Lab: 3

Execution Time, Linear Search and Binary Search Implementation

Due Date: January 21, 2021

Lab Session:Virtual/ Remote (A3, 16603)

Adrian Gomez
ID: 20119988

Linear Search Algorithm Simulation Result (n = 10 terms)

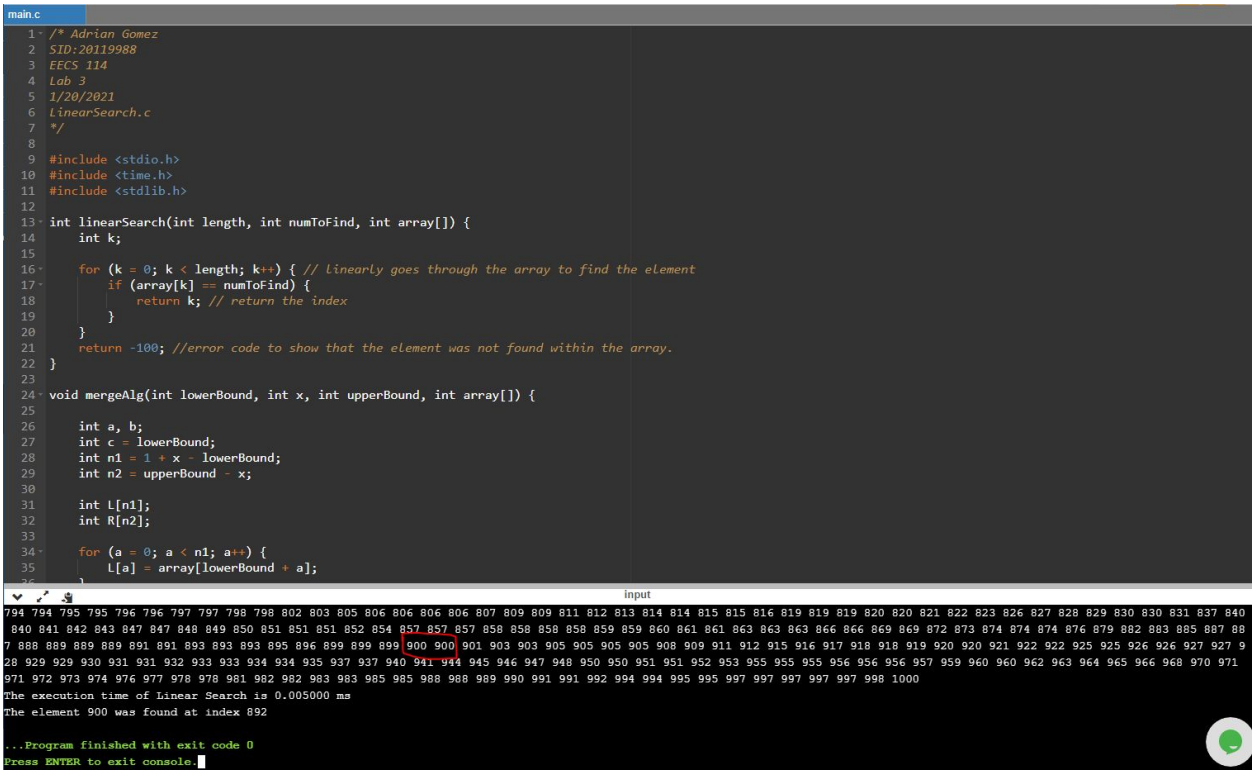
main.c

```c
/* Adrian Gomez
SID:20119988
EECS 114
Lab 3
1/20/2021
LinearSearch.c
*/

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int linearSearch(int length, int numToFind, int array[]) {
    int k;

    for (k = 0; k < length; k++) { // linearly goes through the array to find the element
        if (array[k] == numToFind) {
            return k; // return the index
        }
    }
    return -100; //error code to show that the element was not found within the array.
}

void mergeAlg(int lowerBound, int x, int upperBound, int array[]) {

    int a, b;
    int c = lowerBound;
    int n1 = 1 + x - lowerBound;
    int n2 = upperBound - x;

    int L[n1];
    int R[n2];

    for (a = 0; a < n1; a++) {
        L[a] = array[lowerBound + a];
```

input

```
4 7 8 6 4 6 7 3 10 2
2 3 4 4 6 6 7 7 8 10
The execution time of Linear Search is 0.002000 ms
The element 10 was found at index 9

...Program finished with exit code 0
Press ENTER to exit console.
```

Linear Search Algorithm Simulation Result (n = 1000 terms)



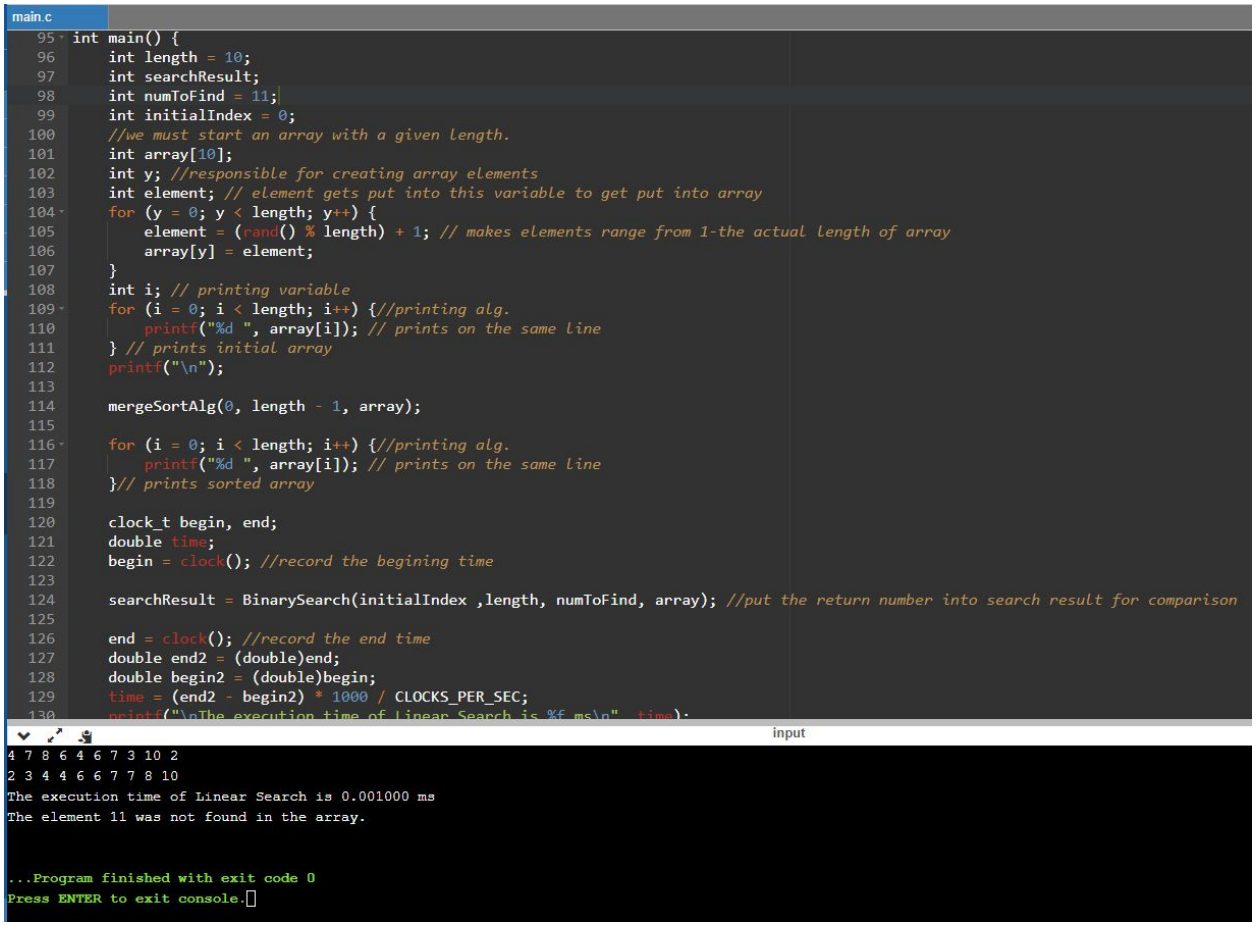
main.c

```c
/* Adrian Gomez
SID:20119988
EECS 114
Lab 3
1/20/2021
LinearSearch.c
*/

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int linearSearch(int length, int numToFind, int array[]) {
    int k;

    for (k = 0; k < length; k++) { // linearly goes through the array to find the element
        if (array[k] == numToFind) {
            return k; // return the index
        }
    }
    return -100; //error code to show that the element was not found within the array.
}

void mergeAlg(int lowerBound, int x, int upperBound, int array[]) {

    int a, b;
    int c = lowerBound;
    int n1 = 1 + x - lowerBound;
    int n2 = upperBound - x;

    int L[n1];
    int R[n2];

    for (a = 0; a < n1; a++) {
        L[a] = array[lowerBound + a];
```

input

```
794 794 795 795 796 796 797 797 798 798 802 803 805 806 806 806 806 807 809 809 811 812 813 814 814 815 815 816 819 819 819 820 820 821 822 823 826 827 828 829 830 830 831 837 840
 840 841 842 843 847 847 848 849 850 851 851 851 852 854 857 857 857 858 858 858 858 859 859 860 861 861 863 863 863 866 866 869 869 872 873 874 874 874 876 879 882 883 885 887 88
7 888 889 889 889 891 891 893 893 893 895 896 899 899 899 900 900 901 903 903 905 905 905 905 908 909 911 912 915 916 917 918 918 919 920 920 921 922 922 925 925 926 926 927 927 9
28 929 929 930 931 931 932 933 933 934 934 935 937 937 940 941 944 945 946 947 948 950 950 951 951 952 953 955 955 955 956 956 956 957 959 960 960 962 963 964 965 966 968 970 971
971 972 973 974 976 977 978 978 981 982 982 983 983 985 985 988 988 989 990 991 991 992 994 994 995 995 997 997 997 997 997 998 1000
The execution time of Linear Search is 0.005000 ms
The element 900 was found at index 892

...Program finished with exit code 0
Press ENTER to exit console.
```

Binary Search Algorithm Simulation Result (n = 10 terms)



```
main.c
int main() {
    int length = 10;
    int searchResult;
    int numToFind = 11;
    int initialIndex = 0;
    //we must start an array with a given length.
    int array[10];
    int y; //responsible for creating array elements
    int element; // element gets put into this variable to get put into array
    for (y = 0; y < length; y++) {
        element = (rand() % length) + 1; // makes elements range from 1-the actual length of array
        array[y] = element;
    }
    int i; // printing variable
    for (i = 0; i < length; i++) {//printing alg.
        printf("%d ", array[i]); // prints on the same line
    } // prints initial array
    printf("\n");

    mergeSortAlg(0, length - 1, array);

    for (i = 0; i < length; i++) {//printing alg.
        printf("%d ", array[i]); // prints on the same line
    }// prints sorted array

    clock_t begin, end;
    double time;
    begin = clock(); //record the begining time

    searchResult = BinarySearch(initialIndex ,length, numToFind, array); //put the return number into search result for comparison

    end = clock(); //record the end time
    double end2 = (double)end;
    double begin2 = (double)begin;
    time = (end2 - begin2) * 1000 / CLOCKS_PER_SEC;
    printf("\nThe execution time of Linear Search is %f ms\n", time);
input
4 7 8 6 4 6 7 3 10 2
2 3 4 4 6 6 7 7 8 10
The execution time of Linear Search is 0.001000 ms
The element 11 was not found in the array.


...Program finished with exit code 0
Press ENTER to exit console.
```

Binary Search Algorithm Simulation Result (n = 1000 terms)

main.c

```c
/* Adrian Gomez
SID:20119988
EECS 114
Lab 3
1/20/2021
BinarySearch.c
*/

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int BinarySearch(int initialLength,int length, int numToFind, int array[]) {

    if (length > initialLength) {

        int middle =(length - initialLength) / 2;
        middle += initialLength; // to find the middle of the array

        if (array[middle] == numToFind) {
            return middle;
        }
        else if (numToFind < array[middle] ) {
            return BinarySearch(initialLength, middle - 1, numToFind, array); // if not middle search lower
        }
        else {
            return BinarySearch(middle + 1, length, numToFind, array); // if its not middle or in the left side of the array look in top part
        }

    }

    return -100;
}

void mergeAlg(int lowerBound, int x, int upperBound, int array[]) {
```

input

```
794 794 795 795 796 796 797 797 798 798 802 803 805 806 806 806 807 809 809 811 812 813 814 814 815 815 816 819 819 819 820 820 821 822 823 826 827 828 829 830 830 831 837 840
 840 841 842 843 847 847 848 849 850 851 851 851 852 854 857 857 857 858 858 858 858 859 859 860 861 861 863 863 863 866 866 869 869 872 873 874 874 874 876 879 882 883 885 887 88
7 888 889 889 889 891 891 893 893 893 895 896 899 899 899 900 900 901 903 903 905 905 905 905 908 909 911 912 915 916 917 918 918 919 920 920 921 922 922 925 925 926 926 927 927 9
28 929 929 930 931 931 932 933 933 934 934 935 937 937 940 941 944 945 946 947 948 950 950 951 951 952 953 955 955 955 956 956 956 957 959 960 960 962 963 964 965 966 968 970 971
971 972 973 974 976 977 978 978 981 982 982 983 983 985 985 988 988 989 990 991 991 992 994 994 995 995 997 997 997 997 997 998 1000
The execution time of Linear Search is 0.001000 ms
The element 900 was found at index 892

...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion

In conclusion, we can conclude that Binary Search is a faster search algorithm in comparison to Linear Search. From our intuition, we can have an assumption that Binary Search will be the faster algorithm since it will divide and conquer the array. On the other hand, the Linear Search would essentially brute force the array and go through all the elements to search for the element. We can see the difference between these two algorithms from the time complexities given from lecture. For instance, Binary Search has the execution time of $T(n) = T(n/2) + \theta(1) = \theta(lg\ n)$. On the other, Linear Search will have the time complexity of $T(n^2)$. As time gets larger, Linear Search will get larger faster than Binary Search, which is consistent with the simulation results provided above.