Name : Deshmukh Mehmood Rehan

MIS No. : 612303050

SY Computer Science – Div 1

Searching Algorithms:

Linear Search:

Linear search is a simple search algorithm that finds the position of a target value within an array. It sequentially checks each element of the array until a match is found or the whole array has been searched.

Implementation

```
int linearSearch(int arr[], int size, int key) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == key) {
            return i;
        }
    }
    return -1;
}
```

Time Complexity Analysis:

1. Best Case: O(1)

In the best case, the element is found at the first index of the array. The time complexity is O(1) because the algorithm will only perform one comparison.

2. Worst Case: O(n)

In the worst case, the element is not present in the array, and the algorithm will have to compare the key with all elements of the array. The time complexity is O(n), where n is the size of the array.

3. Average Case: O(n)

The averge case time complexity can be approximated as O(n) as the element can be present at any position in the array.

Calculation for average case time complexity:

```
T(n) = 1/n * (1 + 2 + 3 + ... + n)
T(n) = 1/n * n(n+1)/2
T(n) = (n+1)/2
T(n) = O(n)
```

Testing

The linear search algorithm was tested on arrays of different sizes with a randomly chosen key. The time taken for each test case was recorded.

Array Size	Time Taken (ms)
5	5.141086
10	6.790809
20	4.561740
100	2.807038
500	4.484241
1000	7.722760
5000	7.390687
10000	4.397700
20000	9.926198
50000	13.813592

Insights

The linear search algorithm has a time complexity of O(n) in the worst case, where n is the size of the array. The time taken for the linear search increases linearly with the size of the array. For larger arrays, the time taken for linear search can be significant.

Binary Search

Binary search is a search algorithm that finds the position of a target value within a sorted array. It compares the target value to the middle element of the array and eliminates half of the remaining elements each time.

Implementation

```c
int binarySearch(int arr[], int size, int key) {
    int left = 0, right = size - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == key) {
            return mid;
```

```
        }
        if (arr[mid] > key) {
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return -1;
}
```

Time Complexity Analysis

1. Best Case: O(1)

In the best case, the element is found at the middle of the array. The time complexity is O(1) because the algorithm will only perform one comparison.

2. Worst Case: O(log n)

In the worst case, the element is not present in the array, and the algorithm will keep dividing the array in half until the search space is empty. The time complexity is O(log n), where n is the size of the array.

```
T(n) = n/2^k
it will continue until n/2^k = 1
1 = n/2^k
2^k = n
k = log n
```

3. Average Case: O(log n)

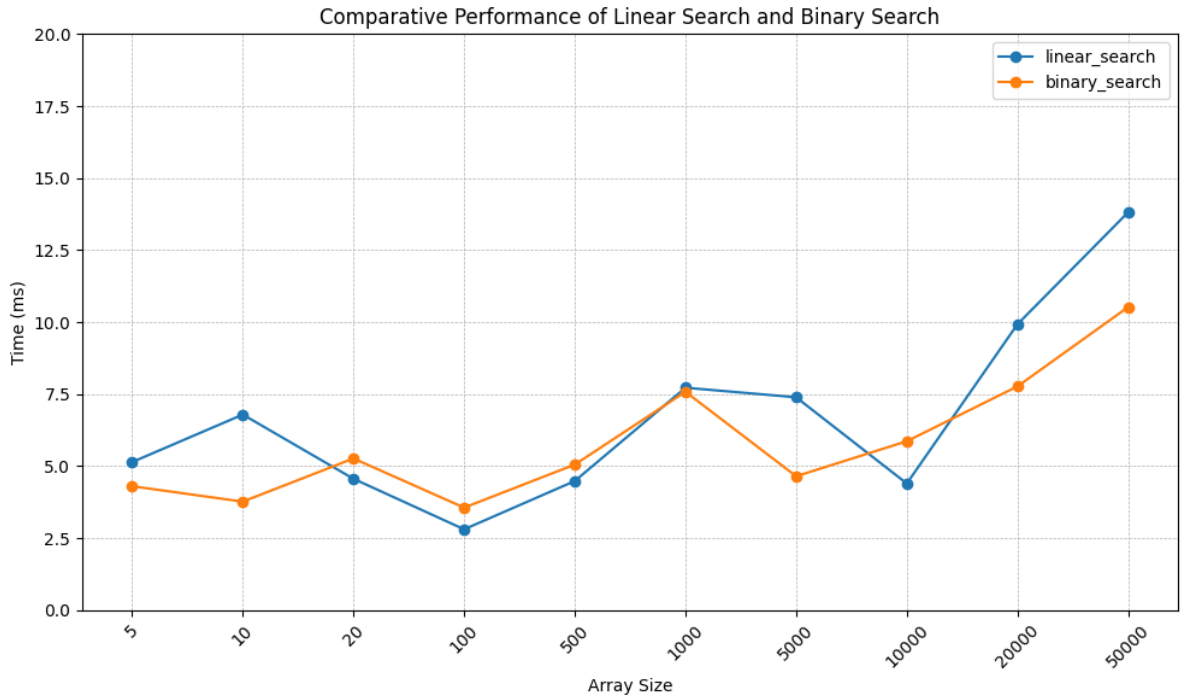
Testing

The binary search algorithm was tested on arrays of different sizes with a randomly chosen key. The time taken for each test case was recorded.

Array Size	Time Taken (ms)
5	4.303127
10	3.768525
20	5.266830
100	3.560187
500	5.055026
1000	7.582028
5000	4.649418
10000	5.864479
20000	7.769131
50000	10.522539

Comparison

The binary search algorithm has a time complexity of O(log n) in the worst case, where n is the size of the array. Binary search is preferable over linear search when the data is sorted because it has a lower time complexity. For larger arrays, binary search is significantly faster than linear search.



Conclusion

In conclusion, linear search and binary search are two fundamental searching algorithms with different time complexities. Linear search has a time complexity of O(n) in the worst case, while binary search has a time complexity of O(log n) in the worst case. Binary search is preferable over linear search when the data is sorted because of its lower time complexity. The choice of algorithm depends on the size of the data and whether it is sorted or unsorted.

Questions for deeper understanding

Why is binary search more efficient than linear search on sorted data?

Binary search eliminates half of the remaining elements in each step, resulting in a time complexity of O(log n) in the worst case. In contrast, linear search has a time complexity of O(n) in the worst case because it sequentially checks each element of the array. Therefore, binary search is more efficient than linear search on sorted data.