CSE 250 - FA23 - Written Assignment #4

Content Covered: PA2 Reflection, Trees

Submission Process, Late Policy and Grading

Due Date: 11/15/23 @ 11:59PM
Total points: 100

Your written solution may be either handwritten and scanned, or typeset. Either way, you must produce a PDF that is legible and displays reasonably on a typical PDF reader. This PDF should be submitted via autolab as WA4. You should view your submission after you upload it to make sure that it is not corrupted or malformed. Submissions that are rotated, upside down, or that do not load will not receive credit. Illegible submissions may also lose credit depending on what can be read. Ensure that your final submission contains all pages.

You are responsible for making sure your submission went through successfully.

Written submissions may be turned in up to one day late for a 50% penalty.
No grace day usage is allowed.

Problem 1 - Pre-Computation

[50/100 points]

For PA3, the StreetGraph class used to store the maps we were searching was implemented using an EdgeList data structure. The first function you had to implement created an external AdjacencyList that you could use for searching your graph.

a. **[10 points]** Derive the unqualified worst-case runtime (in terms of |V| and |E|) to perform a BFS search on a Graph that is implemented using an EdgeList. Show your work. **Note:** this is not what you implemented in PA2.

b. **[5 points]** For PA2, to perform a BFS search of the graph, we first computed the AdjacencyList and then searched the graph. What is the total unqualified worst-case runtime (in terms of |V| and |E|) to compute the AdjacencyList *and then* perform the BFS search. Justify your answer.

c. **[5 points]** Given your answer to part (b) is it worth it to take the time to compute the AdjacencyList before performing BFS if your input Graph is implemented using an EdgeList? Why or why not?

d. **[9 points]** Now imagine instead of searching a Graph, we are searching an LinkedList of data. If the LinkedList contains *n* elements, what is the unqualified worst-case runtime in terms of *n* to:

   i. Find a specific value in the LinkedList?
   ii. Insert all *n* items in the LinkedList into a balanced BST?
   iii. Find a specific value in a balanced BST?

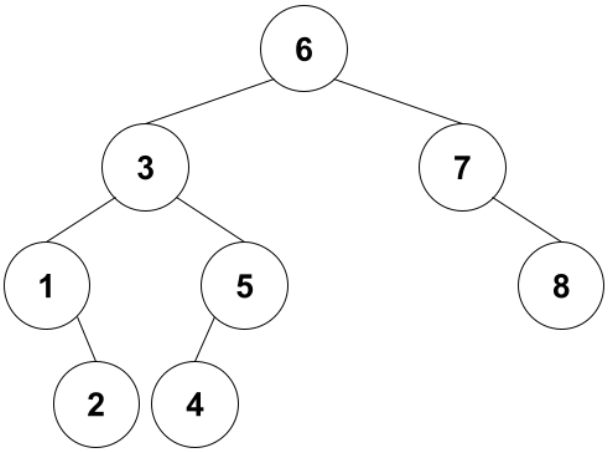
e. **[12 points]** Based on your answers for (d):

   i. What is the runtime to perform *n* searches in row on the LinkedList?
   ii. What is the runtime to perform *n* searches in row on the balanced BST?
   iii. What is the total runtime to convert the LinkedList to a balanced BST and then perform the *n* searches?
   iv. Is it worth it to add the LinkedList to a balanced BST first and then perform the searches?

f. **[9 points]** Now imagine that the LinkedList contains the reviewer scores for every single movie on RottenTomatoes, and you want to find the 10 highest rated movies. Is it worth it to store the LinkedList in a BST before searching for the top 10 movies? Explain.

Problem 2 - Balanced Binary Trees

[30/100 points]



a. **[16 points]** The above tree meets the structural requirements for a Red-Black tree. Determine the balance factor for each node, and a valid coloring for each node. When writing out your answer, write out 1-8 each on its own line, followed by the balance factor and color for that node.

b. **[10 points]** Does the above tree meet the structural requirements to be an AVL tree? Explain why or why not.

c. **[4 points]** Give a value that could be inserted into the above tree and not break the Red-Black tree constraints. Give a value that could be inserted into the above tree that would break the Red-Black tree constraints and therefore require additional operations to fix. (Assume the tree stores integers only, and does not allow duplicate values)

Problem 3 - Heaps

[20/100 points]

For this problem, consider the array of integers [64, 32, 16, 8, 4, 2]

a. **[8 points]** Draw (as a tree), the min heap that would result from adding each integer in the above array to an initially empty min heap one at a time, in order (starting with the 64).

b. **[2 points]** What was the total number of swaps required when creating the min heap in part (a)? (Note we are looking for the total number of individual swaps, not the number of calls to fixUp or fixDown)

c. **[8 points]** Draw (as a tree), the min heap that would result from calling heapify on the above array.

d. **[2 points]** What was the total number of swaps required when creating the min heap in part (c)? (Note we are looking for the total number of individual swaps, not the number of calls to fixUp or fixDown)