COP-4534: Algorithm Techniques

Homework 4

DUE: Sunday April 8 at 11:55 PM

- Please remember that all submissions must be typeset. Handwritten submissions will NOT be accepted. These must be uploaded to SCIS moodle in PDF format only.

- Please remember to type your name on top of your submission.

1. (a) (10 points) Give an algorithm that sorts \(n\) 0/1-valued records (that is, the key of each record is either 0 or 1). Your algorithm should be stable and run in \(O(n)\) time.

(b) (10 points) Give an algorithm that sorts \(n\) 0/1-valued records. Your algorithm should be an in-place algorithm (that is, the algorithm uses a constant amount of storage space in addition to the original input array) and runs in \(O(n)\) time.

2. (20 points) Augment the binary search tree (BST) data structure so that it supports queries of the form “\(\text{rank}(k) = \?)”, where \(\text{rank}(k)\) is defined to be the number of nodes in the tree whose keys are less than or equal to \(k\). The running time of each query should be \(O(h)\), where \(h\) is the height of the BST. (that is, show how to add some additional fields to the BST data structure so that there is an efficient algorithm that can answer such queries.)

3. (20 points) Suppose we have \(k\) lists, \(S_1, \ldots, S_k\), each containing \(n\) real numbers. Design an efficient algorithm that lists all the sums of the form \(s_1 + \cdots + s_k\) in sorted order, where \(s_i \in S_i\) for every \(1 \leq i \leq k\). What is the time complexity of your algorithm?

4. (20 points) Given an array of \(n\) integers \(A = \{a_1, \ldots, a_n\}\), design an efficient algorithm that prints out all the missing number in the range defined by \(A\). That is, if \(S = \text{min}_{1 \leq i \leq n} a_i\) and \(L = \text{max}_{1 \leq i \leq n} a_i\), then your algorithm should list all integers \(k\) such that \(S < k < L\) and \(k \notin A\).

5. (20 points) In this problem we will compute the expected average size of a chain in the hash table, under the Simple Uniform Hashing assumption. The simple uniform hashing assumption states that each key is hashed into every bucket in the hash table with equal probability, regardless where any other elements have already been placed. Suppose a hash table has \(m\) buckets and \(n\) elements, and uses chaining to resolve collisions. Let \(n_i\) be the number of elements in bucket \(i\) and \(L_i\) be the length of the chain attached to bucket \(i\). Define \(\alpha = n/m\) to be the load factor of the hash table. You may find the following equation useful: when \(m\) is large (as we always assume), \((1 - \frac{1}{m})^m \approx e^{-1}\).

(a) What is the relation between \(n_i\) and \(L_i\)?

(b) Show that \(\sum_{i=1}^{m} L_i = n - m + N_0\), where \(N_0\) is the number of empty buckets.

(c) Show that \(E(\frac{1}{m} \sum_{i=1}^{m} L_i) = \alpha - 1 + e^{-\alpha}\) by computing the expectation of \(N_0\).