Sorting and Hashing Assignment

This assignment was jointly created by Eduardo and Raunak. Kindly direct any queries about the **programming part to exb406@case.edu and written part to rxb641@case.edu.**

For this assignment, there will be 2 *additional* office hours on top of the usual TA office hours:

- Eduardo – Monday April 15th at 5pm (https://cwru.zoom.us/my/eduardob?pwd=Z1VNVWZvL3Bsck9KV28wTjhWbEJNdz09)
- Raunak – TBA in Canvas announcements next week

Written Part (40 P):

Submit your answers to this part as a pdf on canvas. Do not include the Written Part in your zip file for the programming part.

1. Given an array of unordered integers A and a target difference D, design an algorithm with $\mathcal{O}(n)$ runtime complexity to find the first pair of integers within A such that their difference equals D. The order in which the integers appear in the pair should match their order in the array (i.e., if the pair is (a_i, a_j), then $i < j$). Your algorithm should utilize a hash table of size n the size of the array, for efficient computation. Note: The algorithm only needs to return the first pair that meets the criteria. (10P)

2. Given input: 4371, 1323, 6173, 4199, 4344, 9679, 1989 and hash function h(x)=x (mod 10), show the result of inserting these keys into a hash table (of size 10) with:
a. Separate chaining. (3.5 P.)
b. Open addressing with linear probing. (3.5 P.)
c. Open addressing with quadratic probing. (4 P.)
d. Open addressing with double hashing and the secondary hash function h2(x) = 7 - (x mod 7). (4 P)

3.Let an array arr = [9, 8, 8, 5, 7, 7, 4, 4, 4, 2]. Sort arr from the smallest to largest value using:
a. Selection sort. (3.5P)
b. Insertion sort. (3.5P)
c. Quicksort, by partitioning around the last element. (4P)
d. Mergesort. (4P)
To earn credit, you must show all steps for all algorithms by:
● Writing the arrays after each step for parts a and b
● Drawing the recursive call tree, similar to figure 16-3 on page 11 of slide Ch16, for parts c and d

Programming Assignment (60 p):

Public class Binary

Write a class Binary and implement the following methods using binary search:

- **public int findMin(int[] nums):** Suppose an array sorted in ascending order is rotated to the right between 1 and len(array) times. Given the sorted array nums of unique elements, return the minimum element of this array. Notice that rotating an array [a[0], a[1], a[2], a[3]] 1 time results in the array [a[3], a[0], a[1], a[2]].

Public class RandomQuickSort

Write a class RandomQuickSort and implement the following methods implementing the quickSort algorithm:

- **public static <T extends Comparable <?super T>> void quickSort(T[] nums, int first, int last):** sort the array using QuickSort. This time you will have to choose a random pivot. You are allowed to use java.util.Random. You are also allowed to create a helper function for partition. **However, you are not allowed to use any built-in functions that do the sorting for you like .sort().**

Public class Solution

Write a class Solution and implement the following method using any sorting algorithm of your choice that runs at a time complexity of O(nlogn) or faster:

- **public int[][] kClosest(int[][] points, int k) :** Provided an integer k and an array of points denoted as points[i] = $[x_i, y_i]$, indicating coordinates on the X-Y plane, find the k points closest to the point (x = 233, y = 233). The distance between two points on the plane is calculated using the Euclidean distance formula: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. The solution can be returned in any sequence. N**ote: You are allowed to use any Java built-in functions except for sort().**

Final Notes: Please submit a ZIP file with only the necessary files to run your code and your unit tests. Your code will be tested using BlueJ, so make sure your code works properly using that IDE.

Rubric
[40 pts] Written Problems

[10 pts] Binary Class

[10 pts] RandomQuickSort Class

[20 pts] Solution Class

[10 pts] Comments: All loops and recursive calls should have a comment explaining what they do.

[10 pts] Unit tests for each class and method.