Computer Science CSC263H
St. George Campus

March 2, 2017
University of Toronto

Homework Assignment #4

Due: March 16, 2017, by 5:30 pm

- You must submit your assignment as a PDF file of a typed (**not** handwritten) document through the MarkUs system by logging in with your CDF account at:

  `markus.teach.cs.toronto.edu/csc263-2017-01`

  To work with one or two partners, you and your partner(s) must form a group on MarkUs.

- If this assignment is submitted by a group of two or three students, the PDF file that you submit should contain for each assignment question:

  1. The name of the student 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.

- The PDF file that you submit must be 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.

- 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.

Question 1. (20 marks) Consider the following "Graph Dismantling Problem". Let $G = (V, E)$ be a connected undirected graph with $n \geq 4$ nodes $V = \{1, 2, \ldots, n\}$ and m edges. Let $e_1, e_2, \ldots, e_m$ be all the edges of G listed in some specific order. Suppose that we ***remove*** the edges from G one at a time, ***in this order***. Initially the graph is connected, and at the end of this process the graph is disconnected. Therefore, there is an edge e_i, $1 \leq i \leq m$, such that just before removing e_i the graph has at least one connected component with more than $\lfloor n/4 \rfloor$ nodes, but after removing e_i every connected component of the graph has at most $\lfloor n/4 \rfloor$ nodes. Give an efficient algorithm that determines this edge e_i. Assume that G is given to the algorithm as a (plain) linked list of the edges appearing in the order $e_1, e_2, \ldots, e_m$.

The worst-case running time of your algorithm ***must be asymptotically better than*** $O(mn)$***. More efficient solutions will get more marks.***

Hint: See "An application of disjoint-set data structures" in pages 562-564 of CLRS (Chapter 21).

Solutions that are unclear, inefficient, or overly complex may not get any points.

Question 2. (20 marks) A manufacturing company producing Product X maintains a display showing how many Product X's they have manufactured so far. Each time a new Product X is manufactured, this display is updated. Interestingly, this company prefers **base 3** notation; instead of using bits (0 and 1) as in base 2 notation, they use **trits** (0, 1 and 2) to represent this number.

The total cost of updating the display is $\$(c + dm)$, where c is a fixed cost to open up the display, d is the cost of changing each display digit, and m is the number of digits that must be changed. For example, when the display is changed from 12201222 to 12202000 (because $12201222 + 1 = 12202000$ in base 3) the actual cost to the company is $\$(c + 4d)$ because 4 digits must be changed. The manufacturing company wants to amortize the cost of maintaining the display over all of the Product X's that are manufactured, charging the same amount to each. In other words, we are interested in the amortized cost of incrementing the display by one.

Let $A(n)$ be the **amortized** cost per display change operation when we perform a sequence of n successive display changes, starting from the number 0. For example, $A(3) = c + \frac{4}{3}d$.

In this question, we want you to give an upper bound on $A(n)$. More precisely, consider the list:

$$c + \frac{4}{3}d, \quad c + \frac{17}{12}d, \quad c + \frac{3}{2}d, \quad c + \frac{7}{4}d, \quad c + 2d, \quad c + \frac{5}{2}d$$

(note that the numbers in this list are sorted from smallest to largest).

What is the *smallest* cost B in the above list such that $A(n) \leq B$ **for all** $n \geq 1$?

Prove the correctness of your answer in two ways: first analyse $A(n)$ using the aggregate method, and then analyse $A(n)$ using the accounting method with an appropriate charging scheme.

Make sure you justify why your answer is the *smallest* such B from the above list.

Question 3. (10 marks) A *multi-set* is like a set where repeated elements matter. For example, $\{1, 8, 3\}$ and $\{3, 3, 8, 1, 8\}$ represent the same *set* but different *multi-sets*. Consider the abstract data type that consists of a multi-set of integers S and supports the following operations:

- INSERT(S, x): insert integer x into multi-set S;
- DIMINISH(S): delete the $\lceil |S|/2 \rceil$ largest integers in S.
  (For example, if $S = \{3, 3, 8, 1, 8\}$, then after DIMINISH(S) is performed, $S = \{3, 1\}$.)

A simple data structure for this ADT, and the corresponding algorithms for each operation, are as follows: The elements of S are stored in an unsorted linked list, and the size of S is stored in a variable n.

- INSERT(S, x): Append x at the end of the list; increment n.
- DIMINISH(S):
  i. $m \leftarrow$ MED(S) find the median of the elements in S
  ii. loop over all elements in S: keep all those $< m$, delete all those $\geq m$
  iii. add as many copies of m as required to have exactly $\lfloor n/2 \rfloor$ elements remaining
  iv. $n \leftarrow \lfloor n/2 \rfloor$

The above uses an algorithm MED that returns the "median" of any list of integers. To simplify this question, we use a slightly non-standard definition of "median": MED returns the $\lceil n/2 \rceil$-th largest integer in the list (including duplicates); for example MED$([3, 2, 1, 1, 3, 3, 3])$ returns 3. Also for this question, assume that MED performs at most $5n$ pairwise comparisons between integers during its execution on any list of length n.

Prove that when we execute an arbitrary sequence of INSERT and DIMINISH operations starting from an empty set S, the *amortized* cost of an operation, *in terms of the number of pairwise comparisons between the elements of the set*, is *constant*. To prove this, use the accounting method to analyse the amortized complexity of the INSERT and DIMINISH algorithms: (a) state clearly what charge you assign to each operation, (b) state and prove an explicit credit invariant, and (c) use your credit invariant to justify that the amortized cost of an operation is constant.