Lab1

Title: WAP to perform the empirical analysis of iterative algorithm to find nth Fibonacci number.

Source Code:

```
#include<stdio.h>
#include<time.h>
int main()
{
int n,i;
double first=0,second=1,temp, time;
clock_t start, end;
printf("Enter the position of fibonacci number:");
scanf("%d",&n);
start=clock();
printf("%f,%f",first,second);
i=3;
while(i<=n)
{
     temp=first+second;
     first=second;
     second=temp;
     printf("%f \n", temp);
     i++;
}
printf("\n\n\n");
end=clock();
printf("The nth fibonacci number is: %lf \n",temp);
time=((double)(end-start)*1000)/ CLOCKS_PER_SEC;
printf("Time=%lf mili seconds",time);
}
```

Result Analysis and Discussion

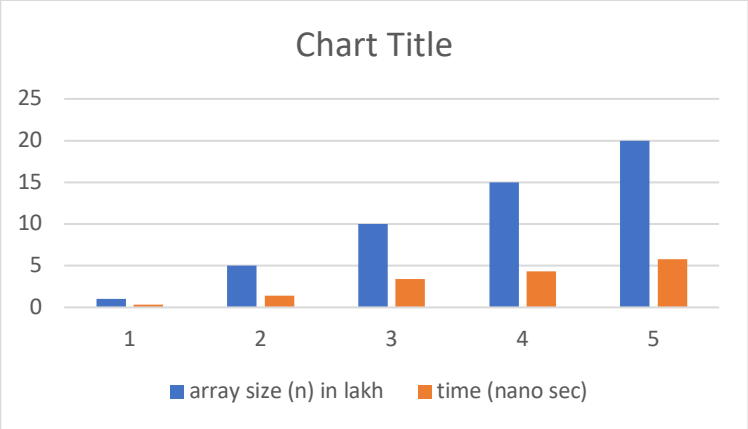
This experiment is conducted using following specifications. The algorithm is implemented using C language (clang-1400.0.29.202). During this test all the apps were closed to improve the results of the experiment.

Fig: PC used in experiment

In this experiment the algorithm to find the n^{th} Fibonacci number has been implemented and executed for different value of n. During this experiment for different value of n the time taken by the algorithm has been measured and tabulated as shown in table below.

Input size (n)	Time (micro sec)
100	16
300	17
600	18
900	20
1200	22
1477	24

The graph shown below is the plot of input n and the time in microseconds taken by the algorithm while running on a system recorded in table above.



Based on the above table and bar graph it is clearly seen that the array size n has linear relationship with the time taken by the system to find the input number.

Conclusion:

In this experiment it has been found that the size of input (n) has linear relationship with the time taken by the system to find the n^{th} Fibonacci number. This is equivalent with the asymptotic time complexity of the algorithm. Hence, this experiment proves complexity of the algorithm to find n^{th} Fibonacci number is O (n).

Lab 2

Title: WAP to perform the empirical analysis of iterative algorithm for linear search.

Source Code:

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
int main()
{
    int n, x, result;
    printf("Enter the size of array:");
    scanf("%d", &n);
    int array[n], randNum;
    for (int i = 0; i < n; i++)
    {
        randNum = rand() % n;
        array[i] = randNum;
    }
    printf("Enter the number to be searched:");
    scanf("%d", &x);
    clock_t start, end;
    start = clock();
    for (int i = 0; i < n; i++)
        if (array[i] == x)
            result = i;
    result = -1;
    end = clock();
    printf("Time=%lf nano seconds\n",
    ((double)(end - start) * 1000000) / CLOCKS_PER_SEC);
    (result == -1) ? printf("Element not found")
    : printf("Element found at index: %d", result);
}
```

Result Analysis and Discussion

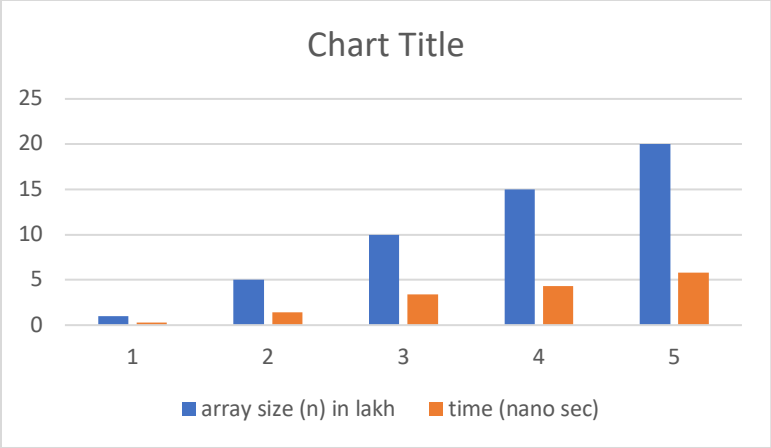
This experiment is conducted using following specifications. The algorithm is implemented using C language (clang-1400.0.29.202). During this test all the apps were closed to improve the results of the experiment.

Fig: PC used in experiment

In this experiment the linear search algorithm has been implemented and executed for different value of n. During this experiment for different value of n the time takenby the algorithm has been measured and tabulated as shown in table below.

Array size (n) in lakh	Time(nano sec) in thousand
1	0.308
5	1.422
10	3.03
15	4.222
20	5.686
20.5	5.865

The graph shown below is the plot of input n and the time in milliseconds taken by the algorithm while running on a system recorded in table above.



Based on the above table and graph it is clearly seen that the array size n has linear relationship with the time taken by the system to find the input number.

Conclusion:

In this experiment it has been found that the size of array (n) has linear relationship with the timetaken by the system to find the given number. This is equivalent with the asymptotic time complexity of the algorithm. Hence, this experiment proves complexity of the algorithm got is O(n).