CSci 5421: Practice Questions for the Final

Note: These questions are on material covered after Midterm 2. The syllabus for the Final includes all topics covered in the course.

1. Let \( Q \) be a queue, with the usual operations \text{ENQUEUE} and \text{DEQUEUE}. We wish to implement \( Q \) using two stacks \( S_1 \) and \( S_2 \), so that the total cost of any sequence of \( n \) queue operations is \( O(n) \). (Assume that \( Q \) is empty initially.) Describe in words how this can be done and give pseudocode for \text{ENQUEUE} and \text{DEQUEUE}. Analyse your solution using the potential method of amortized analysis.

You may use the stack operations \text{Push} and \text{Pop} as black boxes, without writing code for them.

2. Let \( S \) be a stack subject to an arbitrary sequence of \text{Push} and \text{MultiPop} operations. The size of \( S \) never exceeds \( k \) (\( k \) need not be a constant). After every \( k \) of these operations, the items currently on \( S \) are copied to back-up storage using a \text{Copy} operation, at a cost of \( k \). (\( S \) is not modified in the process.) Use the accounting (credits) method to prove that any sequence of \( n \) operations, consisting of \text{Push}, \text{MultiPop}, and \text{Copy} operations, on an initially-empty stack, takes \( O(n) \) time. (Wlog, assume that a \text{Push} on a full stack leaves it unchanged and costs 0.)

Hint: For purposes of assigning credits, you may find it useful to distinguish between the slots in \( S \) and the items occupying slots in \( S \). Consider storing credits with slots and/or items suitably.

3. Let \( T \) be a dynamic table that is subjected to insertions only. \( T \) is managed as discussed in class; that is, \( T \) is expanded to twice its size when it is full and all the elements from the old table are moved over into the new one. Prove, via the accounting (credits) method, that the amortized cost of a \text{TABLE-INSERT} operation is a constant. (Assume that the table is empty initially.)

State clearly the invariant that you use and the number of credits assigned to each \text{TABLE-INSERT}. Your approach must store any excess credits with specific items in the table, not with the operations.

4. Do a \text{DECREASE-KEY} operation on the Fibonacci Heap below, by decreasing key 22 to 18. Show intermediate steps and marked nodes clearly. (Marked nodes are indicated below by ‘*’.)

```
              h
               /|
              / |\*2
             / *9 |
*9 / 7 10/ \
 24 / 11 15 *14 20 12 25*
   / / / / 16 *19
  / / / / 22 *25
```

5. An arbitrary sequence of Fibonacci heap operations is executed on an initially-empty Fibonacci heap. Recall that each \text{DECREASE-KEY} operation generates a sequence of zero or more calls to
Casc-Cut. The last call (if any) in the sequence is said to be a last call; all other calls in the sequence are said to be non-last calls. (For simplicity, assume that there are no DELETE operations in the sequence; thus any Casc-Cut is due to a DECREASE-KEY operation only.)

Argue carefully that the total number of non-last calls to Casc-Cut, taken over all the DECREASE-KEY operations, is at most the number of DECREASE-KEY operations. Hence conclude that the total number of calls to Casc-Cut (last and non-last) is at most twice the number of DECREASE-KEY operations.

Hint: Associate each marked node in the heap with a suitable DECREASE-KEY operation in the past.

6. Recall the 2-approximation algorithm for the minimum weight vertex cover problem. Suppose that we change the condition for including a vertex \( v \) in the cover set \( C \) to (say) \( \overline{x}(v) \geq 3/4 \) (line 4 of the algorithm on page 1126). Verify that the analysis in Theorem 35.7 then yields an approximation ratio of 4/3. Then explain what is wrong with the proposed approach, i.e., why this approach does not really yield a 4/3-approximation algorithm for the problem.

7. Problem 7 in Homework 5. (This problem was “unassigned”.)

8. Recall that a boolean expression is in 3-CNF form if it is the conjunction (logical “AND”) of clauses, where each clause is the disjunction (logical “OR”) of exactly three literals (i.e., a variable or its negation). In class, we discussed the problem of maximizing the number of satisfiable clauses in a boolean expression in 3-CNF form and designed a randomized, polynomial-time \((7/8)\)-approximation algorithm. (See pages 1123–1124\footnote{Note that, as discussed in class, our convention for stating the approximation ratio is slightly different from that in the book, which is why we have a ratio of 7/8, whereas the book has a ratio of 8/7.}. Suppose now that we no longer restrict our CNF expression to have exactly three literals per clause; so a clause can now have one or more literals and different clauses can have different numbers of literals. Give a randomized, polynomial-time 2-approximation algorithm for this problem. Be sure to establish the approximation ratio.

9. Let \( T \) be a tree with \( n \) vertices. Describe a greedy algorithm to find a minimum-size vertex cover of \( T \) in \( O(n) \) time. Prove that your algorithm is correct.

Hint: A leaf in \( T \) is a node with one incident edge. (Any tree with two or more vertices has at least two leaves.) Which of the two endpoints of the edge would you pick (greedily) to include in the cover?