Exercises #3

Greg Plaxton

February 21, 2013

These exercises are intended to reinforce the lecture material. Sample solutions will be handed out on March 5th. It is recommended that you consider the problems on your own before reading the sample solutions.

1. Exercise 14.3–7, page 354 (second edition, page 317).

2. Problem 14–1, page 354 (second edition, page 318).

3. Problem 20–1(b), page 557 (does not appear in second edition).

4. Problem 20–2, parts (a) and (b), page 558 (does not appear in second edition). Before attempting to solve this problem, please read the following comments, which address what appear to be flaws in the problem statement.

   First, perfect hashing as described in Section 11.5 does not seem to be enough for this problem since we are maintaining a dynamic set. Dynamic perfect hashing schemes are known but it seems to me that the time bounds for INSERT and DELETE claimed in parts (b) and (g) should still have qualifications associated with them; specifically, these should be expected time bounds (as opposed to worst-case) that hold only in an amortized sense. In order to set aside issues related to the performance of the hash table component of the data structure, please assume throughout that we are given an "idealized hash table" that allows insertion, deletion, and find operations to be performed in worst-case constant time, while using only linear space (with respect to the number of items currently in the table).

   Second, I believe that the preliminary data structure described in the question is not adequate to achieve all of the time bounds claimed in part (b). One way to repair the data structure is to maintain some additional auxiliary data along with each entry in the hash table. Specifically, along with a given prefix α, it is convenient to keep pointers to the minimum and maximum elements in the set that have prefix α. (Including such maximum and minimum pointers for every single prefix is actually overkill. See the definition of x-fast tries, e.g., on Wikipedia, to understand why it is enough to maintain only a subset of these pointers.) Doing so, we need to show how to maintain these minimum and maximum pointers while achieving the stated time bounds for the various operations.

(a) Here you should give a worst-case O-bound in terms of the two parameters n and u that is tight for n sufficiently smaller than u (e.g., $n \leq \sqrt{u}$). Include a brief justification of your claimed bound.

(b) For this part, just describe how to do PREDECESSOR, SUCCESSOR, INSERT, and DELETE in $O(\lg u)$ time. The question asks how to perform the MEMBER operation in $O(\lg \lg u)$ time, but this operation can be performed in $O(1)$ time.

5. Problem 20–2, parts (e), (f), and (h), page 558 (does not appear in second edition). As in the previous question, which deals with two earlier parts of the same problem, we will assume an idealized hash table. Furthermore, I believe that the clause "The perfect hash table stores only the representatives", which should now be interpreted as "The idealized hash table stores only the representatives" is misleading. Please read this instead as "The idealized hash table stores every prefix of every representative, as in the preliminary data structure of parts (a) and (b)".

6. Problem 21–3, parts (a), (b), and (c), page 584 (second edition, page 521).

7. Recall that Section 21.4 of the text establishes an $O(m\alpha(n))$ time bound for the union-find problem, where $\alpha(n) = \min\{k : A_k(1) \geq n\}$. Let $\beta(n) = \min\{k : A_k(1) \geq \lg n\}$. The result of part (a) below implies that the complexity of union-find is $O(m\beta(n))$. Let $\gamma(m,n) = \min\{k : A_k(\lceil m/n \rceil) \geq \lg n\}$. In parts (b) and (c) we establish a stronger $O(m\gamma(m,n))$ union-find bound under the assumption that $\gamma(m,n) > 0$. Remark: If $\gamma(m,n) = 0$ then an $O(m)$ union-find bound is easy to establish using similar arguments.

   (a) Prove that $\alpha(n) = O(\beta(n))$.

   (b) Let $k = \gamma(m,n) - 1$ and let $f(\ell) = A_k^{(\ell)}(1)$. Prove that $f(2\lceil m/n \rceil) \geq \lg n$. (Hint: First prove that $f(\lceil m/n \rceil - 1) \geq \lceil m/n \rceil$.)

   (c) In order to establish an $O(m\gamma(m,n))$ union-find bound, we modify the argument of Section 21.4 as follows. First, we modify the definition of *level*(x) so that it never exceeds $\gamma(m,n) - 1$ (i.e., the "new" value of *level*(x) is the minimum of the "old" value of *level*(x) and $\gamma(m,n) - 1$). We then define a node to be *special* if *level*(x) (under the new definition) is $\gamma(m,n) - 1$. We then split the task of bounding the total length of all find paths into two parts: (1) bounding the total number of non-special nodes on all find paths; (2) bounding the total number of special nodes on all find paths. For (1), we claim that the argument of the text can be used, simply replacing $\alpha(n)$ everywhere by $\gamma(m,n)$. The goal of this question is to establish bound (2). Use the result of the previous part, along with reasoning similar to that used in the proof of Lemma 21.13 (second edition, Lemma 21.12), to establish (2).