

CSE 3004 DAA LAB TASK - 3

NAME : V.SAIKRISHNA

REG. NO :19BCE7638

Fibonacci

Using Naive algorithm:

```
class Main {  
    static int fib(int n) {  
        if (n <= 1)  
            return n;  
        return fib(n-1) + fib(n-2);  
    }  
}
```

```
}  
public static void main (String args[]) {  
    int n = 10;  
    System.out.println(fib(n));  
}  
}
```

OUTPUT:

```
8- class Main {  
9- static int fib(int n) {  
10     if (n <= 1)  
11         return n;  
12     return fib(n-1) + fib(n-2);  
13 }  
14- public static void main (String args[]) {  
15     int n = 10;  
16     System.out.println(fib(n));  
17     }  
18 }  
19
```

A screenshot of a console window with a title bar that says "input". The window has a dark background. In the top-left corner, there are three small icons: a downward arrow, a square with a diagonal line, and a person icon. The text "55" is displayed in the top-left corner of the console area. Below it, the message "...Program finished with exit code 0" is shown in green. The final line of text is "Press ENTER to exit console." in green, followed by a white cursor character (a small square).

Using DP:

```
class Main
{
    static int fib(int n)
    {
        int f[] = new int[n+2];
        f[0] = 0;
        f[1] = 1;
```

```
for (int i = 2; i <= n; i++) {
```

```
    f[i] = f[i-1] + f[i-2];
```

```
}
```

```
return f[n];
```

```
}
```

```
public static void main (String args[]) {
```

```
    int n = 20;
```

```
    System.out.println(fib(n));
```

```
}
```

```
}
```

OUTPUT:

```
7
8 class Main
9 {
10     static int fib(int n)
11     {
12         int f[] = new int[n+2];
13         f[0] = 0;
14         f[1] = 1;
15         for (int i = 2; i <= n; i++) {
16             f[i] = f[i-1] + f[i-2];
17         }
18         return f[n];
19     }
20     public static void main (String args[]) {
21         int n = 20;
22         System.out.println(fib(n));
23     }
24 }
```



input

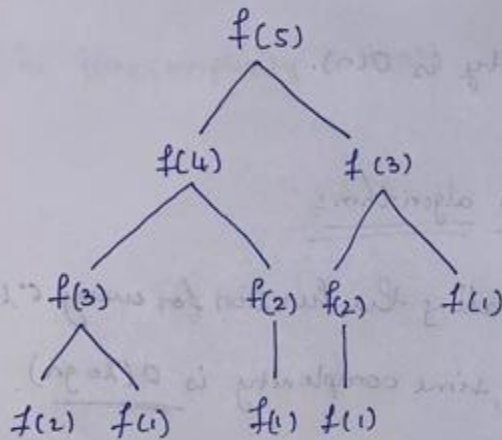
6765

...Program finished with exit code 0
Press ENTER to exit console.

ANALYSIS:

Fibonacci:

(1.) Using naive approach:-



Here it almost calling the method function at 2^n times.

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

(2.) Using DP:

Here, we are saving the ~~new~~ value of $f(5)$, $f(4)$ etc.

in an array. so at worst case we call it at

$O(n)$ times.

GCD

Using Navie Algorithm:

```
import java.lang.Math;
import java.util.Scanner;
public class Main {
    static int GCD(int a,int b) {
        int maximum=Math.max(a,b);
        int currentNumber=maximum-1;
        while(currentNumber>1){

            if((a%currentNumber==0)&&(b%currentNumber==0)
            ){ return currentNumber;
            }
        }
    }
}
```

```
else {
```

```
    currentNumber--;
```

```
}
```

```
}
```

```
return 1;
```

```
}
```

```
public static void main(String[] args){
```

```
    System.out.println(GCD(20,42));
```

```
}
```

```
}
```


OUTPUT:

```
8 import java.lang.Math;
9 import java.util.Scanner;
10 public class Main {
11 static int GCD(int a,int b) {
12 int maximum=Math.max(a,b);
13 int currentNumber=maximum-1;
14 while(currentNumber>1){
15     if((a%currentNumber==0)&&(b%currentNumber==0)){ return currentNumber;
16 }
17 else {
18     currentNumber--;
19 }
20 }
21 return 1;
22 }
23 public static void main(String[] args){
24 System.out.println(GCD(20,42));
25 }
26 }
```



The screenshot shows a Java IDE window titled "input". The console output displays the result of the GCD calculation for 20 and 42, which is 2. The output text is: "2", "...Program finished with exit code 0", and "Press ENTER to exit console.".

```
2
...Program finished with exit code 0
Press ENTER to exit console.
```

Using Euclidean Algorithm:

```
import java.util.*;
import java.lang.*;
class Main {
    public static int gcd(int a, int b) {
        if (a == 0)
            return b;
        return gcd(b%a, a);
    }
    public static void main(String[] args) {
        System.out.println(gcd(3918848,1653264));
    }
}
```

OUTPUT:

```
8 import java.util.*;
9 import java.lang.*;
10 class Main {
11     public static int gcd(int a, int b) {
12         if (a == 0)
13             return b;
14         return gcd(b%a, a);
15     }
16     public static void main(String[] args) {
17         System.out.println(gcd(3918848,1653264));
18     }
19 }
```



input

61232

...Program finished with exit code 0
Press ENTER to exit console.

ANALYSIS:

GCD :-

(1) Using naive algorithm :-

Here, we are calling the function for n times then,

the complexity is $O(n)$.

(2) Using Euclidean algorithm :-

there, we are calling the function for every \log

remainder. so, time complexity is $O(\log n)$.