Written Assignment #3 - Answer Key

Content Covered: PA3 Reflection, Balanced Binary Search Trees

Problem 1 - Pre-Computation

[16/30 points]

a. **[4 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 PA3.
The algorithm is structured the same, we know that each vertex will be enqueued and dequeued from the work queue exactly one time. But now, instead of the work for each vertex being proportional to its degree, it is proportional to the total number of edges in the graph, since we have to check every edge. Therefore the runtime becomes:

$$O(|V| * |E|)$$

b. **[2 points]** For PA3, 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.
As specified in the handout, or via analysis, the cost to build the adjacency list is *O(|E|)*, and the cost to perform BFS using an adjacency list is *O(|E| + |V|)*. Therefore the runtime to build an adjacency list then perform BFS is:

$$O(|E|) + O(|E| + |V|) = O(|E| + |V|)$$

c. **[2 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?

Yes it is worth it. The complexity when precomputing is O(|E|+|V|) which is less than the complexity of searching with an edge list, which is O(|E|*|V|).

d. **[3 points]** Now imagine instead of searching a Graph, we are searching an Array of data. If the Array contains *n* elements, what is the unqualified worst-case runtime to:

   i. Find a specific value in the array if it is unsorted? **O(n)**
   ii. Find a specific value in the array if it is sorted?**O(log n)**
   iii. Sort the array? **O(n log n)**

e. **[3 points]** Based on your answers for (d):

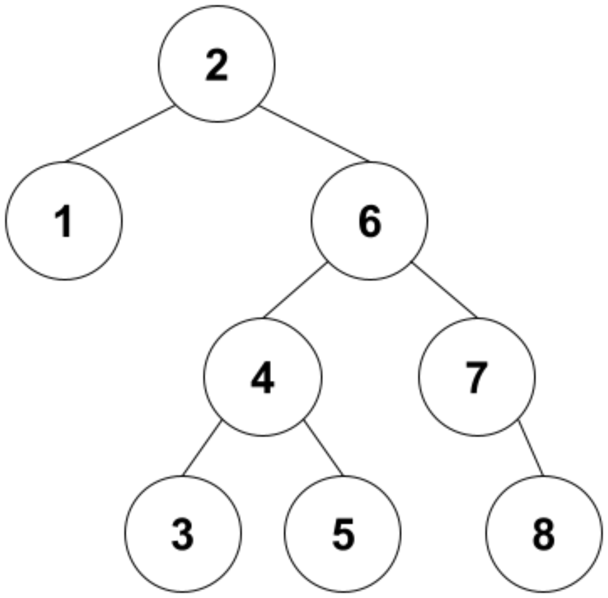
   i. What is the runtime to perform *n* searches in row on the unsorted array? **O(n^2)**
   ii. What is the runtime to sort the array first *and then* perform the *n* searches? **O(n log n)**
   iii. Is it worth it to sort the Array first and then perform the searches? **Yes**

f. **[2 points]** Now imagine that the Array contains times for competitors at an event to complete a race, and you want to find the top 3 times. Is it worth it to sort the Array first or not? Explain.

   No. We can find the top 3 times in one single pass through the array, which costs O(n). If we sort first it costs O(n log n) to sort, then O(1) to get the first 3 elements, but that is more expensive than just one pass through the unsorted array.

Problem 2 - Balanced Binary Trees

[10/30 points]



a. **[4 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.
   1. 0, black
   2. 2, black
   3. 0, red
   4. 0, black
   5. 0, red
   6. 0, red
   7. 1, black
   8. 0, red

b. **[2 points]** Does the above tree meet the structural requirements to be an AVL tree? Explain why or why not.
   No. The node with value 2 has a balance factor that exceeds +/-1

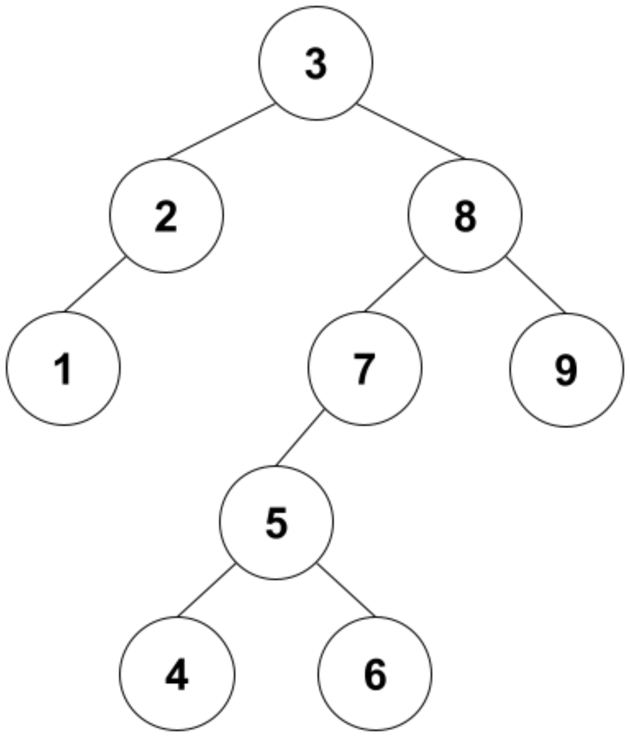
c. **[2 points]** Give a value that could be inserted into the above tree and not break the Red-Black tree constraints. Give a value that could be inserted into the above tree that would

break the Red-Black tree constraints and therefore require additional operations to fix. (Assume the tree does not allow duplicate values)
Inserting 0 (or less) would not break constraints (could color it red)
Inserting 9 (or greater) would break the constraints (can't color it red or black because the deepest leaf now has depth 5 and the shallowest has depth 2

d. **[2 points]** Recall that a Tree is also a Graph, therefore we can perform BFS and DFS on trees as well. Does performing a Depth-First Search starting at the root of an arbitrary BST to an arbitrary node of the tree find the shortest path to that node? Explain why or why not.
Yes. By definition a tree only has one path from the root to any node in the tree. DFS will find that path, which is by definition the only and therefore shortest path.

Problem 3 - Fixing Balanced Binary Trees

[4/30 points]



1. **[2 points]** The above tree does not meet the structural contraints to be an AVL tree OR a Red-Black tree. What is the **shortest series** of rotations you could perform that would cause the tree to meet the structural constraints of an AVL tree. Write your answer as a series of rotations, where each rotation is either: `rotateLeft(N)` or `rotateRight(N)` where `N` is the value of the node that you are rotating around.
   rotateRight(8)

2. **[2 points]** After performing the rotations from part (a) to make the tree meet AVL tree constraints, will it also meet Red-Black tree constraints? Why or why not?
   Yes. AVL constraints are more restrictive than Red-Black trees, so ALL AVL trees are red black trees. Alternatively, for the newly rotated tree, the depth of the deepest and shallowest EmptyTree nodes differ by at most a factor of two, and/or we can show a valid coloring.