Adam Wexler

COP3530

Documentation: Search.cpp

Code Organization:

Significance of Variables and Methods:
1. Public variables:

- int size1—variables defines elements that are being searched through. As the input values are concerned this variable is known as int n.
- int size2—variable defines the number of elements that are in the list of numbers that are being searched for. When dealing with the input values, this variable is known as int s. This are mostly known as keys.
- int A []--This is the array that covers the values that will be searched, which includes the size1 variable.
- int B []--This is the array that covers the keys, which are the values are being searched for. This array will include the variable size2.
- void linearSearch(int A[], int B[], int size1, int size2)—Function that emulates the general Linear Search Algorithm.
- void binarySearch(int A[], int B[], int size1, int size2)—Function that runs the general Binary Search Algorithm.

Specific Significance of Methods:
1. void linearSearch(int A[], int B[], int size1, int size2)
This function runs the linear search algorithm. It is set up with double for loops, in which the inner for loop will cycle through the integers being searched and the outer for loop cycles through the keys or the items that the program is attempting to find. The method begins a clock start in order to generate and check the speed of the algorithm, then the method continues with a double for loop and individually checks one by one integer value until an equal value has been found. This function doesn't require a sorting algorithm because each value in the array is being evaluated. An if statement is created in order to check if the values being searched and the values that are being searched for are equivalent. If these values are not found, then the program with print a “No” string to the screen. The program will exit out of both for loops when all the keys have been searched for and all values that are being searched are finished. The program will end with the clock stopping, the time is converted to seconds, and then prints to the screen. This is the end of the Linear Search algorithm. The general time complexity of this such algorithm is quadratic, since you are running N elements by S keys (i.e. O(N^2)).

2. void binarySearch(int A[], int B[], int size1, int size2)
This function runs a binary search algorithm. This method is created a little different, than the linear search was set up. The main difference is that instead of running through the entire array, the array is cut in half and the search will be narrowed down. The narrowing process will go something like this: if the key value is larger than the middle values in the array being searched, then the lower half of the array will be excluded. However, if the opposite where to occur, then the upper bound will be excluded. If neither of these options are the case, then the value that being searched for should be the middle value, unless these values are not equal to each other. The algorithm that was implement in the cpp file starts a sorting algorithm that is laid out with the library class called #include <algorithm>. After the array is sorted, then the program will start a clock to gauge the amount of time that the program will

take to run a Binary Search. After the sorting is done and the clock begins to tick, the program runs a for loop to run all the keys that will need to be searched for in the array that is being searched. The while loop is used instead of a for loop in order to consistently cut the array size in half. There are three values that are initialized, which are the int start, the int end, and the int searching. The searching value is consistently cut in half. During each iteration of the while loop, if the key is larger than the value being searched, then the searching value will increment by 1 and will be set to the start variable. If the other case occurs, then the program will set the end variable to searching minus 1. Counters are used in this function and in the linear search, so that an extra "No" will not be printed to the screen when there is indeed a "Yes." The clock timer will stop and evaluate the time. This will end the Binary Search function. This function is evaluated to O(S*lg N), but since S is constant in this exercise we have a logarithmic function.

Test Runs:

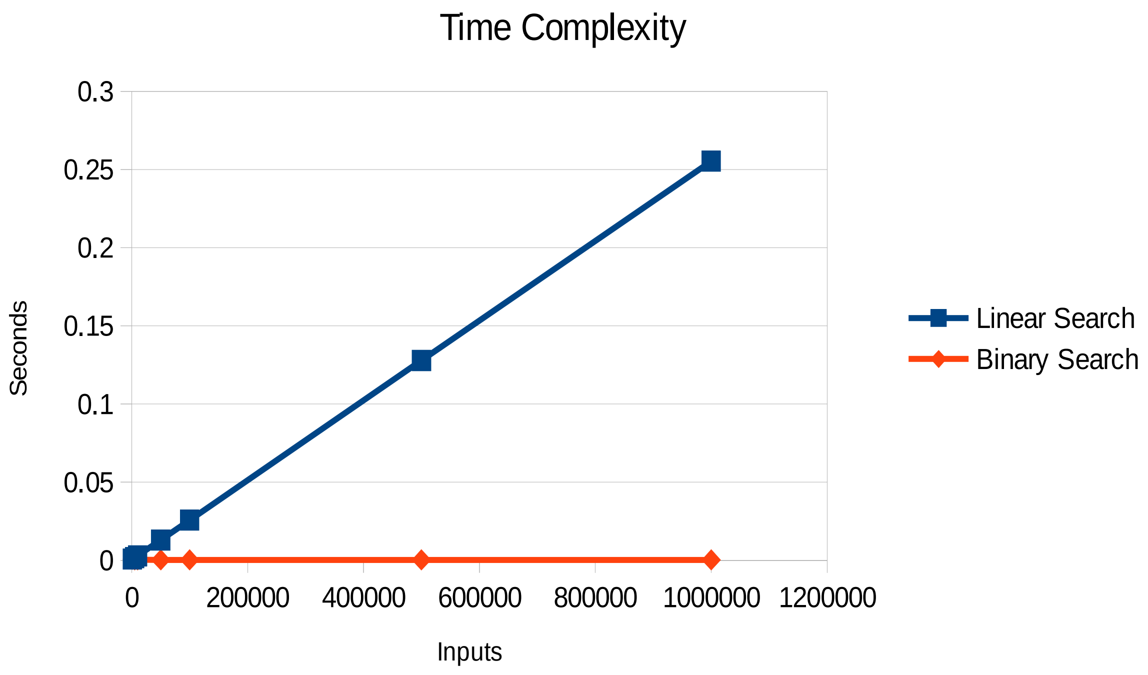
Trial 1:

Input Values	Linear Search Time	Binary Search Time
N=1,000 & S= 100	.000862 sec	.00724 sec
N=10,000 & S= 100	.002837 sec	.000371 sec
N=100,000 & S= 100	.025693 sec	.000301 sec
N=1,000,000 & S= 100	.254743 sec	.000311 sec

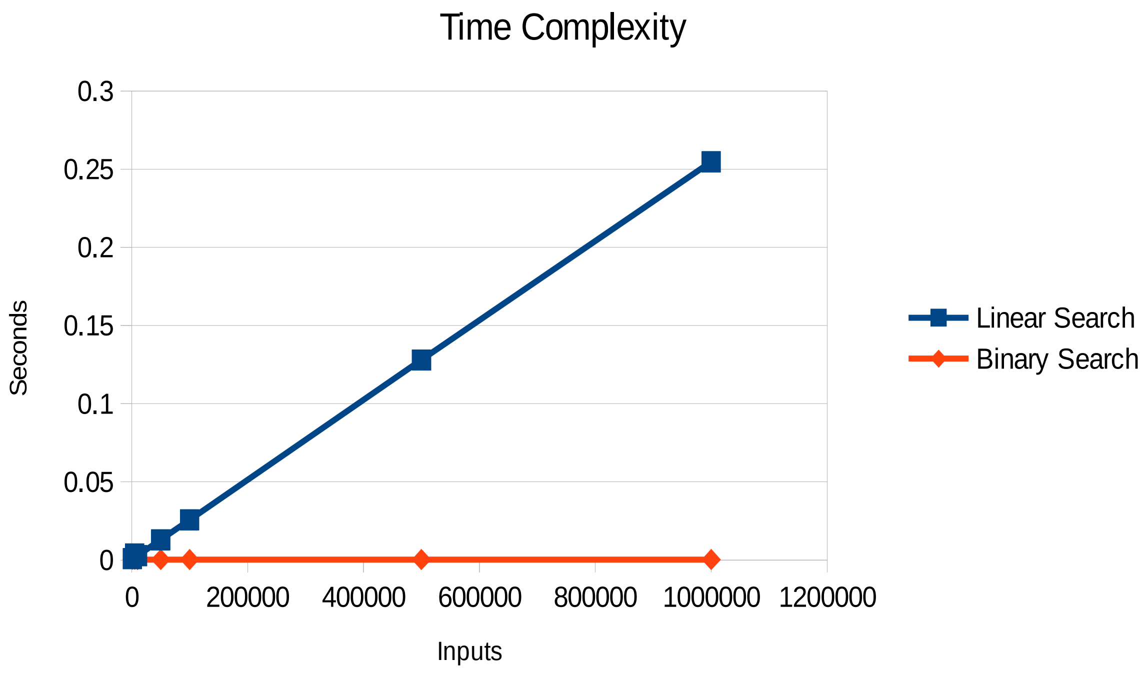
Trial 2:

Input Values	Linear Search Time	Binary Search Time
N=1,000 & S= 100	.000857 sec	.000401 sec
N=10,000 & S= 100	.002837 sec	.000298 sec
N=100,000 & S= 100	.025693 sec	.000301 sec
N=1,000,000 & S= 100	.254743 sec	.000311 sec

Trial 1



Trial 2


Time Complexity
Seconds
Inputs
Linear Search
Binary Search