CS 466/666, Fall 2007 due Friday Oct. 19

A. Lubiw

ASSIGNMENT 2

Do not copy from others, and acknowledge your sources.

1. [10 marks] Most graph algorithms that you have seen assume that the graph is given to you as input, and does not change. In *dynamic graph problems*, by contrast, the graph may change along the way.

   Suppose you want to store a dynamic forest on n nodes subject to the two operations: Insert(u, v), add a new edge (u, v); and Connected(u, v), decide whether or not u and v are in the same tree of the current forest.

   (i) Describe how to solve this problem using a union-find data structure, and give the amortized time per operation if the structure described in class is used.

   (ii) What if you want to add an operation Delete(u, v) that deletes an edge (u, v) of the dynamic forest. (Note that this is not the same as deleting an edge of a tree used in the data structure.) How much time will this operation take using the union-find data structure from class.

   Comment: The time bound in part (ii) will be quite bad. There are methods known to solve this dynamic forest problem in time $O(\log n)$ per operation, where n is the number of vertices in the forest. They are not simple. Reference (useless for answering the above question): G.N. Frederickson, Data structures for on-line updating of minimum spanning trees, SIAM J. Computing 14, 1985, 781–798.

   (iii) Consider the version of the problem where you want to Delete edges, but never Insert edges, and where the initial graph is a path. Describe a data structure that implements Delete, and Connected in $O(\log n)$ amortized time per operation.

2. [10 marks] Suppose you have a data structure to answer range queries for points in R^4 in time $O(\log^4 n + k)$ where k is the output size. Describe a data strucutre to store n rectangles in the plane and answer queries of the form: given a point (x, y) give all rectangles containing that point. Queries should be answered in time $O(\log^4 n + k)$ where k is the output size.

3. [10 marks] Suppose we have a Monte Carlo algorithm A for problem Π whose expected run time is at most $t_1(n)$ on any input of size n, and that produces a correct answer with probability $p(n)$. Suppose that we have an algorithm to test correctness of a solution to Π in time $t_2(n)$ (where n is still the size of the input to Π). Give a Las Vegas algorithm for Π that runs in expected time $O((t_1(n) + t_2(n))/p(n))$. Give complete details in your answer, because a main point of the question is to make you review some basic probability results.

4. [10 marks] This problem is about using randomness to maintain balance in a binary search tree, thus yielding a dictionary data structure with expected cost $O(\log n)$ for search, insert and delete. As usual each element in the dictionary has a key attached to it. In addition,

we will associate a *priority* with each element. The priority of an element will be chosen at random when the element is first inserted into the dictionary. Assume that keys and priorities are distinct. The search tree will be in binary search order with respect to the keys and *simultaneously* in heap order with respect to the priorities (minimum priority at the root).

(i) Prove that given the keys and priorities there is a unique tree with these ordering requirements.

(ii) Prove that such a tree has expected height $O(\log n)$.

(iii) Describe how to insert and delete in time $O(h)$ where h is the height of the tree. You may assume that single and double rotation routines are given, i.e. you do not need to give details about implementing rotations.