The University of British Columbia

Department of Computer Science

Computer Science 420—Advanced Algorithm Design and Analysis

Homework Assignment 5

Due: 2015 February 26

Please review the class policy on collaboration before starting this assignment. You are free to discuss problems in groups of size at most three. However, your actual homework submission must be prepared on your own. At the top of the first page of every homework submission you must clearly acknowledge all sources (including books, websites and discussions with fellow students) that you have used in the preparation of your submission.

1) [This question is based, in part, on Problem 17-2 from the Cormen et al. text ; you may find it helpful to review material on amortized analysis, e.g. Chapter 17 of Cormen et al.]

   You are all familiar with the fact that binary search takes $O(lg\,|S|)$ time, in the worst case, to perform MEMBER queries on a set S represented as a sorted array. Unfortunately, the time for INSERT operations on a sorted array is linear in the size of the array, in the worst case. The cost of INSERT operations can be reduced, at least in the *amortized* sense, by maintaining S as an array only parts of which (called *blocks*) are sorted.

   Specifically, suppose that at any moment in time set S contains n elements. Let $n = \sum_{j=0}^{k} b_j 2^j$, where $k = \lfloor lg\, n \rfloor$ (i.e. $\langle b_k \ldots b_1 b_0 \rangle$ is the binary representation of n.) The set S is represented as something that we will call a *block-sorted array* (or B-S array for short) of size $2^{k+1} - 1$. A B-S array $A[1 : 2^{k+1} - 1]$ is divided into $k + 1$ *blocks* where the i-th block, $0 \leq i \leq k$ is the subarray $A[2^i : 2^{i+1} - 1]$. The invariant maintained by the B-S array representation is that (i) blocks whose index i corresponds to a 0-bit in the binary representation of n contain a special sentinal ∞ only; (ii) all other blocks contain keys of S, without duplication; and (iii) the elements in each block are sorted in increasing order. Note that no assumption is made about the relationship between elements in different blocks.

   a) Describe (using pseudo-code) an algorithm to perform a MEMBER operation on a B-S array. Your algorithm should try to minimize the average (over all possible queries from S) of the cost successful MEMBER queries.

   b) Analyse the worst-case and average-case running times for successful MEMBER queries using your algorithm. [Assume, for the sake of simplicity, that the cost of performing a binary search on a sub-array of size 2^i is i.]

   c) Describe how to implement an INSERT operation on a B-S array. [Hint: here you will find it helpful to review the amortized analysis of maintaining a binary counter, e.g. Cormen et al., Chapter 17.]

   d) What is the worst-case running time of INSERT? What is the amortized running time of an INSERT (in a long sequence of INSERT operations)? Explain your analysis.

   e) Suppose that you know that the most recently inserted elements are the most likely to be the subject of MEMBER queries. How would you implement MEMBER queries to take advantage of this? (Hint: where do the most recently inserted elements reside?)

   f) What is the worst-case cost of a MEMBER query (using your implementation) for a B-S array with n keys, expressed as a function of i and n, when the query element x happens to be the i-th most recently inserted element?

g) Describe how to modify a B-S array (keeping the same asymptotic cost for INSERT operations) so that the worst-case cost of MEMBER is independent of n (i.e. it depends only on i), when the query element x happens to be the i-th most recently inserted element? (Hint: keep more than one block of the same size in the B-S array.)

2) A connected, acyclic undirected graph is called a *(free) tree*.

a) Describe an algorithm that determines whether or not a given graph $G = (V, E)$, represented as a adjacency-list, is a tree. Your algorithm should run in $O(V)$ time even on dense graphs.

b) Prove that any algorithm that tests whether or not a given graph is a tree, using an adjacency matrix representation, must use $\Omega(V^2)$ time (in fact, must look at $\Omega(V^2)$ matrix entries) in the worst case.

3) Imagine that you are given a directed graph $G = (V, E)$ each of whose edges is coloured by one of χ distinct colours. (Think of the edges of a fixed colour as forming a sub-network.)

a) Design, describe and analyse the most efficient algorithm that you can think of to find a path from a specified vertex s to another specified vertex t that minimizes the number of colour changes (that is the number of transitions between sub-networks). We will refer to this as the *minimum colour-transition path problem*. (Here you can imagine that you have to pay a unit cost when transferring from one sub-network to another).

b) Design, describe and analyse the most efficient algorithm that you can think of to solve the related *minimum colour path problem*, which asks for the s, t-path that uses the smallest total number of distinct edge colours. (Here there is a unit cost for every sub-network used by a path, independent of the number of transfers.)