Department of EECS
York University

Term: Fall 2017
Instructor: Andy Mirzaian

EECS2011A: Fundamentals of Data Structures

Assignment 4

Due: 10 pm, Tuesday, December 5, 2017

- **Print your name, eecs account and student ID# on top of every file you submit.**
- **You may give your algorithms in either java code or precise pseudo-code.**
- **Make sure you also submit a4sol.pdf.**

Problem 1: [35%] ([GTG] Exercise C-11.35, page 527)
Consider a sorted map that is implemented by means of a standard Binary Search Tree T. Give an algorithm (in pseudo-code) to perform the operation $removeSubMap(k_1, k_2)$ that removes all entries whose keys fall within $SubMap(k_1, k_2)$, in worst-case time $O(s + h)$, where s is the number of entries removed and h is the height of T. Analyze and prove the claimed time complexity.

[See also Exercise C-11.34. Note that keys k_1 and k_2 may or may not be in T. Note also that the required time bound is $O(s + h)$, not $O(s * h)$. You may give a less efficient algorithm for partial credit.]

Problem 2: [25%] ([GTG] Exercise C-12.49, page 570)
Bob has a set A of n nuts and a set B of n bolts, such that each nut in A has a unique matching bolt in B. Unfortunately, the nuts in A all look alike, and the bolts in B all look alike as well. The only kind of a comparison that Bob can make is to take a nut-bolt pair (a, b), such that a is in A and b is in B, and test to see if the threads of a are larger, smaller, or a perfect match with the threads of b. Describe and analyze an efficient algorithm in pseudo-code for Bob to match up all of his nuts and bolts.

[Here efficiency is measured in terms of the number of nut-bolt comparisons. Note that we cannot have any nut-nut or bolt-bolt type comparisons. We can obviously compare every nut against every bolt, but that would require $\Omega(n^2)$ comparisons. Can we do better? Hint: think about QuickSort.]

Problem 3: [40%] (Center of a Graph)
Consider a given connected undirected graph G. (Here for simplicity we assume each edge has unit length.) The *eccentricity* of a vertex v in G is the length of the shortest path from v to the vertex farthest from v. The *diameter* of G is the maximum eccentricity of any vertex in G. The *radius* of G is the smallest eccentricity of any vertex in G. A *center* is a vertex whose eccentricity is the radius.

a) Prove that for every graph G : $radius \leq diameter \leq 2\, radius$.
b) Design and analyze efficient algorithms (in pseudo-code) that implement the following graph instance methods for any instance graph G:

$eccentricity(v)$: returns eccentricity of vertex v in G. [Hint: Use BFS.]
$diameter(\,)$: returns diameter of G.
$radius(\,)$: returns radius of G.
$center(\,)$: returns a center vertex of G. (Break ties arbitrarily.)

Problem 4: [20% Extra Credit and Optional] ([GTG] Exercise C-14.67, page 683)
Consider a diagram of a telephone network, which is an undirected graph G whose vertices represent switching centers, and whose edges represent communication lines joining pairs of centers. Edges are marked by their bandwidth (which are arbitrary positive numbers), and the bandwidth of a path is equal to the lowest bandwidth among the path's edges. Design and analyze an efficient algorithm (in pseudo-code) that, given such a network and two switching centers a and b, outputs the bandwidth of a maximum bandwidth path between a and b.
[Hint: Use a suitable adaptation of Dijkstra's algorithm.]