0 Overview

Instructions

Due Date: Sunday Nov 9 @ 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 WA3. 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

4. [**5 points**] Draw a valid min heap (as a tree) containing the following values: $\{5, 15, 25, 35, 45, 55, 65, 75, 85\}$

5. [**5 points**] Write out the array representation of the heap you drew in the previous question.

6. [**5 points**] Draw a valid BST with a height of 4 containing the following values: $\{5, 15, 25, 35, 45, 55, 65, 75, 85\}$

7. [**5 points**] Answer the following questions about the tree you drew for the previous question:

   a) Give a value that, if inserted into your tree from the previous question, would increase its height.

   b) Give a value that, inserted into your tree from the previous question, would not change the height.

8. [**5 points**] Draw a valid AVL Tree containing the following values: $\{5, 15, 25, 35, 45, 55, 65, 75, 85\}$

9. [**5 points**] Give a value that, if inserted into your tree from the previous question, would require one or more rotations to restore AVL properties, or state that no such value exists.

Part 3 - Induction

Consider the following statement:
$P(h)$: A perfect binary tree of height h has exactly $n = 2^{h+1} - 1$ nodes.

For this part we will prove, using induction on h, that $P(h)$ is true for all $h \geq 0$.

10. [**5 points**] Prove that $P(h)$ is true for an appropriate base case.
11. [**5 points**] State an appropriate inductive assumption.
12. [**10 points**] Prove the inductive step based on your given assumption.

    Hint: Consider how many nodes are in one level of a perfect binary tree. From there you know how many nodes you need to add to a perfect binary tree of height k to get to a perfect binary tree of height $k + 1$.