Tutorial 03 – Binary Search

Task Explanation

Download the BinarySearch.java

Implement the BinarySearch API, which has been left uncompleted. Design and implement an algorithm by completing the implementation of the method with the signature **public static int indexOf(int[] a, int key)**, which, given a set of integers as input to be searched upon another sorted array of integers, it returns only those integers, which cannot be found on the sorted array. For instance, given as input the integers from tinyW.txt to be sorted and binary searched, as well as the set of integers from the file tinyT.txt to check upon, the algorithm shall return:

- 50
- 99
- 13

Since these are the only integers that do not exist in the input set.

Task Items

a) Having completed the implementation of the method, suggest its performance in terms of order of growth classification.

b) Verify the estimated order of growth classification by measuring the times spent on generating the outcomes.

Download the tinyT.txt test file. (It is compressed, unzip it to get the text file). Using the test file perform the test against each input file shown on the table, and measure the time take. (The input files are compressed, unzip to get the text file).

Input File (click to download)	Input Count (N)	T(N) Time
tinyW.txt		
largeW.txt		

c) Suggest an empirical method on how to estimate performance when it comes to using large sets of integers. That is largeW.txt as an input with a 80MB largeT.txt as a test file.

Homework

1. Analyse the performance of BinarySearch with largeW.txt as the database with largeT.txt as search file.
2. Analyse the performance of Binary Search Tree. It uses a Tree structure to structure the data points.
    a. Download BinarySearchTree.java
    b. Modify it to hold integer values. Then use tinyT.txt as a test file and measure the time with tinyW.txt, largeW.txt. Compare the results.