CS466/666, Fall 2009: Assignment 3

Out: October 19, Due: November 4, 5pm

1. **Minimum spanning trees in changing graphs:** Given a graph G and an MST T.

   (a) Suppose we decrease the weight of one e in T. How quickly can you compute the MST T' of the resulting graph G'?

   (b) Suppose we decrease the weight of one edge e not in T. How quickly can you compute the MST T' of the resulting graph G'?

   (c) Suppose we add a new vertex v with an arbitrary number of incident edges. How quickly can you compute the MST T' of the resulting graph G'?

   You may use randomized Las Vegas algorithms if they have a faster expected run-time.

2. **Union/Find with lists and weighted union heuristic:** Recall that one option for Union/Find was to keep lists with head-references, and during Union($|A|, |B|$) change the head-references of the not-longer list. We showed in class that m unions will take at most $O(n \log n)$ time.

   Professor Dumb thinks she can prove something better with potential functions. Assume that Union($|A|, |B|$) takes time $1 + \min\{|A|, |B|\}$ (i.e., ignore constant overhead.) Furthermore, assume that initially (at time 0) the data structure contains exactly one element. Define the following potential function:

   $$\Phi(i) = \log \left(\frac{n^n}{\prod_x \text{size}(x)} \right),$$

   where n is the number of elements stored at time i, the product is over all elements x stored at time i, and size(x) denotes the size of the list containing x at time i.

   (a) Show that with this potential function, Find and Union both take $O(1)$ amortized time.

   (b) Since one can easily show that $\Phi(0) = 0$ and $\Phi(i) \geq 0$ (you need not prove this), why doesn't this show that Union-Find can be implemented in $O(1)$ amortized time for all operations?

3. **Maintaining lists while splitting:** Show how to maintain a set of doubly-linked lists that can be split. Initially, the set contains just one list with n elements in it. Two operations can be applied, in arbitrary order:

   - Find(x): This returns a unique identifier of the list that currently contains x.
   - Split(ℓ, x): This splits list ℓ into two lists, by splitting after x.

   For both operations, x knows where it is in the list. Explain how to implement this data structure such that any Find can be done in $O(1)$ time, and the total time for all Split operations is $O(n \log n)$ time.

4. **Union/Find with Remove:** Consider the Union/Find problem, but with an additional operation Remove(x) that removes element x from its set and places it in a set by itself. You may assume that each element has a reference to where it is stored in the data structure, so Remove(x) need not find the element, only remove it.

   Show how to modiy the data structure such that a sequence of m Union, Find, and Remove operations on a set of n elements takes $O(n + \alpha(n)m)$ time.