Practical Questions CSCA48– Week 10

What is the worst-case big-oh time complexity of each of the following problems? Assume that the problems are solved efficiently and that there are no restrictions on the algorithms that may be used.

a) Search an unsorted list of n elements for a given value.

b) Search a sorted linked list of n elements for a given value.

c) Search a binary search tree with n nodes for a given value.

d) Determine whether or not a sorted list of n elements has any duplicate values.

e) Determine whether or not an unsorted list of n elements has any duplicate values.

f) Given an integer list of length n, print the list in reverse order

g) Given a queue containing q items (are represented as a circular list), and a stack containing s items (and represented as a linked list), remove the head of the queue and push it onto the stack.

h) Given a tree of n notes as height h, return the number of occurrences of a given object.

i) Given a binary search tree of n nodes and height h, return the maximum value in the tree.

Sort the following functions in Big-Oh order. That is, write them in an order so that each function is in Big-Oh of the one immediately after it. For example, f1 < f2, f4 < f3.

$$6n^3,\ 2^n + \ n^2 - logn,\ 4^n, 98(logn),\ 3nlogn,\ 36,\ 8n^2,\ \ 92^n, \log{(n^2)}$$

$$n^2, \log_2 n\,, n, \log_2 n^2,\ 2^n, \log_{10} n\,, \log_2 2^n\,, n!\,, \log_2 n!$$

For each of the following operations briefly describe the most efficient (in the big-O sense) algorithm. Give the worst-case runtime efficiency in big-O (using the smallest, simplest expression).

j) Given a balanced binary search tree of n nodes containing integers, return tree if it contains the number 52.
Algorithm:

Runtime efficiency:

k) Given a linked list of n nodes containing integers, move the node with the largest value to the end if the list.
Algorithm:

Runtime efficiency:

l) Given two lists of n integers, check if every integer in the first list also appears in the second list.
Algorithm:

Runtime efficiency:

m) Given a balanced binary search tree of n nodes containing integers, return the biggest difference between any node and its successor.
Algorithm:

Runtime efficiency:

n) Given a sorted list of n integers, return the smallest integer in the list which is larger than 42 (if any).
Algorithm:

Runtime efficiency:

o) Given any list of n integers, determine if there are two elements in the list which sum to 100.
Algorithm:

Runtime efficiency:

p) Given a list of n integers, check if there is subset of this list whose elements sum to 0.
Algorithm:

Runtime efficiency:

q) Given a string of length n, return the letter that occurs the most times.
Algorithm:

Runtime efficiency:

r) Given a binary search tree of n nodes containing integers, return the difference between the smallest and largest values of the tree.
Algorithm:

Runtime efficiency:

s) Given a binary search tree of n nodes containing integers, return the sum of all the integers in the tree.
Algorithm:

Runtime efficiency:

t) Given a sorted list of n integers, return true of the list contains the number 42.
Algorithm:

Runtime efficiency:

u) Given a linked list of n nodes containing integers, remove the smallest m integers.
Algorithm:

Runtime efficiency:

v) Given an empty binary search tree, insert n values into the tree.
Algorithm:

Runtime efficiency:

w) Traverse a binary search tree of n nodes in order.
Algorithm:

Runtime efficiency:

x) <u>(Challenge)</u>

A square matrix of size n is a data structure that supports the following operations:

READ(i, j) returns the value in location (i, j)
WRITE(i, j, x) writes value x to location (i, j)
INITIALIZE allocates storage for a new matrix of size n,
and sets the value of all locations to 0.

Here, n is a global variable and $1 \le i, j \le n$. In addition, the operations *READ* and *WRITE* are performed in (worst-case) constant time.
Explain how to augment this data structure so that the following two operations can also be performed in (worst-case) constant time:

EMPTYROW(i) returns true if all elements of row i of the matrix are 0, and returns false otherwise.
EMPTYCOLUMN(j) returns true if all elements of column j of the matrix are 0, and returns false otherwise.

Justify why your data structure is correct and has the required performance.

y) <u>(Challenge)</u>

Consider a WWW search in which each hit is given a "percentage" rating where 100% means highly related to the search words and 0% means unrelated. The search engine needs to return the top k hits (ie. k hits with highest percentages.)

This is equivalent to the problem:
"Given a sequence of length N and a value k < N return the k highest values from the sequence. Assume that k << N."

There are two complexity issues:
(a) Time: We would like an algorithm that is faster than simply sorting the sequence and returning the first k elements.
(b) Space: We don't want to store all the elements while searching for the k elements.

Give an algorithm based upon a *min-heap* (the heap we learned in class) that has complexity better than O(N log N) time and better than O(N) space.

Explain what your algorithms complexity is for both time and space.

Discussion

1. Consider a function f(n) = 3(n^2) – 35n + 2,625,107.
   Which term is the most important?
   Why do we only care about this term?
   What is f(n) in Big-Oh?

Logic Question (Challenge)

With 10 characters… what is the largest number you can represent?

That's it… simple (this week's exercise is already a challenge, so you've got 2 this week… don't complain). You can use standard characters and mathematical notation (brackets don't count as characters, you can bracket things how you want). You can't do things like superscript or subscript a character.