Computer Science B63 — March 8, 2007
Scarborough Campus — University of Toronto

Homework Assignment #4
Due: March 21, 2007, by 1:10 pm

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.*

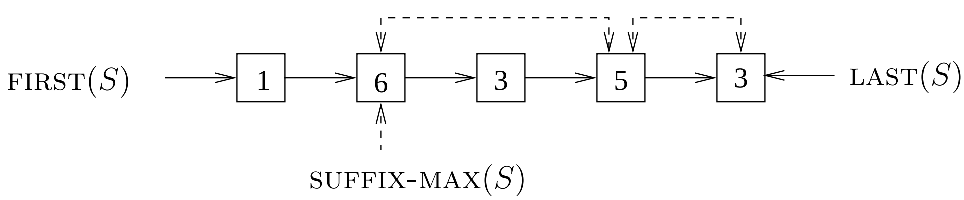
3. *Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision and conciseness of your presentation.*

Question 1. (25 marks) Consider the abstract data type MAXQUEUE that maintains a sequence S of integers and supports the following three operations:

- DEQUEUE(S): removes the first element of S and returns its value.
- ENQUEUE(S, x): appends the integer x to the end of S.
- MAXIMUM(S): returns the largest integer in S, but does not delete it.

An element x is a *suffix maximum* in a sequence if all elements that occur after x in the sequence are strictly smaller in value. For example, in the sequence 1,6,3,5,3, the suffix maxima are the second, fourth, and fifth elements.

One way to implement the MAXQUEUE abstract data type is to use a singly linked list to represent the sequence S, with additional pointers FIRST(S) and LAST(S) to the first and last elements of that list; and to have a doubly linked list of the suffix maxima (arranged in the same order as they are in the sequence), with an additional pointer SUFFIX-MAX(S) to the first element of that list. For example, the sequence $S = 1, 6, 3, 5, 3$ would be represented as follows:



a. (15 marks) Describe algorithms to implement each of the three operations for the MAXQUEUE abstract data type using this representation. Your algorithms should be such that the operations have amortised complexity $O(1)$, assuming that initially the MAXQUEUE contains the empty sequence. Prove that your implementation achieves this amortised complexity.

b. (10 marks) Determine an asymptotic tight bound for the worst-case cost of an individual operation in a sequence of m operations. Justify your bound.

c. (no marks) [*This question will not be corrected/graded. It is given only as an interesting problem you may want to think about.*] Find an alternative implementation of this abstract data type such that the worst-case time needed for an individual operation is $O(\log n)$, where n is the number of elements in the sequence. The amortised complexity in this case need not be $O(1)$.

Question 2. (15 marks) Solve the *Film Star Problem*. You walk into a bar in Toronto during the International Film Festival, and you want to determine whether there is a film star among the $n \geq 2$ customers that are at this bar. By definition, a customer is a *film star* if *both*:

1. every customer at the bar knows the film star, and

2. of course, the film star does *not* know anyone in the bar (except him/herself).

To determine whether there is a film star (and if so to determine who it is), you can choose any customer in the bar and ask him (her) "*excuse me, Sir (Madam), do you know this person over there?*" while you are pointing at another bar customer. All you get is a dry yes/no answer. You can ask several such questions. Of course, you can assume that every customer knows him/herself.

Since you are a CSCB63 alumnus, to solve this problem you decide to model the above situation as a directed graph $G = (V, E)$, where the set of nodes $V = \{1, 2, \ldots, n\}$ represents the n customers, and the set of (directed) edges E is such that $(i, j) \in E$ iff customer i knows customer j. A film star is any node $x \in V$ such that: (i) for all j, $(j, x) \in E$, and (ii) for all $j \neq x$, $(x, j) \notin E$. You assume that you are given the adjacency matrix A of G: with this assumption, asking customer i whether he/she knows customer j (which takes constant time) corresponds to checking the value of $A(i, j)$ (which also takes constant time).

a. (12 marks) Give an *efficient* algorithm to determine whether there is a film star in the bar, and if so, to find which customer it is. Your algorithm should use the above matrix representation of the problem. Describe your algorithm using pseudo-code and prove it correct.

To obtain full marks, your algorithm should be minimum in the "number of questions that it asks" (i.e., the number of accesses the adjacency matrix A) in the worst-case.

b. (3 marks) How many questions does your algorithm asks (i.e., how many times it accesses the adjacency matrix A) in the worst-case? Give the *exact* number (i.e., do not use the asymptotic notation) and justify your answer.

Question 3. (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.