CSci 5421: Practice Questions for Final

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

1. Redo question 6 in Homework 5 (on building a binary heap) via the accounting (credits) method. State clearly your invariant and the amortized cost (i.e., number of credits) assigned to each HEAPIFY operation.

2. Consider an implementation of a binary min-heap as a binary tree (See Chapter 6). It is known that INSERT and EXTRACT-MIN each take time $O(\log n)$ in the worst case, where $n$ is the size of the heap. It is possible to use amortized analysis to derive a more informative bound, as requested in part (a) below.

(a) Use the potential method to prove the following: If an arbitrary sequence of $n$ operations, consisting of INSERT and EXTRACT-MIN, is done on an initially-empty heap, then the amortized cost of INSERT is $O(\log n)$ and that of EXTRACT-MIN is $O(1)$. Do not change these operations in any way. Describe your potential function carefully and show that it works.

(b) Is it possible to achieve an amortized cost of $O(\log n)$ for EXTRACT-MIN and $O(1)$ for INSERT? Justify your answer.

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 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 TABLE-INSERT. Your approach must store any excess credits with specific items in the table, not with the operations. This problem was discussed in class. For maximum learning effectiveness, try to do it yourself without looking at your notes.

4. Problem 17-2, p. 473, parts (a) and (b). It is enough to describe the search and insertion algorithms in words. In part (b), to analyze the amortized cost for insertion assume that you start with an empty structure and do $n$ insertions into it.

Use the accounting (i.e., credits) method for your analysis. State clearly the invariant that you use and the number of credits assigned to each operation.

5. Consider the problem of incrementing an infinite binary counter, but assume that instead of starting at a value of zero, the counter starts with $b > 0$ 1’s, for some integer $b$. (The 1’s can be anywhere.) Use the potential method to argue that the total cost of a sequence of $n$ INCREMENT operations is $2n + b$. Conclude that the total cost of the sequence is $O(n)$ if $n = \Omega(b)$.

Hint: Compute $\hat{c}_i$ using a suitable potential function. Then rewrite the total actual cost in terms of the total amortized cost and $\Phi_0 - \Phi_n$, and upper bound the last term suitably. (The non-negativity condition does not hold, hence we have to rewrite the relationship and introduce $b$ into the final bound.)

This problem illustrates two things: (1) How the potential method can still be used to handle an initial condition that is not necessarily “nice”; and (2) Sufficiently many operations have to be done to overcome the burden placed by the non-zero number of 1’s in the initial counter.

This problem is discussed (for a finite counter) in the text. Again, for maximum learning effectiveness, try to do it yourself without looking at the text.

Over $\implies$
6. Do a \texttt{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 ‘*’.)

\begin{center}
\begin{tikzpicture}
  \node (h) at (0,0) {h};
  \node (3) at (-1.5,-1) {3};
  \node (2) at (-1,-1.5) {*2};
  \node (1) at (-1,-2) {1};
  \node (9) at (-2.5,-2) {*9};
  \node (5) at (-2,-3) {5};
  \node (7) at (-1.5,-3) {7};
  \node (10) at (-1,-3) {*10};
  \node (11) at (-3.5,-3) {11};
  \node (24) at (-3,-3.5) {24};
  \node (15) at (-2.5,-4) {15};
  \node (14) at (-2,-4) {*14};
  \node (20) at (-1.5,-4) {20};
  \node (12) at (-1,-4) {12};
  \node (25) at (-2,-4.5) {*25};
  \node (16) at (-3.5,-4.5) {16};
  \node (19) at (-3,-5) {*19};
  \node (22) at (-2.5,-5) {*22};
  \node (19) at (-2,-5.5) {*25};

\end{tikzpicture}
\end{center}

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

Argue carefully that the total number of non-last calls to \texttt{CASC-CUT}, taken over all the \texttt{DECREASE-KEY} operations, is at most the number of \texttt{DECREASE-KEY} operations. Hence conclude that the total number of calls to \texttt{CASC-CUT} (last and non-last) is at most twice the number of \texttt{DECREASE-KEY} operations.

\textit{Hint:} Associate each marked node in the heap with a suitable \texttt{DECREASE-KEY} operation in the past.

8. Problem 19-1, p. 526–527. (Use the potential function given on page 509.)