

Activity No. <n>

<Seatwork 6.1 Linear and Binary Search >

Course Code: CPE010	Program: Computer Engineering
Course Title: Data Structures and Algorithms	Date Performed: 09/11/25
Section: CPE 010-CPE21S4	Date Submitted: 09/11/25
Name(s): Kerwin Jan B. Catungal	Instructor: Jimlord Quejado

6. Output

Answer the following questions:

1. What is a search tree in data structures?

A Binary Search Tree (BST) is a type of binary tree data structure in which each node contains a unique key and satisfies a specific ordering property and also organizing data like a tree structure so we can quickly find, insert, or delete elements.

2. What are the Different types of search algorithm in data structures? Differentiate each type of search.

The two main types of search are linear search, which checks items one by one, and binary search, which repeatedly divides the data in half to find the target faster

3. What operations / implementations can be performed using binary and linear search operations?

Linear and binary search can be used to look for specific elements, check if something exists, or get the position of an item in a list or array.

4. What are the advantages in using binary search tree as data structure?

A binary search tree is good because it keeps data sorted, allows faster searching compared to linear search, and makes insertions or deletions more efficient.

5. Give an example program using binary search and Linear search.

LINEAR SEARCH:

```
1 #include <iostream>
2 using namespace std;
3
4 // Function to perform Linear Search
5 int linearSearch(int arr[], int size, int target) {
6     for(int i = 0; i < size; i++) {
7         if(arr[i] == target)
8             return i; // Target found at index i
9     }
10    return -1; // Target not found
11 }
12
13 int main() {
14     int data[] = {34, 78, 12, 56, 89, 23};
15     int size = sizeof(data)/sizeof(data[0]);
16     int target = 56;
17     int result = linearSearch(data, size, target);
18     if(result != -1)
19         cout << "Element found at index " << result << endl;
20     else
21         cout << "Element not found in the array." << endl;
22     return 0;
23 }
24
25 // Output:
26 // Element found at index 3
```

```
C:\Users\TIPQC\Documents
Element found at index 3
-----
Process exited after 0.01248 seconds with return
value 0
Press any key to continue . . . |
```

BINARY:

The screenshot shows a code editor window with a file named ex.cpp containing C++ code for binary search. The code defines a function binarySearch that takes an array, its size, and a target value. It uses a while loop to repeatedly calculate the middle index (mid) and compare the target value with the element at that index. If they match, it returns the index. If the target is greater, it ignores the left half; if smaller, it ignores the right half. The main function initializes an array with values {12, 23, 34, 56, 78, 89}, sets the size to 6, and the target to 56. It calls binarySearch and prints the result. The terminal window shows the output "Element found at index 3".

```
2  using namespace std;
3
4 // Function to perform Binary Search
5 int binarySearch(int arr[], int size, int target) {
6     int left = 0;
7     int right = size -1;
8     while(left <= right) {
9         int mid = left + (right - left) / 2; // Prevents potential overflow
10        // Check if target is present at mid
11        if(arr[mid] == target)
12            return mid;
13        // If target greater, ignore left half
14        if(arr[mid] < target)
15            left = mid + 1;
16        // If target is smaller, ignore right half
17        else
18            right = mid - 1;
19    }
20    return -1; // Target not found
21 }
22 int main() {
23     int data[] = {12, 23, 34, 56, 78, 89};
24     int size = sizeof(data)/sizeof(data[0]);
25     int target = 56;
26     int result = binarySearch(data, size, target);
27     if(result != -1)
28         cout << "Element found at index " << result << endl;
29     else
30         cout << "Element not found in the array." << endl;
31     return 0;
32 }
33
34 // Output:
35 // Element found at index 3
```

Paul E. Black, "search tree", in [Dictionary of Algorithms and Data Structures](#) [online], Paul E. Black, ed. 14 December 2005.

GeeksforGeeks. (n.d.). *Searching algorithms*. GeeksforGeeks. Retrieved September 11, 2025,

Singh, R. (2025, January 3). *Understanding searching algorithms in C++: Guide to linear and binary search*. Medium. <https://medium.com/@RobuRishabh/understanding-searching-algorithms-in-c-guide-to-linear-and-binary-search-1c0db52dba9e>

8. Conclusion

I understood that linear and binary search are useful ways to find data, with binary search being faster but needing sorted data. And also using a binary search tree helps keep data organized and makes searching and updating more efficient.

9. Assessment Rubric