the operation `findKth` to our repertoire. The operation `findKth(k)` returns the kth smallest item in the tree. Assume all items are distinct. Explain how to modify the binary search tree to support this operation in $O(\log N)$ average time, without sacrificing the time bounds of any other operation.

Exercise 3 : *exercise 6.8 page 246* [8pts]

Show the following regarding the maximum item in the heap:

1. It must be at one of the leaves.
2. There are exactly $\lceil N/2 \rceil$ leaves.
3. Every leaf must be examined to find it.

Exercise 4 : *exercise 6.10 page 246, question a* [6pts]

Give an algorithm to find all nodes less than some value, X, in a binary heap. Your algorithm should run in $O(K)$, where K is the number of nodes output.

Exercise 5 : *exercise 6.13 page 247, questions a and b* [8pts]

Each deleteMin operation uses $2 \log N$ comparisons in the worst case. N is the number of nodes in the heap structure.

1. Propose a scheme so that the deleteMin operation uses only $\log N + \log \log N + O(1)$ comparisons between elements. This need not imply less data movement.
2. Extend your scheme in the previous question so that only $\log N + \log \log \log N + O(1)$ comparisons are performed.

Programming Part: Min-Max Heap [68pts]

A **min-max heap** is a data structure that supports both `deleteMin` and `deleteMax` in $O(\log N)$ per operation. The structure is identical to a binary heap, but the heap-order property is that for any node, X, at even depth, the element stored at X is smaller than the parent but larger than the grandparent (where this makes sense), and for any node X at odd depth, the element stored at X is larger than the parent but smaller than the grandparent (see figure 1).

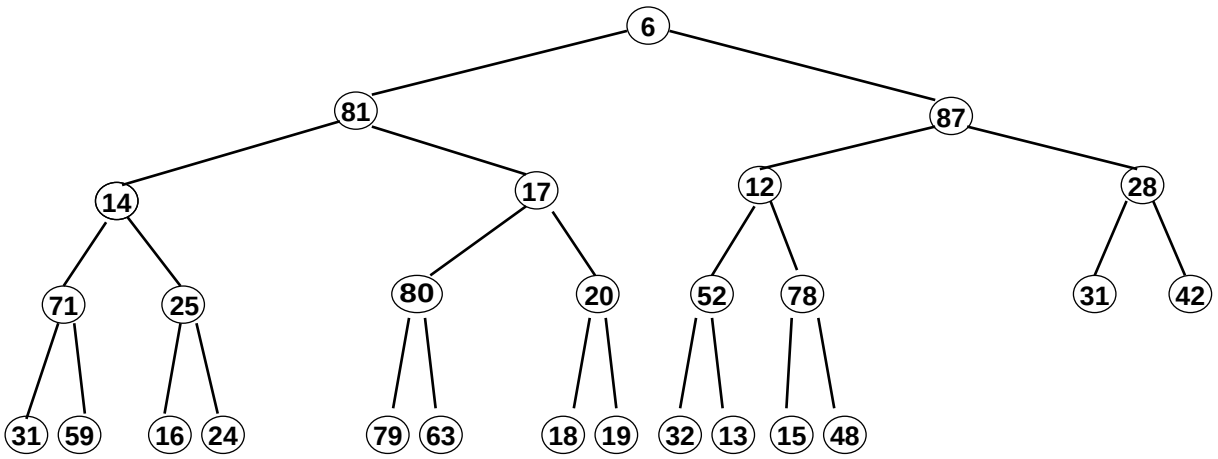

Figure 1: A min-max heap.

Using an array to represent the min-max heap structure (in the same way as for min heap or max heap), implement the following operations.

1. `buildHeap`: Builds a min-max heap from a list of naturals read from standard input.
2. `findMin` and `findMax`: Returns the minimum (resp the maximum) element.
3. `insertHeap`: Inserts a new element into the min-max heap.
4. `deleteMin` and `deleteMax`: Deletes the minimum (resp the maximum) element.

Marking scheme

1. Readability : 10pts
2. Compiling and execution process : 10pts
3. Correctness : 48pts

2 Hand in

2.1 Written Part

- Submit electronic version : pdf(preferred) or word via URCourses. Your file should be named assign2.pdf (or assign2.doc).

2.2 Programming Part

2.2.1 Single file submission for each programming part

Using webct, submit the file containing the C++ code of the programming part **assign2.cpp**. At the top of this file add the following comments :

1. the compiling command you have used
2. an example on how to execute the program
3. other comments describing the program

2.2.2 Multiple files submission for each programming part

Using webct, submit all files required by each programming part :

1. README (file explaining the compilation and execution of your program including the format of input and other details)
2. headers (.h)
3. implementations (.cpp)
4. the Makefile :
   - should be named "**makefile**". In the makefile, the generated executable should be named : "**assign2**"

You can give any name to your source files. The marker will only have to run "**make**" to compile your program and "**assign2**" to execute it.