Name : Sonal Dilip Gholap

UID : 2021600020

Branch : CSE-AIML

Batch : B

Experiment : 3

Aim : Experiment on recurrence relation.

- **Objectives :**

1. To understand recurrence relations.
2. To implement the algorithms of Bubble Sort and Fibonacci number using recursion.
3. To derive the time complexities of Fibonacci series and Bubble Sort from their recurrence relation.
4. To understand the difference between base case and recursive case.

- **Algorithm for Bubble Sort**

```
BubbleSort (arr, n)
    if n = 1
       return
    for i = 0 to n − 1
        if  arr[i] > arr[i + 1]  then
            swap arr[i] and arr[i + 1]

    BubbleSort (arr, n-1)
```

- **Algorithm for nth Fibonacci number**

```
fib (n)

    if n = 1
```

```
    return 0
if n = 2
    return 1
else
    return fib (n-1) + fib (n-2)
```

- **<u>Derivation of Time Complexity</u>**

1. Bubble Sort

Date
Page No.

II) Recurrence relation for Bubble sort

$$T(n) = 1 \qquad n=1$$

$$= T(n-1)+(n-1) \qquad n>1$$

* Substitution method

$$T(n) = T(n-1)+(n-1)$$
$$T(n-1) = T(n-2)+(n-2)$$
$$T(n-2) = T(n-3)+(n-3)$$

$$\therefore T(n) = T(n-2)+(n-2)+(n-1)$$

$$= T(n-3)+(n-3)+(n-2)+(n-1)$$

$$= T(n-3)+3n-(1+2+3)$$
$$\vdots$$
$$= T(n-k)+kn-(1+2+..+k)$$

Assume $n-k=0$
$n=k$

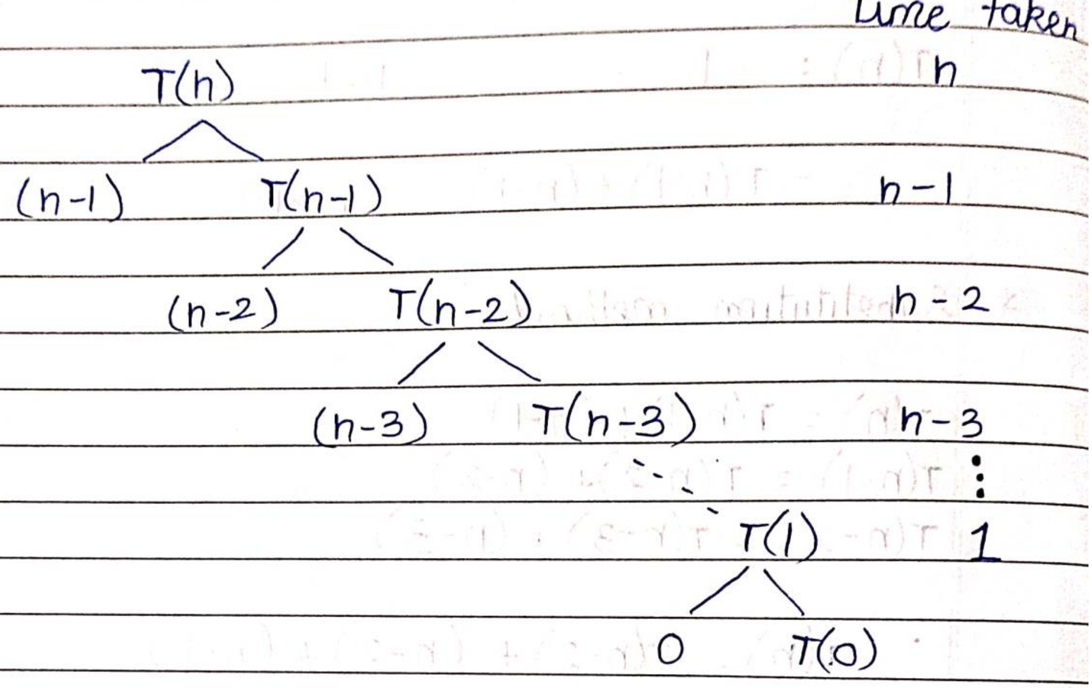
$$T(n) = T(0)+n^2-(1+2+..n)$$

$$= 1+n^2-\frac{n^2}{2}-\frac{n}{2}$$

$$T(n) = \frac{n^2}{2}-\frac{n}{2}+1$$

$\therefore$ Time Complexity of Bubble sort is $O(n^2)$

* Recurrence Tree method



$\therefore T(n) = 1 + \dots (n-2) + (n-1) + n$

$= \frac{n(n+1)}{2}$

$= \frac{n^2+n}{2}$

$\therefore T(n) = O(n^2)$

$\therefore$ Time complexity of Bubble Sort is $O(n^2)$

2. Fibonacci Number

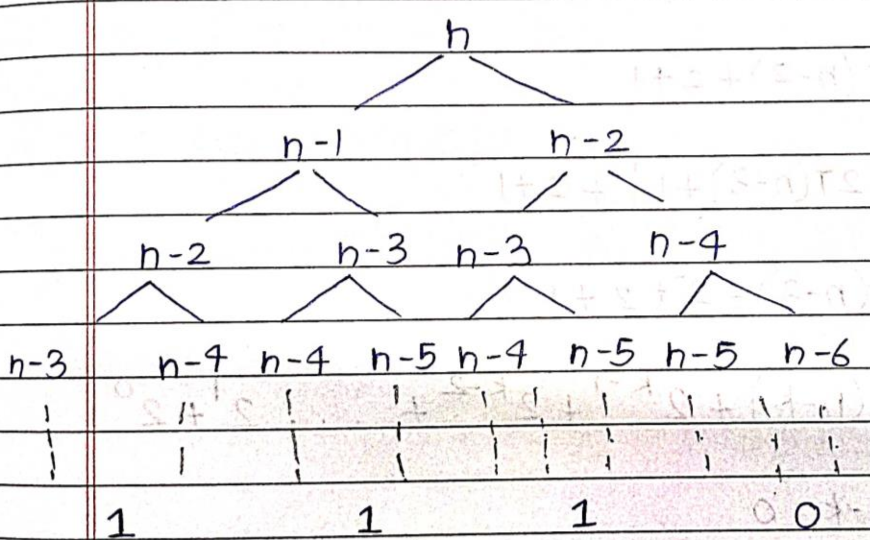
I) Recurrence relation for Fibonacci Number

$$T(n) = 1 \qquad n = 1 \text{ or } n = 2$$

$$= T(n-1) + T(n-2) \qquad n > 2$$

* Recurrence Tree method

$$T(n) = T(n-1) + T(n-2) + c$$



Time Taken

$2^0 c = c$

$2c = 2^1 c$

$4c = 2^2 c$

$8c = 2^3 c$

$\leq 2^{n-1} c$

$$\therefore T(n) \leq 2^0 c + 2^1 c + 2^2 c + \ldots 2^{n-1} c$$

$$< c(2^n - 1)$$

$$\therefore T(n) = 2^n$$

Time complexity of fibonacci number is $O(2^n)$

FABER-CASTELL

Date ______

Page No. ______

* Substitution method

$T(n) = T(n-1) + T(n-2) + O(1)$

To calculate the worst case time complexity i.e. $O(f(n))$ we assume, $T(n-2) \approx T(n-1)$

$\therefore \ T(n) = 2T(n-1) + 1$

$= 2[2T(n-2)+1]+1$

$= 2^2 T(n-2) + 2 + 1$

$= 2^2 [2T(n-3)+1] + 2 + 1$

$= 2^3 T(n-3) + 2^2 + 2 + 1$

$\vdots$

$T(n) = 2^k T(n-k) + 2^{k-1} + 2^{k-2} + \ldots 2^1 + 2^0$

Assume $n-k=0$

$n=k$

$T(n) = 2^n T(0) + 2^{n-1} + 2^{n-2} + \ldots 2^1 + 2^0$

$= 2^n + 2^n - 1$

$= 2.2^n - 1$

$= O(2^n)$

$\therefore$ Time complexity of fibonacci series is $O(2^n)$

Code :

```
#include<iostream>
#include <bits/stdc++.h>

using namespace std;

void towerOfHanoi(int n,char source, char dest, char middle){

if(n==0){
return;
}
towerOfHanoi(n-1,source,middle,dest);
cout<<"Moving disk "<<n<<" from "<<source<<" to "<<dest<<endl;
towerOfHanoi(n-1,middle,dest,source);

}

void bubblesort(int arr[],int n){
    if(n==1){
        return;
    }
    for(int i=0;i<n-1;i++){
        if(arr[i]>arr[i+1]){
            swap(arr[i],arr[i+1]);
        }
    }
    bubblesort(arr,n-1);
}

int fib(int n){
    if(n==1){
        return 0;
    }
    else if(n==2){
        return 1;
    }
    return fib(n-1)+fib(n-2);
}


int main(){

cout<<"Enter the nth term of the fibonacci series which you wish to find: ";
int p;
cin>>p;
cout<<p<<"th term of the fibonacci series is "<<fib(p);

int random[10] = {2,12,76,5,29,7,88,102,35,234};
cout<<endl<<"Array: ";
```

```
for(int i=0;i<10;i++){
    cout<<random[i]<<" ";
}
cout<<endl<<"Sorted Array: ";
bubblesort(random,10);
for(int i=0;i<10;i++){
    cout<<random[i]<<" ";
}
return 0;
}
```

Output :

```
-vjttazru.or5' '--stdout=Microsoft-MIEngine-Out-uwpqzfsa.hej' '--stderr=Microsoft-MIEngine-Err
or-xqoy4cqt.c5c' '--pid=Microsoft-MIEngine-Pid-wz2roff3.bxy' '--dbgExe=C:\Program Files (x86)\
mingw-w64\i686-8.1.0-posix-dwarf-rt_v6-rev0\mingw32\bin\gdb.exe' '--interpreter=mi'
Enter the nth term of the fibonacci series which you wish to find: 8
8th term of the fibonacci series is 13
Array: 2 12 76 5 29 7 88 102 35 234
Sorted Array: 2 5 7 12 29 35 76 88 102 234
PS C:\C++ Learning Course>
```

Conclusion :

1. Recurrence relation of Fibonacci series – **T(n) = T(n-1) + T(n-2)**
2. Time complexity of Fibonacci series – **O(2^n)**
3. Recurrence relation of Bubble Sort – **T(n) = T(n-1) + (n-1)**
4. Time complexity of Bubble Sort – **O(n^2)**
5. Recurrence equations are used to describe the runtime of Divide and Conquer algorithms.
6. The recursive case keeps on calling the next recursive case till it encounters the base case.
7. The recurrence relation always terminates when it reaches the base case.