Alejandra Reyes

3/25/2021

Homework 1 Question 2

Test whether a recursive, iterative or linked-type binary search is faster by testing it on

arrays of sizes 1 million and 10 million with arrays that are filled with random numbers.

You will need to either fill it in a “sorted way” or sort it before doing the binary search.

Program Description:

This program finds the average runtime over a specified number of iterations of iterative and recursive binary search on arrays, and iterative binary search on a linked list. The program considers the worst-case scenario where the value we are searching for is not found. A modified timing template was used to find the average runtime of a function in nanoseconds, over a specified number of iterations of the function. The program includes several templated functions that are used to fill and search arrays, a templated Node and List class, and templated member functions of the List class to fill and search lists.

Main calls `void fillArray<T>(array, size)` to fill an array of a certain size with random numbers, and uses std::sort to sort the array, as binary search requires that the array be sorted. The fillArray function uses a for loop to iterate through the array and assigns random values using std::rand().

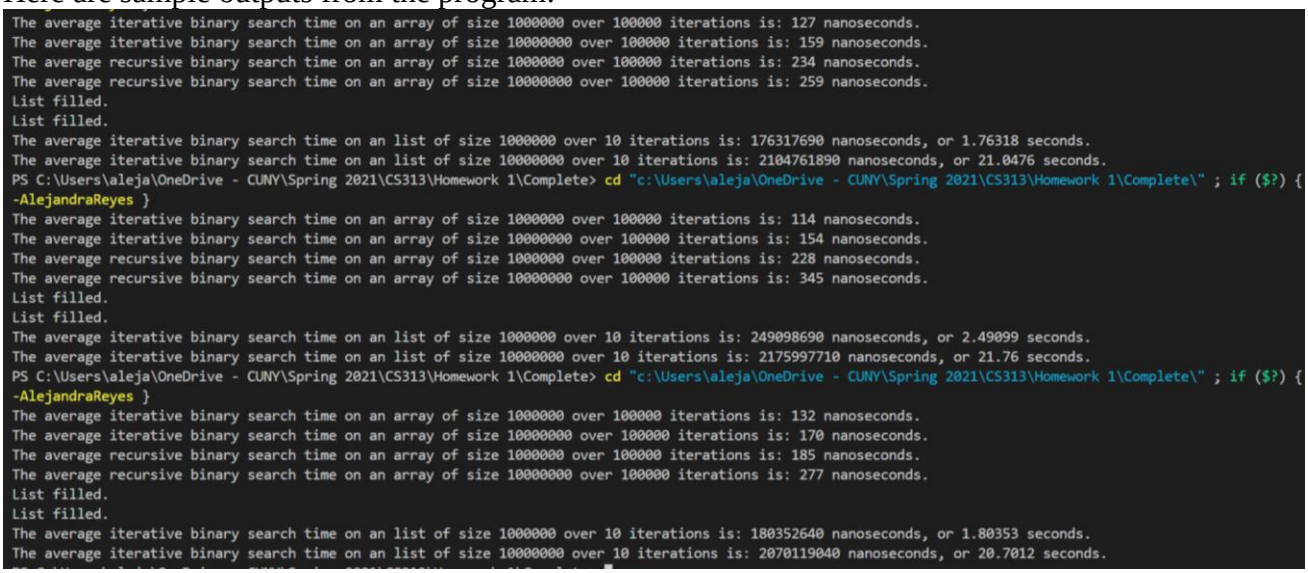
When the timing function is run with the iterative binary search function as an input, iterative binary search is performed on the same array over the set number of iterations. In each iteration, the function finds the middle value in the array, and enters a while loop. The while loop runs until the entire array has been searched, and it divides the search space in half each time it runs by deciding whether to search the left half or the right half of the current search space. To decide which half of the search space to check, the array compares the middle value of the search space with the value being searched for. If the value being searched for is greater than the middle value, the right half of the current search space becomes the new search space, and a new middle value is calculated to compare the target with. The loop runs until the target is found, or until we have searched the entire array.

When the timing function is run with the recursive binary search function as input, recursive binary search is performed on the same array over the specified number of iterations. In each iteration of recursive binary search, the middle value in the array is compared with the target. If the middle value equals the target, we return true. If we need to search the right half of the array, we call recursiveBinarySearch with parameters to indicate we are searching from the middle element to the last element. Otherwise if we are searching the left half of the array, we call recursiveBinarySearch with parameters to indicate we are searching from the first element to the middle element of the list.

In order to perform binary search on a list, we must first have a sorted list. A sorted list can be created by inserting values from a sorted array into an empty list, backwards. Once we have a filled list, iterative binary search is called. Iterative binary search on a list works similarly to an array, except that we need to use a function to retrieve the value of a node at a certain

position in the list. To compare the middle value of a list with the target, we call the member function get(int index) to get the value of the middle element in the list.

Here are sample outputs from the program:



Problems Encountered:

- The linked list had to be filled using a sorted array because lists do not provide random access to elements, meaning that to get to a certain position in the list, we must traverse through all the other nodes that came before.
- Binary search on a linked list takes a long time, so it was necessary to time binary search on a list over fewer iterations than on an array.
- When calculating the middle of an array or list, we cannot simply use mid = (right + left) / 2 because may cause overflow. We can prevent overflow by instead using mid = (left + (right-left)) / 2.

Conclusion:

It is clear from the results that iterative binary search on an array is about twice as fast as recursive binary search on an array. Increasing the size of the array does not significantly affect the iterative search time, however recursive search takes longer as the size of the array increases. On the other hand, iterative binary search on a linked list takes on average at least 20 seconds. The reason that iterative binary search is so slow on a linked list is that each time we want to check the middle value, the get() function traverses the list beginning from the head until it reaches the middle node. As the list gets larger, the time it takes to traverse the list increases.