CS 535 Design and Analysis of Algorithms	**Spring Semester, 2019**

Homework 2 version 1.12

Assigned: Feb 7 *Due: Feb 21*

Problem 1 Let $G = (V, E)$ be a simple undirected graph with weights $w : E \to \mathbb{Z}^+$ and let $n = |V|$. The *inductivity* of a vertex ordering (permutation Π of V) $\left\langle v_{\Pi(1)}, v_{\Pi(2)}, \ldots, v_{\Pi(n)} \right\rangle$ is defined by

$$\max_{j\,:\,2 \leq \Pi(j) \leq n} \sum_{i\,:\,1 \leq \Pi(i) < \Pi(j)} w(v_{\Pi(i)} v_{\Pi(j)}). \tag{1}$$

Use (even if not completely covered yet) Fibonacci heaps to obtain a $O(|E| + |V| \log |V|)$-time algorithm (present pseudocode) to produce a least-inductivity vertex ordering of G, together with the proof of correctness. That is, find the permutation Π of V that minimizes Formula (1)

Hint: Use a greedy strategy paying attention to nodes with smallest weighted degree in G.

Give a $O(|E| + |V|)$-time algorithm for the unweighted case (all the weights are 1).

Problem 2 Describe a binary search tree on n nodes such that the average depth of a node in the tree is $\Theta(\lg n)$ but the height of the tree is not $O(\lg n)$. How large can the height of an n-node binary search tree be if the average depth of a node is $\Theta(\lg n)$?

Problem 3 Assume every node in a binary *search* tree has a pointer to its parent, in addition to pointers to the left and right child. Design an algorithm (write pseudocode), which, given a node v, finds w, the node-successor of v in inorder (the element of w is also the successor of the element of v in the sorted order of elements).

Analyze the running time of s consecutive calls to successor (that is, w is given as the argument to the next call, and so on) in terms of s and h, the height of the tree. A tight (within a constant) analysis is worth one third of the points.

Problem 4 Suppose we wish not only to increment a binary number, but also to reset it to zero (i.e., make all bits in it 0). Counting the cost to examine or modify a bit as 1, show how to implement a binary number as an array of bits so that any sequence of n INCREMENT and RESET operations costs $O(n)$ on an initially zero number.

Hint: Keep a pointer to the high-order 1.