Indian Institute of Technology Bombay
Department of Electrical Engineering
EE-7017 Advanced Computing for Electrical Engineers

Assignment 2

Submission Deadline: October 12, 2013 (Saturday), 11:55 pm (IST)

Note: You can use any language from a set of languages {C, C++, Java, Pascal}.

Question 1: Suppose we want to add the *DecreaseAllkeys (Δ)* operation to the *heap* repertoire. The result of this is that all keys in the *heap* have their value decrease by an amount Δ. Implement all the heap operations including *DecreaseAllkeys (Δ)*. It should retain the same run time of all other operations and run time of *DecreaseAllkeys(Δ)* is *O(1)*.
Test Input: Initialize {5 – 18}, DecreaseAllKeys(2), Add{19, 18}, DecreaseAllkeys(1), Delete, Delete, Delete. Show tree after every operation. Note that the Initialization is a single operation.

Question 2: A modification we could make to 2-3 tree is to keep only keys at interior nodes, but do not require that the keys k_1 and k_2 at node truly be minimum keys of the second and third subtrees, just that all keys k of the third subtree satisfy $k \geq k_2$, all keys k of the second satisfy $k_1 \leq k \leq k_2$, and all keys k of the first satisfy $k < k_1$. Implement such a tree.
Test Input: Initialize {1 – 32}, Delete{15 -- 8}, Insert/Add{33 – 36}.

Question 3: Write a program to implement a *splay tree* which performs standard tree operations, i.e. find, add, and delete. Start with a splay tree that is 15 node full tree, the keys are 1 – 15. Search keys in order 15, 14, 13, 12, 11, 10, ..., 2, 1, and then delete nodes 15, 14, 4, 5, and 1. Draw the tree after each operation. Compute the average number of operations carried out.

Question 4: One of the most expensive operations in an *AVL tree* is delete operation. One way to reduce average case complexity is use lazy deletion strategy. To delete a node, merely mark it as deleted. Actual deletion operation is performed periodically to collect garbage, say using *Deldeleted* operation. Implement tree operations.
Test Input: Initialize {1 – 20}, Delete{3, 4, 8, 9}, Insert/Add{21, 22, 8}, Delete{10, 12, 14}, Add{23, 24}, Delete{24, 8, 23}, Add {24, 25}. Assume that the garbage collection is performed after 5 deletion operations. Show results after every operation after initialization.