Department of EECS
York University

Term: Winter 2019
Instructor: Andy Mirzaian

EECS2011: Fundamentals of Data Structures
Sections M and Z
Assignment 4
Due: 10 pm, Wednesday, April 3, 2019

- Electronically submit your solutions in an a4sol.pdf file.
- Print your name, eecs account, and student ID # on top of the file.
- Express each algorithm in pseudo-code with sufficiently detailed explanation of how it works, its correctness, and its time analysis.
- You will be graded on correctness, efficiency, and clarity.

Problem 1: [35%] [GTG] Exercise C-10.50, page 455 somewhat modified:
Design and analyze an $O(\log n)$ worst-case time algorithm for the following:
Input: two sorted arrays S and T, each of size n, with a total of $2n$ distinct elements, and a positive integer $k \leq n$.
Output: the k^{th} smallest element in $S \cup T$.

Note that the case $n < k \leq 2n$ can be solved symmetrically by thinking "largest" instead of "smallest". The reason is that the k^{th} smallest element is the k'^{th} largest element, where $k' = 2n + 1 - k$. If $k > n$ then $k' \leq n$.

a) What is the output for the below instance with $k = 6$? What is it for $k = 10$?

$S =$	3	5	9	15	27	33	35	41	57	65
$T =$	2	16	18	42	44	46	48	50	52	54

b) First solve the problem for the special case $k = n$, i.e., find median of $S \cup T$.
Hint: investigate the relationship between $median(S)$, $median(T)$, and $median(S \cup T)$.

c) Now solve the problem for the general case of k as expressed in the problem statements.

Problem 2: [35%] [GTG] Exercise C-11.37, page 527:

Suppose we wish to support a new method $countRange(k_1\ ,\ k_2\ )$ that determines how many keys of a sorted map fall in the specified key range $[k_1, k_2] = \ \{k \mid k_1 \leq k \leq k_2\}$. We could clearly implement this in $O(\ s + h)$ time by adapting our approach to $subMap$. Describe how to modify the search-tree structure (e.g., its node structure) to support $O(h)$ worst-case time for $countRange$. Design and analyze such an algorithm.

> **Hint:** Augment each node of the search tree by a new field called $size$, where $size(v)$ is the size of the sub-tree rooted at v, i.e., the number of descendants of v (including itself). How will you use this new field to implement $countRange$ in $O(h)$ time? Also explain how you would revise the dictionary operations $insert$, and $delete$, to properly maintain this augmented field in a consistent manner without degrading their asymptotic running time.

Problem 3: [30%] [GTG] Exercise C-12.36, page 569:

Consider the voting problem from Exercise C-12.35, but now suppose that we know the number $k < n$ of candidates running, even though the integer IDs for those candidates can be arbitrarily large. Describe an $O(n \log k)$ worst-case time algorithm for determining who wins the election.