DSC 40B - Homework 05

Due: Wednesday, February 10

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer. Homeworks are due via Gradescope on Wednesday at 11:59 p.m.

Note: We've decided to push the homework due date back to *Wednesday* for the rest of the quarter, starting with this homework. This is so that it is no longer due on the same day as DSC 30's homework.

Problem 1. *(Eligible for Redemption)*

Suppose you need to keep track of a large dynamic collection of numbers. The three most common operations you will be performing are insertions, deletions, and queries for the maximum element in the collection. You may assume that the number of maximum queries you will be making is close to the number of insertions/deletions. You will also be performing membership queries.

Which data structure would you use: a (balanced) binary search tree, or a hash table? Justify your answer by comparing the time complexity of max queries, membership queries, and insertions/deletions between the two options. You may assume that the expected time complexity is most important for your application.

Programming Problem 1.

Recall that a *mode* of a collection is an element which occurs with the greatest frequency. For example, 4 is a mode of the collection $4, 5, 8, 3, 4, 2, 4, 5, 5$. 5 is also a mode, since it occurs just as frequently as 4.

In a file named `mode.py`, create a function named `mode(numbers)` which takes in an array of numbers and returns a mode. To receive full credit, your function should have the best possible average case time complexity. If there are multiple modes, your function need only return one of them.

Note: it's up to you to determine the best possible time complexity and convince yourself that your code is efficient! Think back to when we talked about theoretical lower bounds. Can you come up with one for this problem? This is a situation you'll find yourself in often when writing code: there's usually no resource that tells you what the best time complexity is for a given problem, so you'll have to convince yourself that your solution is optimal.