Spring 2023 CSDS 410 Analysis of Algorithms
Prof. Dianne Foreback Assignment 2

Due Sunday, Oct 1, 2023 before 11:59 p.m. in Canvas

Directions:
Please upload your Assignment 2 into Canvas and make certain that the quality of the upload is clear. Also, enumerate your answers; e.g. before your answer for problem 1, preceded it with 1a, 1b, 1c, etc.

Please double check that your assignment is properly submitted into Canvas and is visible. Since file dates can be modified, and out of fairness to all, assignments not uploaded into Canvas by the final due date will not be accepted. That is, if your file is on a Google Drive but you forgot to place it into Canvas, it will not be accepted even if the file date shows an acceptable modify date.

Note that this assignment **may** be handwritten providing your writing is easily read.

Assignment Material:
This assignment covers recursive thinking, selection, union-find, and graph representation. Submit your work on canvas. Always check canvas for updates and corrections. Unless otherwise stated, whenever a question asks you to **describe an algorithm**, you should:

a. Explain the main concept of your algorithm
b. Give pseudo-code
c. Present an example of running your algorithm
d. Prove/justify its correctness and its running time.

Please label your problem answers with 1a. 1b. 1c. 1d as appropriate.

Problems:

1. [16 pts] **Describe** a recursive algorithm that prints the digits of an integer n > 0 vertically on a screen. For example, given the integer n=123, the output displayed on the screen is

1
2
3

Iterations of any types are not allowed. Recall, when **describing**, you need to answer parts a-d above the "problems" section of this document.

2. [16 pts] Suppose that we are given an unsorted array *A* with *n* elements and whose entries are integers between 1 and *n*. **Describe** a recursive algorithm for finding a repeated element in A (if any). Iterations of any types are not allowed. Recall, when **describing**, you need to answer parts a-d above the "problems" section of this document.

3. [8 pts] Suppose we have a social network with members A*, B, C, D, E, F, G, H* and *I* and the set of friendship ties

{(A, D), (A, E), (D, E), (F, B), (B, I), (E, H)}

What are the connected components?

Due Sunday, Oct 1, 2023 before 11:59 p.m. in Canvas

4. [3 pts] Suppose we represent a graph G having n vertices with an adjacency matrix. Why, in this case, would inserting an undirected edge in G run in $O(1)$ time while inserting a new vertex would take $O(n^2)$ time?

5. [3 pts] One additional feature of the list-based implementation of the union-find structure is that it allows for the contents of any set in a partition to be listed in time proportional to the size of the set. Explain how this can be done.

6. [8 pts] Suppose we have 20 singleton sets, numbered 0 through 19, and we call the operation **union**(**find**(*i*), **find**(*i+5*)), for *i=0,1,2,…,14*. Draw a picture of a list-based representation of the sets that result.

7. [6 pts] Consider a method, **remove**, which removes *e* from whichever list it belongs to, in a list-based implementation of a union-find structure. Explain how to modify the list-based implementation so that this method runs in O(1) time.

8. [6 pts] Let G be a simple connected graph with *n* vertices and *m* edges. Explain why *O(log m)* is *O(log n).* Note that a **simple graph** is a graph without parallel edges or self-loops. **Problem Hint**: Review the definition of Big-Oh and the combinatorial problem of how large m can be.

9. [10 pts] Suppose G is a graph with *n* vertices and *m* edges. Explain a way to represent *G* using *O(n+m)* space so as to support in *O(log n)* time an operation that can test, for any two vertices *v* and *w*, whether *v* and *w* are adjacent. Additionally, the representation of G should support the deletion of an edge in *O(log n)* time. Explain why your run time for these operations are O(log n).

10. [8 pts] Assume that the Radix-Sorting algorithm (see Canvas, Module 2, M2L3-Bucket-Radix-Sort.pdf, Slide 11 "Radix-Sort") is revised to sort via the most significant digit (MSD) to the least significant digit (LSD). That is, the for loop is revised to

for *i* ← MSD to LSD
    bucketSort(***S***, ***N***)

Give an example that does **not** properly sort digits of length 3. To do this, give your initial sequence of integers, then show the sequence after each pass of the for loop.

11. [16 pts] Consider an array of n distinct comparable elements. **Describe** a variant of the **Selection** algorithm to return an array of the first k-th largest elements. Your algorithm must run in O(n) time. Note, if k = 2, then the maximum element and the next to the maximum element will be returned. Recall, when **describing**, you need to answer parts a-d above the "problems" section of this document.