Testing the relative speeds of the linear and recursive binary search algorithms

Homework #1

By

John Robertson

CS 303 Algorithms and Data Structures

September 1, 2019

1. **Problem Specification**

This assignment was supposed to implement the linear and recursive binary search algorithms and then measure their relative speeds on increasingly large arrays.

2. **Program Design**

This program has a file with the functions implementing the algorithms and a driver file to test the functions. The binary search implementation does not work, as my understanding of what the base case is supposed to be is muddled, as is how I’m supposed to retain the index.

The following steps were required to develop this program:

1. Implement linear search and recursive binary search as the lin\_search() and bin\_search() functions, respectively.
2. Import the above into the driver.
3. Get the target from the input file.
4. Generate the list to search
5. Search with both functions and measure their speeds.

The default Python sort() method was used to sort the inputs to binary search.

3. **Testing Plan**

The first thing that had to be done was test if the functions work by comparing them with Python’s built-in index() function. These tests are available as comments at the bottom of the driver. The next thing was to test the speeds on large inputs. The parameters for the test cases are defined in the assignment; since both the search space and the target are random, the only edge case that needs to be tested for (not finding an item) is tested for. bin\_search() was never supposed to account for the list being unsorted, as that would slow it unnecessarily, so the sorting is done at the beginning of the tests

4. **Test Cases**

Only arrays size 2^4 through 2^25 of random integers were supposed tested, as that is all the assignment asks for.

5. **Analysis and Conclusions**

Since my binary search function does not work, I cannot plot the results. I can only go by what Dr. Unan said when he “gave away the ending.”

Recursive binary search is supposed to be faster than linear search. This is because, by halving the search space, binary search is O(log(n)), where linear, which can and likely will travel most of the length of a given array, is O(n). Linear search is far easier to implement than binary search; sometimes the simplest way to do something isn’t the best. Naïve solutions should only be used to get **a** solution; they shouldn’t be the go-to tool.

6. **References**

The following references were used:

Dr. Unan’s CS303\_01\_Overview and CS303\_02\_Insertion\_Sort slides

https://stackoverflow.com/questions/4606919/in-python-try-until-no-error

https://stackoverflow.com/questions/8420143/valueerror-could-not-convert-string-to-float-id

https://stackoverflow.com/questions/6429638/how-to-split-a-string-of-space-separated-numbers-into-integers

https://stackoverflow.com/questions/12201928/python-open-gives-ioerror-errno-2-no-such-file-or-directory