CPSC 331: Assignment 1

Fall 2023

See the D2L site for due date/time.

1. [20%] Exercise 2-2 on pages 46 of CLRS.

2. [20%] Exercise 3-3, part a, on pages 71–72 of CLRS.

3. [20%] Consider the following recurrence, which is taken from part g of Exercise 4-1 on page 119 of CLRS:

   $$T(n) = 2T(n/4) + \sqrt{n}.$$

   You may assume that $T(n)$ is constant for $n \leq 2$.

   (a) Apply the Master Method to obtain a bound for the recurrence above.

   (b) Suppose in the previous part the Master Method gives you the bound $O(g(n))$. Now use mathematical induction (i.e., Substitution Method) to prove that $T(n) = O(g(n))$.

4. [20%] Implement a *min*-heap class in Java.

   (a) The keys are of type **`int`**.

   (b) Your class shall be named as follows:

   ```
   ca.ucalgary.cpsc331.a1.MinHeap
   ```

   Put your code in a source file **`MinHeap.java`**. Submit only this file.

   (c) Your class shall implement the following Java interface:

   ```
   package ca.ucalgary.cpsc331.a1;

   public interface PriorityQueue {
       boolean empty();
       boolean full();
       void insert(int key);
       int extractMin();
       int min();
   }
   ```

   You do not need to submit the source file for this interface. The autograder will supply it when your source file is compiled.

   (d) Your class shall provide the following constructor:

```
public MinHeap(int N) { ... }
```

The constructor initializes the min-heap to have a maximum capacity of **`N`** (i.e., at most **`N`** elements can be stored in the min-heap). Initially, the min-heap is empty.

(e) Your implementation shall check for underflow (removing or accessing an element when the heap is empty) and overflow (adding an element when the heap is at capacity) of the heap, and throws a **`RuntimeException`** when such situations arise.

(f) The class shall override the **`toString()`** method. Your implementation of **`toString()`** shall return a **`String`** in the following format:

- The string consists of multiple lines, each terminated by a newline character. No line shall contain leading or trailing whitespaces.
- The first line shall look like this:

  ```
  size = 12
  ```

  The size of the heap is the number of cells that are actually occupied. (Of course, your binary heap may have a different size.)
- Subsequent lines list the keys stored in the binary heap, from low indices to high indices. Elements belonging to the same height shall be listed on the same line. Adjacent elements are separated by a single blank space. For example, the following lines list the elements of a min-heap.

  ```
  2
  10 7
  11 20 15 8
  33 12 21 40 16
  ```

Hint: Note that Java array indices start from zero rather than one.

5. [20%] Design an algorithm that takes a max-heap H as input and returns the k largest elements of H in an array $A[1..k]$. Your algorithm shall not modify H. The time complexity shall be dependent on k but not the size/capacity of H. You may safely assume that H contains at least k elements.

   For example, if H contains the numbers 1 to 20, and $k = 3$, then your algorithm shall return the numbers 20, 19, and 18 in an array $A[1..3]$.