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 $T$ 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.

3. When analysing the space bound of the limited node-copying method in class, we used the following potential function: “$\Phi =$ number of full nodes in the current tree”. Why did we not use the potential function “$\Phi' =$ number of full nodes in all trees (past and current)”, i.e., why is the “current tree” important?

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
   /|
  3 *2
 / |
*9 /|\     /
/ | 7 *10
/ \ 5 11 24
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 $\text{Casc-Cut}$. 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 $\text{Delete}$ operations in the sequence; thus any $\text{Casc-Cut}$ is due to a $\text{Decrease-Key}$ operation only.) Argue carefully that the total number of non-last calls to $\text{Casc-Cut}$, taken over all the $\text{Decrease-Key}$ operations, is at most the number of $\text{Decrease-Key}$ operations. Hence conclude that the
total number of calls to \textsc{Casc-Cut} (last and non-last) is at most twice the number of \textsc{Decrease-Key} operations.

\textit{Hint:} Associate each marked node in the heap with a suitable \textsc{Decrease-Key} operation in the past.

\textbf{6.} Let \( S = \{ R_i = [\ell_i, r_i] \times [b_i, \infty), 1 \leq i \leq n \} \) be a set of rectangles in the plane. That is, \( R_i \) has sides parallel to the coordinate axes, has \([\ell_i, b_i]\) as its lower left corner, \([r_i, b_i]\) as its lower right corner, and extends to infinity in the positive \( y \)-direction. We would like to preprocess \( S \) into a suitable data structure so that given any query point \( q = (q_x, q_y) \) we can report efficiently all the rectangles in \( S \) that contain \( q \). (Note that the query is not known ahead of time. Also, the user can issue a large number of different queries on \( S \), so it makes sense to preprocess \( S \) so as to make the queries efficient.)

The goal is to solve this problem using a \textit{persistent} red-black tree and the “persistence + sweep” approach. The desired bounds are \( O(n) \) space, \( O(n \log n) \) preprocessing time, and \( O(\log n + K) \) query time, where \( K \) is the number of rectangles reported. Note that, unlike problem 2, this is a query problem.

Do the following:

(i) Show how to build a data structure for this problem that is based on a persistent red-black tree. Describe briefly the main ideas and give pseudocode.

(ii) Show how to query the structure. Describe briefly the main ideas, give pseudocode, and argue briefly why the query algorithm works.

(iii) Analyse \textit{carefully} the space, preprocessing time, and query time of your solution.

\textit{Note:} Assume that you have available persistent counterparts to the routines for standard (i.e., non-persistent) red-black trees. Specifically, there are available routines \texttt{Insert(tree,key,time)}, \texttt{Delete(tree,key,time)}, \texttt{Search(tree,key,time)} to insert, delete, or search a key in a persistent red-black tree at some time instant, as well as a routine \texttt{Access-Range(tree,range,time)} to retrieve from a persistent red-black tree the \( K \) keys that lie in some range (i.e., interval) at some time instant in \( O(K + \log n) \) time. Assume that the “limited node-copying” method is used; you may treat this as a black-box and make calls to the \texttt{Insert}, \texttt{Delete}, \texttt{Search}, and \texttt{Access-Range} operations, as appropriate. Also assume, for simplicity, that all \( x \)- and \( y \)-coordinates in the input are distinct.

\textbf{7.} 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) “\( \bar{x}(v) \geq 3/4 \)” (line 4 of the algorithm on page 1126). The analysis in Theorem 35.7 then yields an approximation ratio of \( 4/3 \). (You should verify this.) 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.

\textbf{8.} Problem 4 in Homework 5.

\textbf{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.

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