ASSIGNMENT 1
CSC 425, FALL 2018

Due Thursday Sept. 20 at 11:55 PM online. Based on readings: CLR chapter 17, "Disjoint sets Tarjan, ET-trees reading

Homework policy: Homeworks should be submitted on line. Homeworks will be graded on the clarity of presentation, well-defined terms and clear and concise explanations, as well as correctness. I suggest you use .pdf generated with latex (see sample scribe notes in resources as a template). Homeworks are due by 11:59 PM of the due date. Homeworks may be accepted up to 3 days late, with a 15% penalty. Students may work together in groups of 2-3 but each person should understand each solution and write it up themselves. Please write the names of your collaborators on your paper. You are not allowed to ask anyone else for solutions or consult the internet for solutions.

(1) We modify the dynamic tables to allow for deletions. We use the strategy that we double the table size when an item is inserted whem the table is full, but we halve the table size when a deletion causes the table to become less than 1/4 full. (see page 422 in CLR). Use the *accounting method* to determine the amortized cost. Hint: put dollars on deleted entries in the table as well as inserted elements. How many dollars should you put on each? Do the analysis to show that your scheme pays for itself when doing any sequence of insertions and deletions and determine the lowest cost you can per operation.
What if you use the algorithm described in 17.4-3 page 425 (Again, use the accounting method)?

(2) Consider a binary search tree where each node x stores a key and also has a field $size$ = number of nodes in the subtree rooted at x (including x itself). A tree rooted at x is defined to be a-balanced where $0 < a < 1$ if $size$ of x's left child $\leq a\ size(x)$; and $size$ of x's right child $\leq a\ size(x)$.

A method for maintaining binary search trees with good amortized performance:
a) Given any ordinary binary search tree with n nodes, describe a linear time algorithm to build a 1/2-balanced tree. Explain why it is linear time.
b) Prove that the height of a tree in which every subtree is 2/3-balanced is

$O(\log n)$ by induction.
c) Suppose every time we insert or delete a node, we rebuild every subtree which is no longer 2/3-balanced, so that it becomes 1/2-balanced.
Devise an accounting system which can pay for the cost of rebuilding. Hint: leave credits at each node in the path from the node deleted or inserted. How many credits do you need to leave at each node, if it takes n credits to rebuild a tree with n nodes? Explain what is the amortized cost per delete or insert operation.

(3) Partition for ranks of a disjoint set implementation follows the one described in class and in Tarjans chapter on disjoint sets:
$B(0, j) = j$
$B(i, 0) = 0$
$B(i, 1) = B(i - 1, 2)$
$B(1, j) = 2^j j > 1$
$B(i, j) = B(i - 1, B(i, j - 1))$ for $i, j > 1$
a. Suppose that the ranks of nodes in a path from the leaf to a root in a disjoint set data structure have ranks A,B, C, D, E, F, G, H, I ,J K have ranks 0,1,2, 3, 4, 5, 6, 7,8, 9, 10 respectively. What are the levels of the nodes?
b. Now for the same problem, assume instead of using Ackermann's function to , we use this partition
$B[0, j] = j$
$B[i, 0] = 0$
$B[1, j] = 2^j$
$B[i, j] = 2^{\lfloor}B[i - 1, j]$
i. How many levels are needed (as a function of n)?
ii. Answer question (a) for this function.
iii. How does the amortized analysis change and what is the running time using this analysis?

(4) In the ET tree data structure, explain how to enable he data structure so that it quickly returns a new active occurrence if the active occurrence is deleted during a delete edge operation.

The questions below are extra credit for undergrads. Grad students are required to do one of their choice.

(5) Supplement ET-trees to perform the following operations:
$find - val(v)$: return $val(x)$
$find - min - val(v)$: return a vertex of minimum value in the tree set val v

equal to x
$change - val(v, x)$: set $val(v)$ equal to x
$add - val(v, x)$: add x to $val(w)$ for each vertex w in the tree containing vertex v

(6) In class we showed how to modify the fully dynamic connectivity algorithm to maintain MST for edge deletions. Explain what goes wrong when you try to do insertions. Try to come up with an idea to get around this.

(7) Explain how to reduce the worst case running time of Even and Shiloachs decremental connectivity algorithm, while maintaining a total time of $O(nm)$?*