0 Overview

Instructions

Due Date: Sunday Apr 6 @ 11:59PM

Total points: 75

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 may submit as many times as you like, but only your last submission will be graded and should include the entire submission. 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 or incomplete submissions will 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.

1 Questions

Part 1 - Graph Data Structures

For PA2, 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.

1. **[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. **Justify your answer. Answers without justification will not receive credit.**

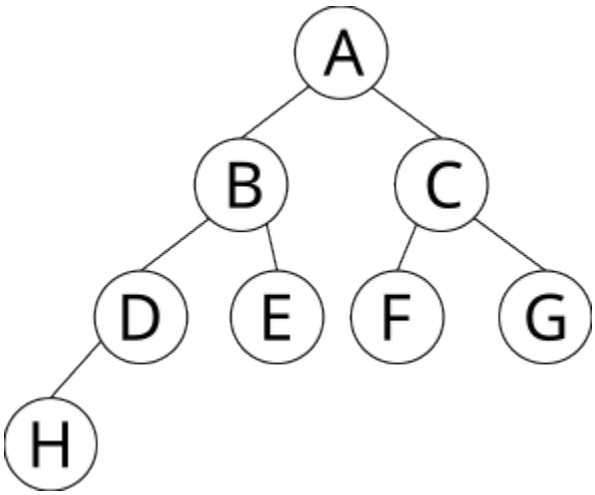
   Note: This is not what you implemented in PA2.

2. **[10 points]** Derive the unqualified worst-case runtime (in terms of $|V|$ and $|E|$) to create an Adjacency List representation of a graph from an Edge List representation. For this implementation you may assume that each `Vertex` object can hold a reference to a `List<Edge>` object, and that your algorithm must populate that list with ALL edges that are incident to the vertex (not just the outgoing edges). You may assume that the `Edge` objects hold a reference to their origin and destination vertices just as they did in PA2. **Justify your answer, and in your justification make sure to include what data structure you would use for the `List<Edge>` and why. Answers without justification will not receive credit.**

3. **[5 points]** Derive the unqualified worst-case runtime (in terms of $|V|$ and $|E|$) to convert a graph represented as edge list to one represented as an adjacency list AND THEN perform a BFS search on the adjacency list representation of that graph. Given that runtime, if you have a graph that is implemented as an edge list that you want to perform BFS on, is it asymptotically faster to just directly do the search using the edge list, or to convert to an adjacency list and then search the adjacency list. **This question requires two answer: a runtime, and whether or not you should convert an edge list before searching it. You must justify both of your answers. For the justification, you may refer to your answers to questions 1 and 2 as well as material discussed in class. Answers without justification will not receive credit.**

Part 2 - Binary Trees

For questions in this section refer to the following binary tree. Note that the letters are not the values of the nodes, just a way for you to refer to nodes in your answers.



4. [**5 points**] For each of the nodes, A-H, assign it a value from the set $\{0, 1, 3, 4, 5, 7, 8, 9\}$ in such a way that the tree would be a valid min heap, **OR** state that it is impossible to do so and explain why it is impossible.

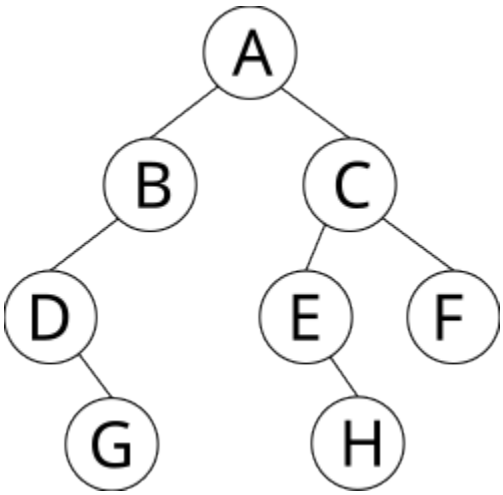
5. [**5 points**] For each of the nodes, A-H, assign it a value from the set $\{0, 1, 3, 4, 5, 7, 8, 9\}$ in such a way that the tree would be a valid BST, **OR** state that it is impossible to do so and explain why it is impossible.

6. [**5 points**] For each of the nodes, A-H, state the **DEPTH** of the node.

7. [**5 points**] What is the depth of the tree?

Part 3 - More Binary Trees

For questions in this section refer to the following binary tree. Note that the letters are not the values of the nodes, just a way for you to refer to nodes in your answers.



8. [**5 points**] For each of the nodes, A-H, assign it a value from the set $\{0, 1, 3, 4, 5, 7, 8, 9\}$ in such a way that the tree would be a valid min heap, **OR** state that it is impossible to do so and explain why it is impossible.

9. [**5 points**] For each of the nodes, A-H, assign it a value from the set $\{0, 1, 3, 4, 5, 7, 8, 9\}$ in such a way that the tree would be a valid BST, **OR** state that it is impossible to do so and explain why it is impossible.

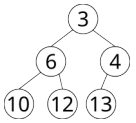
10. [**5 points**] For each of the nodes, A-H, state the **HEIGHT** of the node.

11. [**5 points**] Could the above tree be an AVL tree? Explain why or why not.

Part 4 - Space Complexity

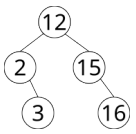
In lecture we discussed a method for using an `ArrayList` to store the elements of a `Heap`. Each entry in the `ArrayList` represents a node in the heap. A non-existent node in the heap corresponds to an empty entry in the `ArrayList`. The children of the node at index i are located at indices $(i+1)*2-1$ and $(i+1)*2$. The parent of the node at index i is located at index $(i-1)/2$. Note we are using integer division here, so for example the parent of index 6 is at $(6-1)/2 = 2$.

For example, the array `[3,6,4,10,12,13]` would correspond to the following heap:



Since a `Heap` is just a binary tree, we could use this same exact strategy to store the data for any binary tree in an `ArrayList`. Since a `Heap` is a complete binary tree, all of the empty spots in the tree come at the end, and therefore all of the empty spots in the `ArrayList` are at the end. For an arbitrary binary tree, this is not necessarily the case. There may be empty spots in the middle of the tree and therefore in the middle of the `ArrayList`.

For example, the array `[12,2,15,-,3,-,16]` would correspond to the following binary tree (the `-` represents an empty spot in the array, for example `null` or `Optional.empty()`):



Since there are no empty spaces between elements, a `Heap` with n elements can be stored in an `ArrayList` in $O(n)$ space. (The number of entries required in the `ArrayList` is $\leq c \cdot n$ for some constant $c > 0$).

12. [**10 points**] Give the worst-case bound on the amount of space required to store an arbitrary BST in an `ArrayList` using this representation. **Justify your answer by writing the number of entries required as a summation. An answer without justification will receive no credit.**

    Hint: Try to first come up with a BST that would have the most empty spaces in the `ArrayList` and go from there.