Lab 1: Implement and analyze iterative algorithms for Fibonacci term, GCD, and Linear Search.

Theory:

Algorithm for Fibonacci number

```
1. Start
2. Read n
3. If n == 1 or n == 2
        return 1
4. Else
   Set a = 1, b = 1, c = 0
   For(i=1; i ≤ n-2; i++)
        c = a + b
        a = b
        b = c
   return c;
5. Stop
```

Analysis: In the worst case, the for loop runs for at most n times, so the worst case time complexity is O (n). And, the algorithm occupies at most 5 cells of RAM model to hold the value of n, a, b, c and i, the worst case space complexity is O (1).

Algorithm for GCD

```
1. Start
2. Read a and b.
3. If b == 0
   Return a
4. Else
   While(b ≠ 0)
        r = a%b
        a = b
        b = r
   Return a
5. Stop
```

Analysis:

Time Complexity: O (n)

Space Complexity: O (1)

Algorithm for Linear Search

Let A[n] be an array where the number are stored.

```
1. Start.
2. Read key as value to be searched.
3. Set flag = 0
4. For(i=0; i < n; i++)
   If(A[i] == key)
           flag = 1
           Display “Search Successful. Element found at i position”
5. If(flag == 0)
   Display “Search Unsuccessful. Element not found”
6. Stop
```

Analysis:

Time Complexity: O (n)

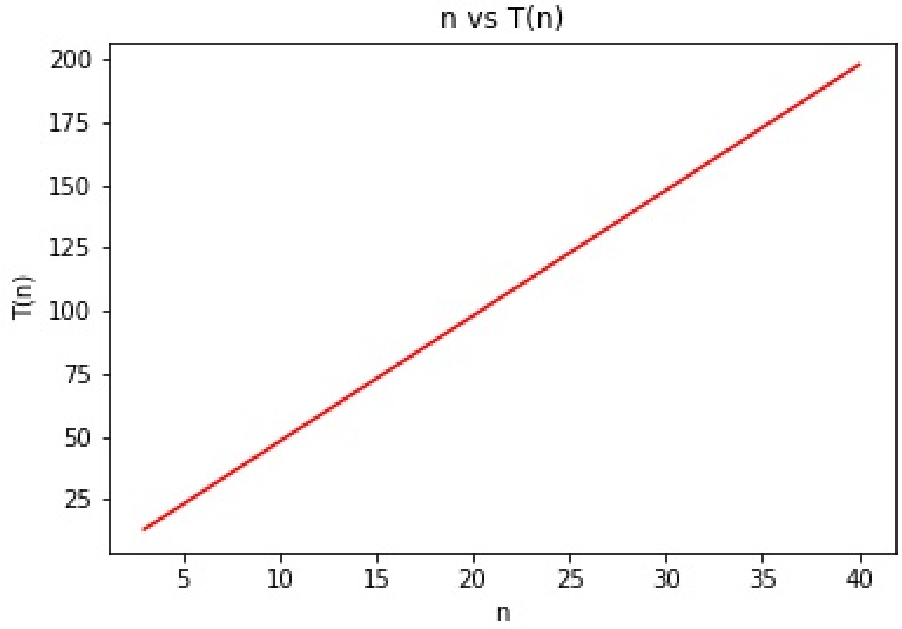
Space Complexity: O (n)

Source Code

Program for Fibonacci number

```
#include <iostream>
using namespace std;
int t = 0;
long int fibo(int n)
{
    t++;t++;t++;
    if(n == 0 || n==1)
    {
        t++;
        return 1;
    }
    else
    {
        int i;
        long int c = 0, a = 1, b = 1;t++;t++;t++;
        for(i=1,t++; i<= n-2; i++,t++)
        {
            t++;
            c = a + b;t++;
            a = b;t++;
            b = c;t++;
        }
        t++;
        return c;
    }
}
int main()
{
    int n;
    cout<<"Enter term: ";
    cin>>n;
    auto start = chrono::high_resolution_clock::now();
    cout<<"The "<<n<<"th Fibonaci term is "<<fibo(n)<<endl;
    cout<<"T(n) = "<<t<<endl;
    return 0;
}
```

n	3	10	20	30	40
T(n)	13	48	98	148	198



Program for GCD

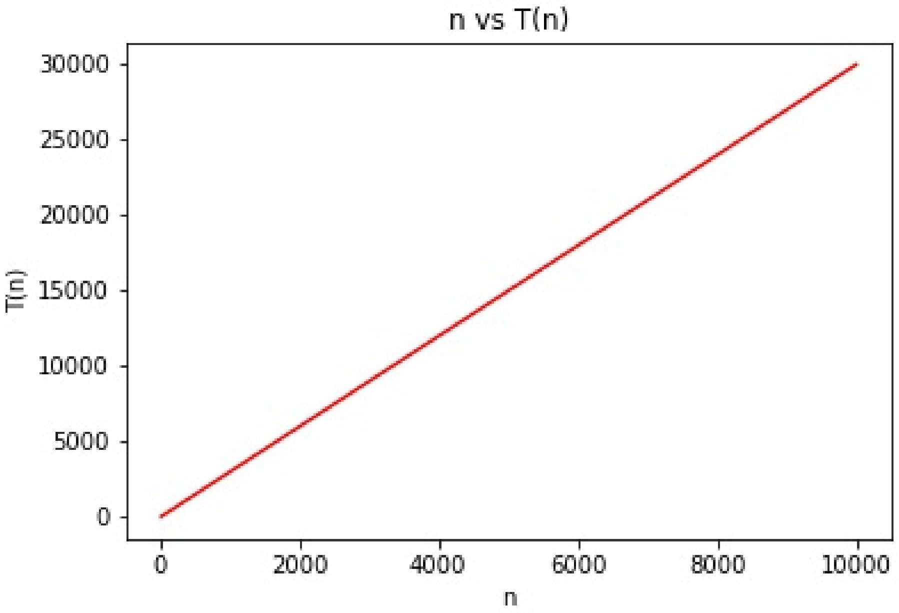
```
#include <iostream>
using namespace std;
int t = 0;
int gcd(int a, int b)
{
    t++;
    if(b == 0)
    {
        t++;
        return a;
    }
    else
    {
        int r;
        while(b != 0)
        {
            t++;
            r = a%b;t++;
            a = b;t++;
            b = r;t++;
        }
        t++;
        return a;
    }
}
int main()
{
    int a, b;
    cout<<"Enter a : ";
    cin>>a;
    cout<<"Enter b : ";
    cin>>b;
    cout<<"GCD = "<<gcd(a,b)<<endl;
    cout<<"T(n) = "<<t<<endl;
    return 0;
}
```

a,b	18,12	211,11	1109, 123	11099, 1234
T(n)	10	14	14	22

Program for Linear Search

```
#include <iostream>
int t = 0;
using namespace std;
void lSearch(int A[], int key, int n)
{
    int i, flag = 0;t++;
    for(i=0, t++; i<n; i++,t++)
       {
           t++;
           t++;
           if(A[i] == key)
            {
                flag = 1;t++;
                cout<<"Search Successful. Element found at "<<i<<"th position"<<endl;t++;
                t++;
                return;
            }
        }
    t++;
    if(flag == 0)
        cout<<"Search Unsuccessful. Element not found"<<endl;
}
int main()
{
    int A[10000],n, i, key;
    cout<<"How many elements? ";
    cin>>n;
    for(i=0; i<n; i++)
        A[i] = rand();
    cout<<"Enter element to search: ";
    cin>>key;
    lSearch(A,key,n);
    cout<<"T(n) = "<<t<<endl;
    return 0;
}
```

n	10	100	1000	10000
T(n)	33	303	3003	30003



Conclusion: Hence, in this lab, we successfully implemented the iterative algorithms for computing Fibonacci term, GCD and sequential search. We also analyzed the time and space complexity of these algorithms.