Computer Science 263 | November 5, 2007
St. George Campus | University of Toronto

Homework Assignment #3
Due: November 20, 2007, by 10:10 am

1. *Please complete and attach (with a staple) an assignment cover page to the front of your assignment. You may work alone or with one other student. If you work in a group, write both your names on the cover sheet and submit only one copy of your homework.*
2. *If you do not know the answer to a question, and you write "I (We) do not know the answer to this question", you will receive 20% of the marks of that question. If you just leave a question blank with no such statement, you get 0 marks for that question.*
3. *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 abstract data type that consists of a set $S$ of positive integers upon which the following two operations can be performed:

- DELETE$(S, i)$: Delete integer $i$ from the set $S$. If $i \notin S$, there is no effect.
- PREDECESSOR$(S, i)$: Return the predecessor in $S$ of integer $i$, i.e. $\max\{j \in S \mid j < i\}$. If $i$ has no predecessor in $S$, i.e. if $i \leq \min S$, then return 0. Note that it is not necessary for $i$ to be in $S$.

Initially, $S$ is the set of $n$ consecutive integers 1 through $n$.

**a.** Describe a data structure for this ADT with $O(\log^*(n))$ amortized cost per operation (for any sequence of operations). Justify the correctness of your data structure and the time complexity of its operations. You may wish to illustrate your data structure on some data of your choosing.

**b.** Explain how you initialize your data structure and justify how much time it takes.

**Question 2.** (20 marks) This question relates to implementing an open addressing hash table (*i.e.,* using probing) using a dynamic array. The hash table will be a dynamic array that doubles whenever it becomes 3/4 (or more) full, and halves whenever it becomes 1/4 (or less) full. More precisely, when the array grows, we have to create a new array of twice the size, go through every slot in the old array and, whenever we find a nonempty slot, rehash the item into the new array. We handle the shrinking case similarly.

Assume that the array starts out empty. The hashing will be done by some arbitrary hash function with some arbitrary type of probing. Throughout the question, we will measure the cost of each INSERT and DELETE by the number of array slots that we need to access (read or write).

**a.** The course textbook shows that the expected number of slots accessed (*i.e.,* the number of probes needed) to INSERT or DELETE an item in a hash table with $n$ elements and $m$ slots is $\frac{1}{1-\alpha}$, where $\alpha = \frac{n}{m}$ is the load factor. In the scheme described above, what is $\alpha_{\max}$, the largest that the load factor can attain?

For simplicity, assume for the rest of this question that every INSERT or DELETE operation requires *exactly* $\frac{1}{1-\alpha_{\max}}$ probes. What is the numeric value of $\frac{1}{1-\alpha_{\max}}$?

**b.** Let $m$ be the current size of the array. What is the cost of doing an INSERT in the case where the array needs to double? What is the cost of doing a DELETE in the case where the array needs to halve? Briefly explain your answers.

**c.** Using the accounting method, give an accounting scheme that shows a good bound on the amortized cost of on the cost of insert and delete operations on this hash table so that the actual cost is always covered. More precisely, specify how much to charge for an INSERT operation and a DELETE operation (these amounts may be different), state what the credit invariant is, and briefly justify that the credit invariant is maintained.

**Question 3.** (5 marks) An undirected multigraph is a generalization of an undirected graph in that the edges form a multiset: for any pair of vertices $u$ and $v$, there may be zero, one, or more edges joining $u$ and $v$. The number of edges between $u$ and $v$ is the *multiplicity* of $(u, v)$.

**a.** (3 marks) Describe how to represent a multigraph using each of these approaches: an adjacency lists representation, and an adjacency matrix representation. Be precise. What is the space usage for each approach, in terms of the number of vertices $n$ and the number of edges $m$?

**b.** (2 marks) Suppose each edge in an undirected multigraph has a weight associated with it (edges between the same vertices may each have *different* weights). Describe an efficient way to represent such a weighted multigraph using adjacency lists. What is the complexity of answering the question "is there an edge between $u$ and $v$ with weight $x$?"

**Question 4.** (25 marks) This question is a programming assignment. To see its description follow the link given in the "Assignments" section of the course web page.