Activity No. 6.1	
Linear and Binary Search	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:9/11/25**
Section:CPEE21S4	**Date Submitted:9/11/25**
Name(s):Crishen Luper S.Pulgado	**Instructor:**

6. Output

1.) **What is a search tree in data structures?**
It is a data structure for organizing and storing data in a sorted data in a sorted matter. All of the characteristics of a binary tree are followed by a binary search tree, where each node has values in its right subtree and values less than the node in its left subtree. The hierarchal structure allows for efficient Searching, Insertion, and Deletions operations on the data stored in the three.

2.) **What are the Different types of search algorithm in data structures? Differentiate each type of search.**
The different types of search algorithm in data structure are the sequential searching and interval searching. The difference is that sequential searching operation traverses through each element of the data sequentially to look for the desired data. For this kind of search, the data does not have to be sorted. Unlike sequential searching, it requires the data to be in a sorted manner in the interval searching. It could be done by either dividing the data into multiple sub-parts or jumping through the indices to search for an element.

3.) **What operations / implementations can be performed using binary and linear search operations?**
In linear search, we can use the operations such as finding an element to check if a given value exists in array. Moreover, finding the first occurrence, returns the first position of the element, and finding the last occurrence which scans through the entire array and return the last index. Additionally, finding the maximum or minimum by checking each element and determining the largest or smallest value. In binary search, we can use the finding an element via locating the index of a value. Moreover, finding the fist and last occurrence which are the lower bound and upper bound. Additionally, we can use operations such as range searching which find the start and end range.

4.) **What are the advantages in using binary search tree as data structure?**
The advantages of using binary search tree as data structure is it is efficient in searching, the time complexity O(log n) for searching with a self-balancing BST. Also, Finding the next or previous element is simple because the elements are kept in a sorted order. Moreover, elements can be added or removed efficiently and a logarithmic height is maintained via balanced BSTs, guaranteeing effective operations. In doubly ended priority queue, we can maintain both maximum and minimum efficiently.

5.) **Give an example program using binary search and Linear search.**
Linear search:

```
#include <iostream>
int linearSearch(int arr[], int n, int target) {
    for (int i = 0; i < n; ++i) {
        if (arr[i] == target) {
            return i; // Element found at index i
        }
    }
    return -1; // Element not found
}
int main() {
    int arr[] = {10, 25, 30, 45, 50, 60};
    int n = sizeof(arr) / sizeof(arr[0]);
    int target = 45;

    int result = linearSearch(arr, n, target);
    if (result != -1) {
```

```
        std::cout << "Element found at index: " << result << std::endl;
    } else {
        std::cout << "Element not found." << std::endl;
    }
    return 0;
}
```

Binary search:

```
#include <iostream>
int binarySearch(int arr[], int left, int right, int target) {
    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == target) {
            return mid; // Element found at index mid
        }

        if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1; // Element not found
}
int main() {
    int arr[] = {10, 25, 30, 45, 50, 60};
    int n = sizeof(arr) / sizeof(arr[0]);
    int target = 45;

    int result = binarySearch(arr, 0, n - 1, target);
    if (result != -1) {
        std::cout << "Element found at index: " << result << std::endl;
    } else {
        std::cout << "Element not found." << std::endl;
    }
    return 0;
}
```

References:

Data structures - searching algorithms. (n.d.). https://www.tutorialspoint.com/data_structures_algorithms/searching_algorithms.htm

GeeksforGeeks. (2025, July 15). *Introduction to binary search tree*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/introduction-to-binary-search-tree/

GeeksforGeeks. (2025a, July 2). *Linear Search vs Binary Search*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/linear-search-vs-binary-search/

GeeksforGeeks. (2025c, July 23). *Applications, Advantages and Disadvantages of Binary Search Tree*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/applications-advantages-and-disadvantages-of-binary-search-tree/

Krishna, A. (2024, August 14). *Search algorithms – linear search and binary search code implementation and complexity analysis*. freeCodeCamp.org. https://www.freecodecamp.org/news/search-algorithms-linear-and-binary-search-explained/

Loner, L. (n.d.). *Linear and Binary Searching Algorithm using C++ | DSA*. Learn Loner. https://learnloner.com/linear-and-binary-searching-algorithm-using-c/

7. Supplementary Activity

8. Conclusion

9. Assessment Rubric