CS 245 - Practice Assignment 3 Fall, 2019

Practice Assignment 3 - Searching

The goal of this assignment is to practice the implementation of the search algorithms we discussed and to determine the efficiency as measured by (unreliable) wall clock time.

Background

We discussed two algorithms for searching: linear search and binary search. Binary search has two variants: a recursive version and an iterative version. We determined that linear search has a running time of O(n) and that binary search has a running time of $O(\log_2 n)$, with the iterative version being more efficient than the recursive version. This assignment will have you implement these algorithms as functions and test the timing of the implementations.

Requirements (Process)

Requirement 1: Get the files you need. You will copy them to your own GitHub repository using the following procedure:

1. Log into GitHub. If you do not have a GitHub account, create one via github.com > Sign Up.
2. Point your browser to the URL https://classroom.github.com/a/XKtIToMn.
3. If necessary, authorize GitHub Classroom by selecting the button “Authorize github.”
4. If available, select your name from the list of names available. This will link your GitHub ID.
5. Accept the assignment by selecting the appropriate button.

If successful, your repository should contain three (3) Java files:

- Practice03Test.java — the main file
- Practice03Factory.java — the factory for the class Practice03Search
- Practice03Search.java — the interface for the search classes

Requirement 2: Add to the code in order to make it run. Specifically, you must create the classes listed under “Class Required” using the interface in Table 1. You are not allowed to make changes to the files you downloaded in Requirement 1 above. You may use any Integrated Development Environment (IDE), such as Eclipse, Sublime, etc. You may also choose not to use an IDE.

This interface requires two functions: `search(...)` and `searchName(...)`. The `search(...)` function finds a target in an integer array and returns the index of the target if it exists, -1 if the target does not exist. The `searchName(...)` function returns the name of the search (eg. for LinearSearch.java, the `searchName(...)` function should return the string “`Linear search`”.

Class Required	Interface
`BinaryIterativeSearch.java`	`Practice03Search.java`
`BinaryRecursiveSearch.java`	`Practice03Search.java`
`LinearSearch.java`	`Practice03Search.java`

Table 1: Required classes, interfaces and functions

Requirement 3: Test your implementation by compiling and running with several values for arguments.

In order, the arguments are:

1) The size of the array to fill
2) The number of searches to perform
3) The type of search algorithm to use.

For example, the following (command line) runs the linear search 50,000 times on an array of 100,000 items:

```
java Practice03Test 100000 50000 linear
```

For the above, the output for a correct implementation may look like the following:

```
Array size  = 100000
Searches    = 50000
Search type = linear
Linear Search: total search time = 1485ms. Average search time =
0.01485ms.
```

Focus specifically on the sizes of the array being searched, which is the first argument. (What trends, if any, do you see when you vary the size of this parameter?)

Grading

This assignment is formally graded only on correctness and style. Each of the three required classes is worth 25% toward your grade, and your style (variable and class names, etc.) constitute the remaining 25% of the grade for this assignment.

Submission

Use GitHub to check in the code to complete the assignment. Once your implementation is on GitHub, submit the URL for your GitHub repository on Canvas.