Practical 4

Aim:

Compare the performance of linear search and binary search on various inputs.

Code:

```c
/**
* Name: Ankit Verma
* Enroll No.: 190170116077
* Date: 9th July 2021
* Brief: Analysis of Sequential & Binary Search - Best & Worst Case
**/

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

int step_counts = 0;
int total_inputs = 10;

void reset_step_counts()
{
    step_counts = 0;
}

int sequential_search(int *nums, int n, int key)
{
    int i;
    ++step_counts;
    for (i = 0; i < n; ++i)
    {
        ++step_counts;
        if (nums[i] == key)
        {
            ++step_counts;
            return i;
        }
    }
    ++step_counts;
    return -1;
}

int binary_search(int *nums, int n, int key)
{
    int l, r, m;
```

```
    ++step_counts;
    l = 0;
    r = n - 1;

    while (++step_counts, l <= r)
    {
        ++step_counts;
        m = (l + r) / 2;

        if (++step_counts, nums[m] == key)
        {
            ++step_counts;
            return m;
        }
        else if (++step_counts, nums[m] > key)
        {
            ++step_counts;
            r = m - 1;
        }
        else
        {
            ++step_counts;
            l = m + 1;
        }
    }
    ++step_counts;
    return -1;
}

int *create_array(int n)
{
    int *array = NULL, i;
    if (n <= 0)
    {
        return NULL;
    }

    array = (int *)malloc(sizeof(int) * n);
    if (!array)
    {
        return NULL;
    }

    for (i = 0; i < n; ++i)
    {
        array[i] = i;
    }
```

```
    return array;
}

void testing_function(char test_type, char *test_title)
{
    int i, n;
    int *nums = NULL;

    if (test_type != 'b' && test_type != 'w')
    {
        return;
    }

    printf("%s\n\n", test_title);

    printf("Input\tLinear-Search\tBinary-Search\n");
    printf("---------------------------------------\n");

    for (i = 1; i <= total_inputs; ++i)
    {
        n = 1 << i;
        printf("%d\t", n);

        nums = create_array(n);
        reset_step_counts();

        sequential_search(nums, n, (test_type == 'b' ? nums[0] : -1));
        printf("%d\t\t", step_counts);

        reset_step_counts();
        binary_search(nums, n, (test_type == 'b' ? nums[(n - 1) / 2] :
n));
        printf("%d\n", step_counts);

        free(nums);
    }

    printf("---------------------------------------\n");
    printf("\n");
}

int main()
{
    testing_function('b', "Best Case Analysis");
    testing_function('w', "Worst Case Analysis");
    return 0;
}
```

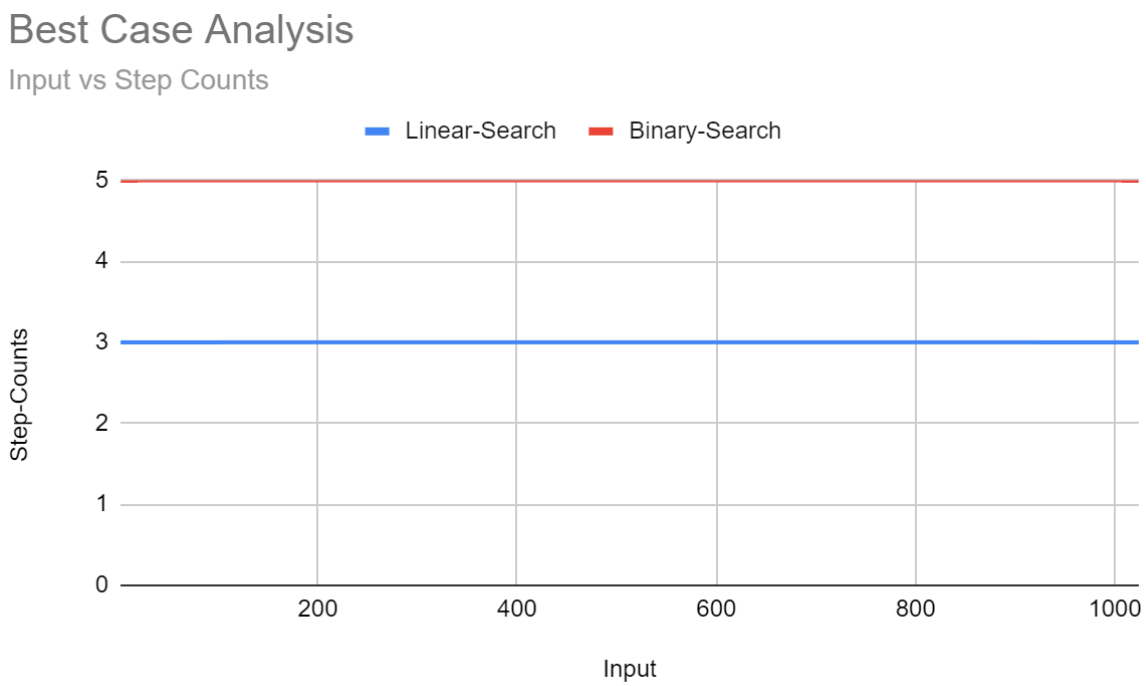
Best Case Analysis:

Screenshot:

```
PS D:\College\Sem 5\ADA\Practicals\Practical 4> & .\"searching_algorithms_a
nalysis.exe"
Best Case Analysis

Input   Linear-Search   Binary-Search
--------------------------------------
2       3               5
4       3               5
8       3               5
16      3               5
32      3               5
64      3               5
128     3               5
256     3               5
512     3               5
1024    3               5
--------------------------------------
```

Graph:



Conclusions:

We can clearly understand from the output step counts & graph, Sequential Search takes constant time for best case i.e. if key to be found is at index 0. So the time complexity of the Sequential Search is Θ(1) whereas Binary Search also takes constant time for best case i.e. if key is to be found is at middle index of the array. So the time complexity of Binary Search is Θ(1). Hence, we conclude both will take similar amount of time for their respective best cases.

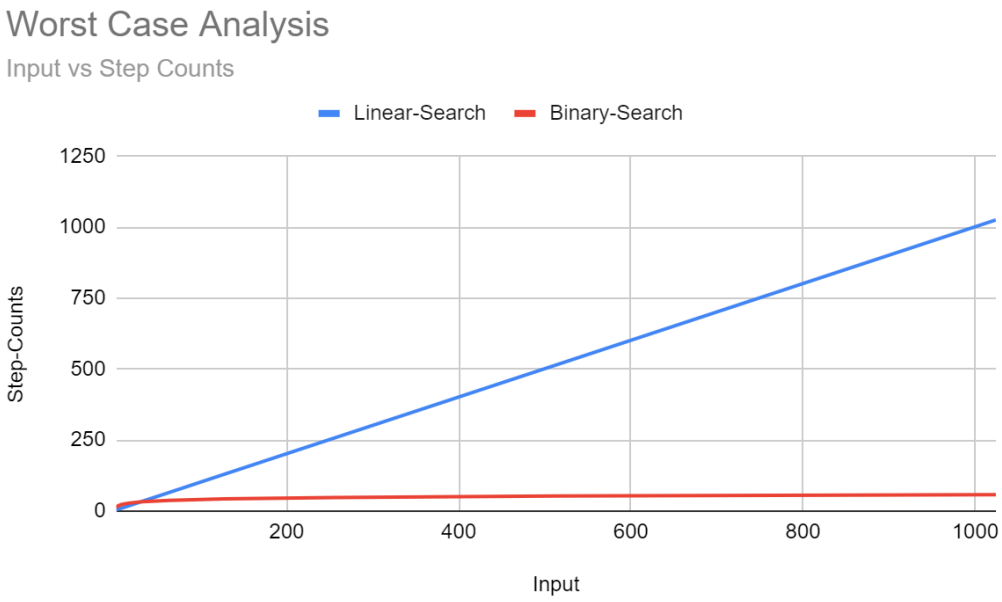
Worst Case Analysis:

Screenshot:

```
Worst Case Analysis

Input   Linear-Search   Binary-Search
--------------------------------------
2       4               13
4       6               18
8       10              23
16      18              28
32      34              33
64      66              38
128     130             43
256     258             48
512     514             53
1024    1026            58
--------------------------------------
```

Graph



Conclusions:

We can clearly understand from the output step counts & graph, Sequential Search takes linear time for worst case i.e. if key to be found is not found in the array. So the time complexity of the Sequential Search is Θ(n) whereas Binary Search takes logarithmic time for worst case i.e. if key to be found is not in the array and it should be at end of array. So the time complexity of the binary search is Θ(lgn). Hence, we conclude Binary Search performs much more better than Linear Search for worst case, given that array is sorted.

Overall Understanding:

For Unsorted Data:
We should use Linear Search as even for worst case it will work with Θ(n) time complexity but Binary Search won't work

For Sorted Data:
We should use Binary Search for its very good time complexity of Θ(lgn) even for worst case whereas Sequential Search will take Θ(n) time.