CS 535 Design and Analysis of Algorithms	**Fall Semester, 2021**

Homework 2 version 1.1

Assigned: September 15 *Due: September 29*

Remote students: please use Blackboard to submit your solutions. Sections 01 and 02: submit both hard copy and on Blackboard.

Notes for pseudocode usage:

1. C/C++/Java instructions are fine. But do not write object-oriented additions. Do not declare or use any class. Declare only procedures (if necessary) and explain in words what each procedure does, and what is the use of each parameter. Feel free to use as procedures algorithms from the textbook; indicate page and edition.
2. One instruction per line
3. Match the brackets with a horizontal line
4. Number your lines
5. Write down if your array is indexed $0 \dots n-1$ or $1 \dots n$.

Problem 1 This problem refers to binary search trees as defined in the Chapter 12 of the textbook.

Show that one TREE-MINIMUM followed by $(n-1)$ calls to TREE-SUCCESSOR takes $\Theta(n)$ times. In which order are the nodes visited?

Problem 2 This problem refers to binary max-heaps as defined in the Chapter 6 of the textbook, and the operations described and implemented there.

We want to claim an amortized cost of $O(1)$ for EXTRACT-MAX, which is actually possible with an amortized cost of $O(\lg n)$ for INSERT. Find a potential function Φ to obtain these bounds.

Problem 3 For Fibonacci heaps, the rule for cutting during the cascade is to cut a node from its parent after it loses 2 children. What if instead, the cut is done this after losing one child? Will the degree of the root still be provable $O(\lg n)$? Same question, if instead of 2 above one uses an integer $k > 2$.

Problem 4 Suppose the cascading cut rule for *decreasekey* were changed to "as soon as x loses *one* child through a cut, cut the edge joining x and its parent." Prove that if the amortized times of INSERT and UNION are $O(1)$ and the amortized time of DELETEMIN is $O(\lg n)$, then the amortized time of DECREASEKEY cannot be $O(1)$, n being the maximum number of elements in the data structure. That is, for any constant c, find an n as number of elements, and a sequence of operations with p INSERT and/or UNION, q DELETEMIN, and r DECREASEKEY, such that the running time exceeds $c(p + q \lg n + r)$.

Hint: one solution keeps all the trees in the Fibonacci Heaps in the shape of the trees in Binomial Heaps. Binomial trees B_k have been seen in class, and are defined as follows: B_0 has exactly one node (root of degree 0). For $k > 1$, the binomial tree B_k has a root with k children, and if they are numbered from $0, 1, \ldots, k-1$, child i is the root of a subtree B_i. One can easily prove by induction that B_k has height k and exaxtly 2^k nodes.