ALGORITHMS, FALL 2018, HOMEWORK 7

Due Sunday, October 21 at 11:59pm. No late submissions, no extensions.

Worth 1% of the final grade.

1. Let $S = \{s_1, \ldots, s_n\}$ be a list of distinct real numbers.

   (a) Show how to find the smallest value $|s_i - s_j|$ $(i \neq j)$ in $O(n \log n)$ time.

   If we initially sort the list using quicksort (which is $O(n \log n)$), and then iterate by twos over the array to get the minimum difference (which is $O(n)$), the runtime of this is $O(n \log n)$.

   (b) Now suppose that there will be a mix of operations: besides needing to handle multiple queries such as the one in part (a), you must also allow insertions into S. Show how to maintain a simple data structure that can answer a query in constant time, and that takes at most logarithmic time to update after each insertion.
   If you can't do constant time queries, then at least manage logarithmic time. Your data structure should be as simple as possible, in the sense that you should not overload it unnecessary extra information.

   You can use a red-black tree to store the elements, which has $O(\log n)$ insertion. Then we also maintain a global min-difference, and every time something is inserted, it will calculate the difference between itself and it's parent, as well as with each node above it for a smaller difference. This is $O(\log n)$ because the tree is balanced.

   It is constant time to request the min-difference because it's stored.

   (c) Assuming you have used the simplest possible data structure in (b), it will probably be hard to also handle deletions from S in logarithmic time. Show how to handle deletions by storing extra information.

   If each subtree stores the min-difference of its children, then as long as the subtree didn't get deleted from or inserted from its min-difference will always be correct.

   If the min-difference needs to be updated we can proceed as follows:

   The root of the subtree will check its left-child for its subtree's min-difference, the right child for THAT subtree's min-difference, its difference with the leftmost child of the right subtree, and its difference with the rightmost child of the left subtree (these differences can be cached for performance), and select the minimum as the new minimum for that subtree. Regardless of whether it has changed or not, it will pass an update request up to its parent, making sure that the parent keeps it's min-differences updated as well. Traversing for leftmost and rightmost is $O(\log n)$ because its a balanced binary tree.

   Constant time for min-difference because it's stored at root