Computer Science CSC263H
St. George Campus

March 10, 2016
University of Toronto

Homework Assignment #5

Due: March 24, 2016, 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.cdf.toronto.edu/csc263-2016-01`. To work with a partner, you and your partner must form a group on MarkUs.
- 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 partner, if you have one) acknowledge that you are aware of the policy on homework collaboration for this course.[a]
- 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.

[a]"In each homework assignment you may collaborate with at most one other student who is currently taking one of the sections of CSC263H taught this term. If you collaborate with another student on an assignment, you and your partner must submit only one copy of your solution, with both of your names. The solution will be graded in the usual way and both partners will receive the same mark. **Collaboration involving more than two students is not allowed. For help with your homework you may consult only the course instructors, teaching assistants, your homework partner (if you have one), your textbook and your class notes. You may not consult any other source.**"

Question 1. (20 marks) Consider the forest implementation of the disjoint-sets abstract data type, with an initial forest of n distinct elements (each one in a one-node tree). Let σ be any sequence of k UNIONs followed by k' FINDs; so *all* UNIONs occur before the FINDs. Prove that the algorithm using Path Compression only (it does *not* use the Weighted-Union rule) executes σ in $O(k + k')$ time, i.e., in time proportional to the length of σ, in the worst-case.

Do *not* make assumptions on k or k' (for example, do not assume that $k = n - 1$ or that $k' \leq k$). As we did in class, assume that the parameters of each UNION are two set representatives, i.e., two tree roots (so there are *no* FINDs "inside" each UNION).

HINT: To compute the "cost" of executing all the FINDs use an amortization-like charging scheme.

Question 2. (36 marks) Consider the following data structure for representing a set I of integers. The elements of the set are stored in a doubly linked list of arrays such that: (1) Each element of I occurs in exactly one of the arrays in this list, (2) each array is sorted in increasing order, (3) the number of elements in each array is a power of 2, (4) no two arrays in the list have the same size, and (5) the arrays in the linked list are kept in order of increasing size. Note that although each array is sorted, there is no particular relationship between the elements in different arrays.

a. (4 marks) Draw two instances of this data structure: one for set $I = \{3, 5, 1, 17, 10\}$ and one for set $I = \{17, 8, 3, 10, 1, 12, 6\}$.

b. (6 marks) To do a SEARCH(x) for an integer x in this data structure, one performs a binary search separately on each array in the list until either x is found in some array, or all arrays have been considered and x is not found.

Give a good upper bound on the worst-case time complexity of this SEARCH algorithm (using the O notation) and justify your answer.

c. (6 marks) To do INSERT(x), i.e, to insert a new integer x into I (assume that x is not already in I), one performs following algorithm:

```
(a) create a new array of size 1 containing x
(b) insert this new array at the beginning of the linked list
(c) while the linked list contains 2 arrays of the same size:
        merge the 2 sorted arrays into one sorted array of twice the size.
        To do each merging use a procedure similar to the one used in Mergesort.
```

Give a good upper bound on the worst-case time complexity of this INSERT algorithm (using the O notation) and justify your answer.

d. (12 marks) Suppose we execute a sequence of n INSERTs starting from an empty set I. Determine a good upper bound on the *amortized* time of an INSERT (i.e., the worst-case total time to execute these n INSERTs divided by n). Justify your answer in two different ways, i.e., give two separate proofs, each proof using a different argument (e.g., use aggregate analysis and the accounting method).

e. (8 marks) Describe an algorithm to perform a DELETE(x) operation (i.e., given a pointer to integer x in one of the arrays of the list, remove x) in $O(n)$ time in the worst case. There is no need to use pseudocode, a clear and concise description of the algorithm in English is sufficient. Explain why the worst-case time complexity of your algorithm is $O(n)$.