Assignment #3

COP-3530, Fall 2022

Purpose:

This assignment has **3 problems**. The problems 1 and 2 are related to some of the important topics we studied in the **Module 6**. The problem 3 is related to some of the important topics of **Module 7**.

The purpose of this assignment is:

- Value the use of the different **sub-quadratic sorting algorithms** and consider the possibility to **sort in linear time** for some particular problems.
- To apply efficient algorithms for some fundamental **problems on graphs**.
- To apply basic techniques of **algorithm analysis** to implement efficient algorithms.

Submitting Your Assignment:

- **Assignments must be turned in via Canvas**.
- **Please follow these steps for every assignment:**
  1. You are allowed to upload only a single **ZIP** file and ***no other kinds of compressed files will be accepted!***
  2. Please name your submission as **3_XXXXXXX.ZIP**, where **XXXXXXX** is your seven-digit Panther ID number.
  3. Inside **ZIP** folder, there should be a separate folder for each question (i.e. Problem 1, Problem 2, Problem 3, etc)
  4. **For questions that require Java implementation:**
     - The **.java** file must be inside the corresponding problem folder. DO NOT MIX ANSWERS.
     - ONLY SUBMIT **.JAVA** FILES. DO NOT SUBMIT YOUR WHOLE PROJETC'S FOLDER!!!
     - If is required, each .java file should contain ITS OWN main method at the bottom of the file. One main method for EACH .java file.
  5. **For written questions:**
     - Submit these files INSIDE the specific problem folder.
     - Each answer MUST be identified. It should be easy to tell which question and subsection you are answering!
     - Written questions must be only in **PDF** format.
  6. Please include the following header for each **Java** program:

```
/**********************************************************************
  Purpose/Description: <a brief description of the program>
  Author's Panther ID:  <your Panther ID number>
  Certification:
    I hereby certify that this work is my own and none of it is the work of
    any other person.
**********************************************************************/
```

  7. **Submissions turned in after the due date and/or which don't meet the established formatting rules will not be accepted.**

☞ ***Failure to follow these simple directions may result in a loss of credit for the assignment.***

Problem #1: (30 pts)

(a) Implement (in **Java**) the **radixSort** algorithm to sort in **increasing order** an array of integer positive keys.

public void radixSort(int arr[])

In your implementation you must consider that each key contains only **even digits** (0, 2, 4, 6, and 8). Your program must detect the case of odd digits in the keys, and, in this case, abort.

Example #1:

Input: 24, 12, 4, 366, 45, 66, 8, 14

Output: *** Abort *** the input has at least one key with odd digits

Example #2:

Input: 24, 2, 4, 466, 48, 66, 8, 44

Output: 2, 4, 8, 24, 44, 48, 66, 466

(b) What is the **running time complexity** of your **radixSort** method? *Justify.*

Important Notes:

- To storage and process the bucket lists, use an **ArrayList** structure.
- You must add the main method in your program in Java in order to test your implementation.
- You can use the array of the previous example to test your program, however, I suggest that you also use other input arrays to validate the correctness and efficiency of your solution.
- Your program MUST be submitted only in source code form (.java file).
- A program that does not compile or does not run loses all correctness points.

Problem #2: (35 pts)

(a) Given the following list of numbers:

3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5

trace the execution for **quicksort** with **median-of-three** partitioning and a cutoff of 3.

(b) The processing time of an algorithm is described by the following recurrence equation (c is a positive constant):

$$T(n) = 3T(n/3) + 2cn; \quad T(1) = 0$$

What is the **running time complexity** of this algorithm? *Justify.*

(c) You decided to improve insertion sort by using binary search to find the position p where the new insertion should take place.

(c.1) What is the worst-case complexity of your improved insertion sort if you take account of only the comparisons made by the binary search? *Justify*.

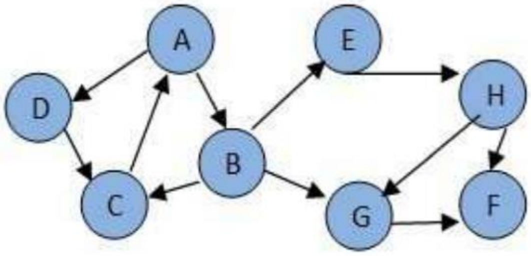
(c.2) What is the worst-case complexity of your improved insertion sort if only swaps/inversions of the data values are taken into account?

Problem #3: (35 pts)

(a) Either draw a graph with the following specified properties, or explain why no such graph exists:

A simple graph with five vertices with degrees 2, 3, 3, 3, and 5.

(b) Consider the following graph. If there is ever a decision between multiple neighbor nodes in the **BFS** or **DFS** algorithms, assume we always choose the letter closest to the beginning of the alphabet first.



(b.1) In what order will the nodes be visited using a **Breadth First Search** starting from vertex A and using a queue ADT?

(b.2) In what order will the nodes be visited using a **Depth First Search** starting from vertex A and using a stack ADT?

(c) Show the ordering of vertices produced by the topological sort algorithm given in class starting from vertex V_1 when it is run on the following direct acyclic graph (represented by its adjacency list, in-degree form). Justify.

V_0	---
V_1	---
V_2	V_0, V_1
V_3	V_0, V_1
V_4	V_0, V_2
V_5	V_1
V_6	V_2, V_4, V_5
V_7	V_6