Computer Science CSC263H
St. George Campus

January 16, 2020
University of Toronto

Homework Assignment #2

Due: January 30, 2020, by 5:30 pm

- **You must submit your assignment through the Crowdmark system.** You will receive by email an invitation through which you can submit your work. If you havent used Crowdmark before, give yourself plenty of time to figure it out!
- You must submit a **separate** PDF document with for **each** question of the assignment.
- To work with one or two partners, you and your partner(s) must form a **group** on Crowdmark (one submission only per group). We allow groups of up to three students, submissions by groups of more than three students will not be graded.
- The PDF file that you submit for each question must be typeset (**not** handwritten) and clearly legible. To this end, we encourage you to learn and use the LaTeX typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You can use other typesetting systems if you prefer, but handwritten documents are not accepted.
- If this assignment is submitted by a group of two or three students, for each assignment question the PDF file that you submit should contain:
  1. The name(s) of the student(s) who *wrote* the solution to this question, and
  2. The name(s) of the student(s) who *read* this solution to verify its clarity and correctness.
- By virtue of submitting this assignment you (and your partners, if you have any) acknowledge that you are aware of the homework collaboration policy that is stated in the csc263 course web page: `http://www.cs.toronto.edu/~sam/teaching/263/#HomeworkCollaboration`.
- For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbook, by referring to it.
- Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision, and conciseness of your presentation.
- The total length of your pdf submission should be no more than 4.0 pages long in a 10pt font.

Question 1. (1 marks)

Whiz Warehouses Inc. sells goods online, packages them, and then passes them on to a variety of delivery services. Each Whiz warehouse maintains its own set of orders to be processed. Each order has an associated integer amount of cents, chosen by the customer, as a dispatch incentive. Whiz promises that if multiple orders are simultaneously waiting to be processed at a given warehouse, those with higher incentives will be processed before those with lower incentives.

Whiz wants to improve its bottom line, so it shows online customers the maximum incentive offered for orders currently waiting to be processed at the warehouse they are placing their order at. The idea is to nudge customers to out-bid the current maximum incentive.

Occasionally Whiz decides to close a warehouse. In such an event, Whiz trans-ships its goods inventory to another of its warehouses and merges set of orders of the now-closed warehouse with that of the other warehouse.

Give pseudocode for an algorithm for each of the following operations with the given performance guarantees. You may create new data structures by combining elementary data types such as integer with data structures that you have seen in this course. Assume the following operations are for a typical warehouse's set of n orders:

- INSERT: Insert a new order entry, with worst-case $O(\log n)$ time complexity.
- EXTRACT-MAX: extract an order with maximum incentive, with worst-case $O(\log n)$ time complexity.
- MAX-INCENTIVE: Report the maximum incentive currently in the set of orders, with worst-case $O(1)$ time complexity.

Also give pseudocode for the following operation.

- UNION: Merge two different sets with worst-case $O(\log n)$ time complexity, where n is the sum of the number of orders in both sets.

In all cases explain why your solution works correctly, and why it has the required worst-case running times.

Question 2. (1 marks) A binomial heap has 56 entries. What is the degree of the root of the smallest tree in this heap (the degree of a node is the number of children the node has)? After an EXTRACT-MIN is carried out, what is the degree of the root of the smallest tree? Justify your answers.

Question 3. (1 marks)

In the following, B_1 and B_2 are two binary search trees such that every key in B_1 is smaller than every key in B_2.

Describe an algorithm that, given pointers b_1 and b_2 to the roots of B_1 and B_2, merges B_1 and B_2 into a single binary search tree T. Your algorithm should satisfy the following two properties:

1. Its worst–case running time is $O(min\{h_1, h_2\})$, where h_1 and h_2 are the heights of B_1 and B_2.
2. The height of the merged tree T is at most $max\{h_1, h_2\} + 1$.

Note that the heights h_1 and h_2 are ***not*** given to the algorithm (in other words, the algorithm does not "know" the heights of B_1 and B_2). Note also that B_1, B_2 and T are ***not*** required to be balanced.

Describe your algorithm, and justify its correctness and worst-case running time, in ***clear and concise*** English.

HINT: First derive an algorithm that runs in $O(max\{h_1, h_2\})$ time, and then optimize it.

[***The questions below will not be corrected/graded. They are given here as interesting problems that use material that you learned in class.***]

Question 4. (0 marks)

In this question, H denotes a *binomial max heap*, n is the number of items in H, x is (a pointer to the node of) an item inside H, and k is a number (key).

a. Describe a *simple* algorithm to *increase* the key of a given item x in a binomial max heap H to become k. Your algorithm should not change anything if $k \leq x.key$. The worst-case running-time of your algorithm must be $O(\log n)$. Give a high-level description of your algorithm in clear English.

b. Using part (a), describe a *simple* algorithm to *delete* a given item x from a binomial max heap H. The worst-case running-time of your algorithm must be $O(\log n)$. Give a high-level description of your algorithm in clear English.

Question 5. (0 marks) Design a data structure called *SuperHeap* that supports the following operations:

- Insert(k): inserts the key k into the SuperHeap,
- ExtractMax(): removes a max key from the SuperHeap,
- ExtractMin(): removes a min key from the SuperHeap,
- Merge(D,D'): merges SuperHeaps D and D' into one SuperHeap.

The worst-case running-time of each operation in your data structure must be $O(\log n)$ where n is the total number of items. *Your data structure must be based on a data structure that we discussed in this course.*

1. Explain the main high-level idea of your solution in clear English.

   As part of this explanation, (1) state what is the underlying data structure that you are using in your solution, and (2) explain clearly all the information that you are *adding* to this underlying data structure. Specifically, what additional information are you storing in each node?

2. Give a high-level description in clear English of how to implement each operation of the SuperHeap.

HINT: Use binomial heaps and your solution to the preceding question.

Question 6. (0 marks)

This question is about the cost of successively inserting k elements into a binomial heap of size n.

a. Prove that a binomial heap with n elements has exactly $n - \alpha(n)$ edges, where $\alpha(n)$ is the number of 1's in the binary representation of n.

b. Consider the worst-case total cost of successively inserting k new elements into a binomial heap H of size $|H| = n$. In this question, we measure the worst-case cost of inserting a new element into H as the maximum number of pairwise comparisons between the keys of the binomial heap that is required to do this insertion. It is clear that for $k = 1$ (i.e., inserting one element) the worst-case cost is $O(\log n)$. Show that when $k > \log n$, the *average* cost of an insertion, i.e., the worst-case total cost of the k successive insertions divided by k, is bounded above by constant.

Hint: Note that the cost of each one of the k consecutive insertions varies — some can be expensive, other are cheaper. Relate the cost of each insertion, i.e., the number of key comparisons that it requires, with the number of extra edges that it forms in H. Then use part (a).