Foundations of Algorithms

Programming Assignment #1

The following is a problem to be completed by the individual (i.e., it is not collaborative) and then implemented. You may use any programming language you wish. Please follow the requirements provided under Syllabus & Course Information under the link Programming Assignment Requirements.

1. ***Programming (non-collaborative)—Due at the end of Module 5.*** Consider the following implementation of a set. We use a pair of arrays where one array is longer than the other, and both arrays are kept sorted. The length of the shorter array is the square root of the length of the longer array; although, it may have empty entries if it is not yet full. To add an element, you use insertion sort to insert the element into the shorter array. If the short array fills up, re-allocate the shorter and longer arrays and merge the two old arrays into the new longer array. The new short array is empty. Then, to find an element, use binary search on both the long and the short arrays.

   (a) [50 points] Given this data structure, prove that the data structure yields $O(\lg n)$ time for searching but $O(\sqrt{n})$ amortized time for adding an element.

   (b) [50 points] Implement the above data structure and insert/search operations. Test your implementation to verify the predicted complexity bounds are achieved.