Title: Exploring Time Complexity: Linear Search and Binary Search Algorithms

Introduction: In this report, we delve into an examination of the time complexity of two pivotal search algorithms: linear search and binary search. These algorithms serve as foundational tools in the field of data processing and retrieval. Our primary objective is to design, implement, and assess these algorithms while meticulously tracking the number of comparisons they execute, providing valuable insights into their operational efficiency.

To achieve this goal, we employ two distinct methods, linear_search and binary_search, which operate on a data structure 'a' of varying sizes ('n'). We conduct a series of experiments utilizing randomly generated data stored in 'search_i.txt' files, where 'i' varies from 1 to 5, with 'n' values ranging from 10,000 to 50,000. This systematic evaluation aims to provide a comprehensive understanding of how these search algorithms perform across different problem sizes, shedding light on their practical utility and efficiency in real-world applications.

Program:

```
#include <iostream>
#include <fstream>
#include <chrono>
#include <string>
#include <cstdlib>
#include <algorithm>
using namespace std;
long long count1 = 1, count2 = 0;
long long binarySearch(long long size, long long target)
{
    long long left = 0;
    long long right = size - 1;
    long long arr[size];
    string file = "search" + to_string(size) + ".txt";
    ifstream input(file);
    for (long long i = 0; i < size; i++)
        input >> arr[i];
    input.close();
    sort(arr, arr + size);
    while (left <= right)
    {
        count2++;
        long long mid = left + (right - left) / 2;
        if (arr[mid] == target)
            return mid;
        else if (arr[mid] < target)
            left = mid + 1;
        else if (arr[mid] > target)
            right = mid - 1;
```

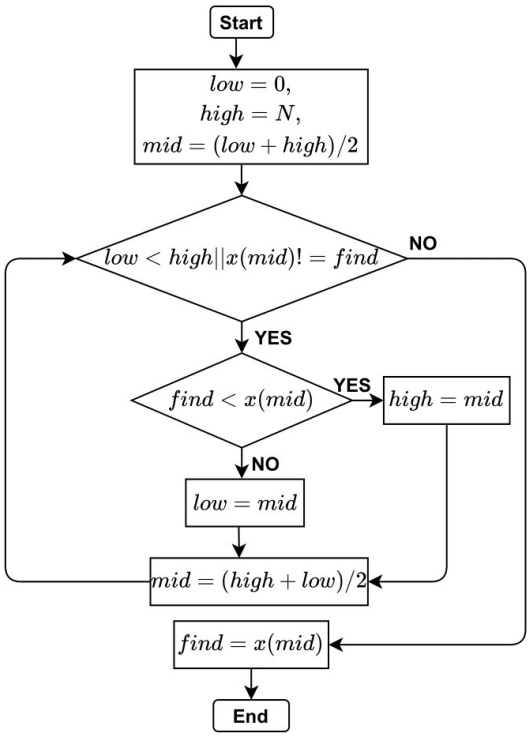
```
    }
    return -1;
}
long long linearSearch(long long size, long long target)
{
    string file = "search" + to_string(size) + ".txt";
    ifstream input(file);
    auto start = std::chrono::high_resolution_clock::now();
    for (long long i = 0; i < size; i++)
    {
        int a;
        input >> a;
        count1++;
        if (a == target)
            return i;
    }
    input.close();
    auto stop = std::chrono::high_resolution_clock::now();
    return -1;
}

int main()
{
    srand(time(0));
    system("cls");
    long long n, target;
    cout << "How many Numbers : ";
    cin >> n;
    string file = "search" + to_string(n) + ".txt";
    ofstream input(file);
    for (long long i = 0; i < n; i++)
        input << rand() << " ";
    input.close();
    cout << "Enter the target: ";
    cin >> target;
    long long result = linearSearch(n, target);
    long long result2 = binarySearch(n, target);
    if (result != -1 || result2 != -1)
    {
        cout << "Linear Search" << endl;
        cout << "Key: " << result << endl;
        cout << "Value: " << target << endl;
        cout << "Steps: " << count1 << endl;
        cout << "######################" << endl;
        cout << "Binary Search" << endl;
```

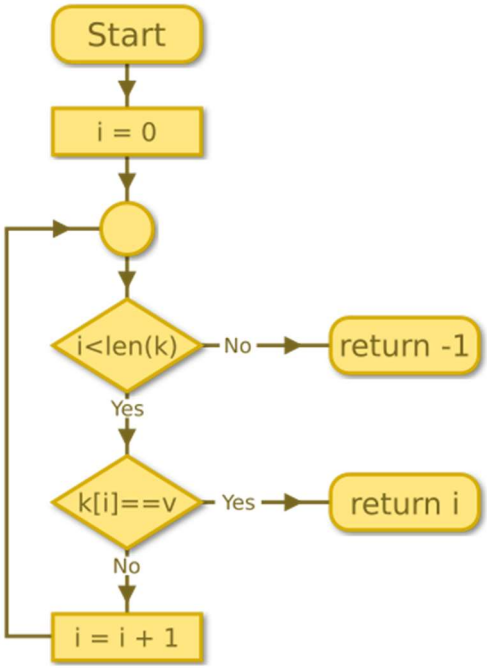
```
        cout << "Key: " << result2 << endl;
        cout << "Value: " << target << endl;
        cout << "Steps: " << count2 << endl;
    }
    else
    {
        cout << "Element not found in the array." << std::endl;
    }

    return 0;
}
```

Flow Chart:

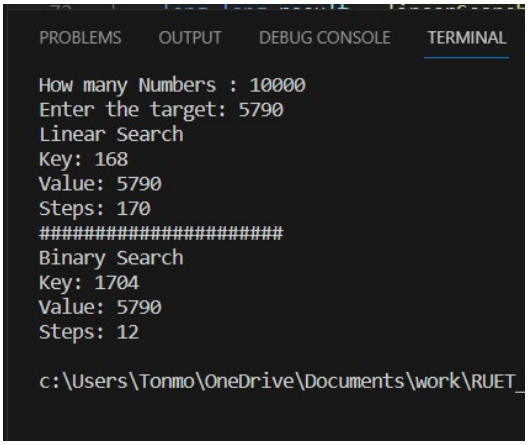


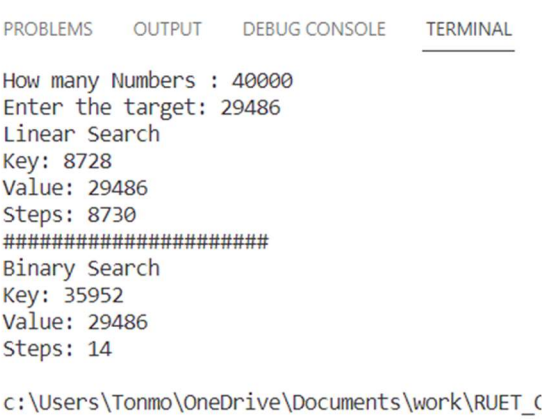
Binary Search

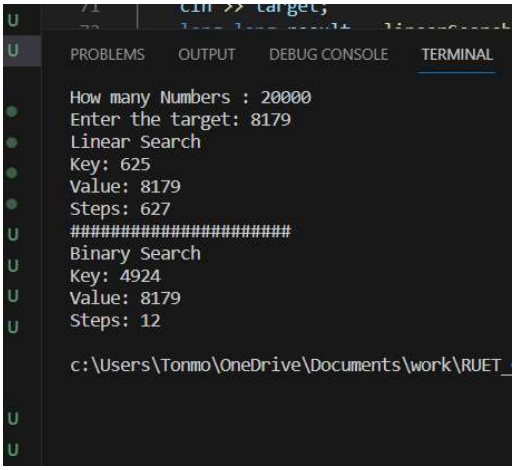


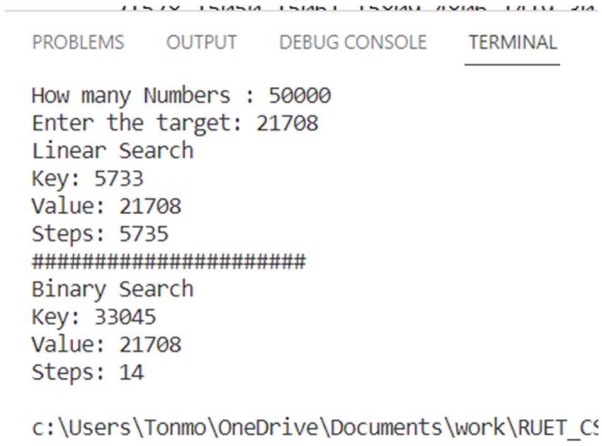
Binary Search

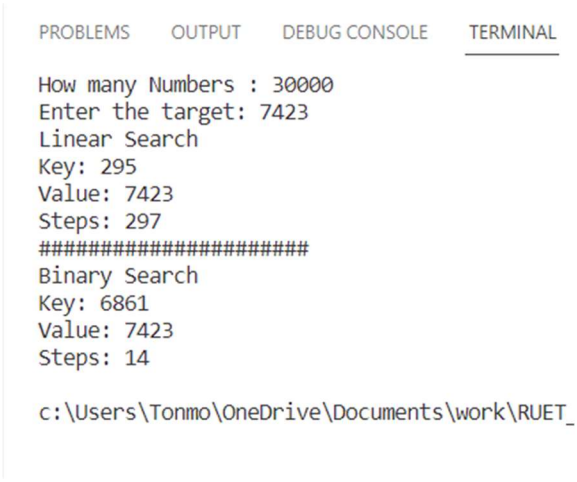
Input Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
How many Numbers : 10000
Enter the target: 5790
Linear Search
Key: 168
Value: 5790
Steps: 170
######################
Binary Search
Key: 1704
Value: 5790
Steps: 12
c:\Users\Tonmo\OneDrive\Documents\work\RUET_


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
How many Numbers : 40000
Enter the target: 29486
Linear Search
Key: 8728
Value: 29486
Steps: 8730
######################
Binary Search
Key: 35952
Value: 29486
Steps: 14
c:\Users\Tonmo\OneDrive\Documents\work\RUET_(


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
How many Numbers : 20000
Enter the target: 8179
Linear Search
Key: 625
Value: 8179
Steps: 627
######################
Binary Search
Key: 4924
Value: 8179
Steps: 12
c:\Users\Tonmo\OneDrive\Documents\work\RUET_


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
How many Numbers : 50000
Enter the target: 21708
Linear Search
Key: 5733
Value: 21708
Steps: 5735
######################
Binary Search
Key: 33045
Value: 21708
Steps: 14
c:\Users\Tonmo\OneDrive\Documents\work\RUET_C


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
How many Numbers : 30000
Enter the target: 7423
Linear Search
Key: 295
Value: 7423
Steps: 297
######################
Binary Search
Key: 6861
Value: 7423
Steps: 14
c:\Users\Tonmo\OneDrive\Documents\work\RUET_

Analysis of the complexity:

Conclusion: In conclusion, our exploration of linear search and binary search algorithms has revealed distinct strengths and weaknesses inherent to each approach. Linear search, while straightforward to implement, exhibits a linear time complexity, making it less efficient for large datasets. On the other hand, binary search, with its logarithmic time complexity, excels in scenarios with sorted data, showcasing superior performance for sizeable datasets. However, binary search necessitates a pre-sorted input, limiting its applicability in certain situations. Understanding the trade-offs between these two algorithms is essential for selecting the most appropriate search strategy based on the specific problem and dataset characteristics.