## Today's Content

- → sum of digits
- → power (a, n)
- → power (a, n, p)
- → TC - {Recursive Relations}
- → SC - { Space Complexity }

## Recursion :-

1) Assumption → Decide what your function does, assume it does.

2) Main → Solve assumption using subproblems.

3) Base → when should function stop.

## * → Sum of Digits

123 → 6        34678

```
int sod(N) {        // Assume function will give
                       sod for any subpart of n.
    if (N == 0) {
        return 0
    }
    int rem = N % 10
    int quo = N / 10
    return rem + sod(quo)
}
```

34678

↓

3467

↓

346

↓

34

↓

3

↓

0

16

457

```
int sod (N=457) {
    if (N==0) {
        return 0
    }
    int rem = N%10 //7
    int quo = N/10 //45
    return rem + sod(quo)
           7     9
}
```

```
int sod (N=45) {
    if (N==0) {
        return 0
    }
    int rem = N%10 //5
    int quo = N/10 // 4
    return rem + sod(quo)
           5     4
}
```

```
int sod (N=4) {
    if (N==0) {
        return 0
    }
    int rem = N%10 //4
    int quo = N/10 // 0
    return rem + sod(quo)
           4     0
}
```

```
int sod (N=0) {
    if (N==0) {
        return 0
    }
    int rem = N%10 //4
```

```
int quo = n/10 // 0
return rem + sod(quo)
}
```

↓

Ques) Pow (a, n) → calculate and return $a^n$.

e.g. a = 3, n = 4 → $3^4$ → 81.

Note:- No Inbuilt function.

$a^5 \to a^4 * a$
$a^{10} \to a^9 * a$
$a^n \to a^{n-1} * a$

```
pow (a, n) {
  if( n==0) { return 1 }
  if(n==1) { return a }     } any of this will work.

  return a * pow (a, n-1)
}
```

2nd Approach :-

$a^{10} = a^5 * a^5$
$a^{12} = a^6 * a^6$
$a^{13} = a^6 * a^6 * a$

$a^{15} = a^7 * a^7 * a$

$a^{18} = a^9 * a^9$

Recursion Better

} -> fast Exponentiation.

```
pow (a, n) {
    if (n==0) {return 1 }
    long he = pow (a, n/2)
    long ha = he * he
    if (n%2 = = 0 ) {
            return ha
    } else {
        return ha * a
    }
```

T.C
↓
$O(\log_2 n)$

$a^{10}$

```
pow (a, n=10) {
    if (n==0) {return 1 }
    he = pow (a, n/2)
    ha = he * he
    if (n%2 = = 0 ) {
            return ha
    } else {
        return ha * a
    }
```

$a^5$

pow (a, n=5)

$\downarrow \quad \uparrow a^2$

pow(a, n=2)

$\downarrow \quad \uparrow a$

pow(a, n=1)

$\downarrow \quad 1 \uparrow$

pow(a, n=0)

```
iterpow(a,n){
    ans=1
    for(i=1; i<=n; i++){
        ans = ans+a
    }
    return ans
}
```

T.C → O(n).

$10^8$ iterations = 1sec

| Input | N | $\log_2 N$ | $\log_2(2^{30})$ |
|---|---|---|---|
| $N=10^9$ | $10^9$ | 30 | |
| | 10sec | $3*10^{-7}$sec. | |

$2^{10} \Rightarrow 1024 \approx 1000 \Rightarrow 10^3$

$\downarrow$

$2^{30} \Rightarrow 10^9$

* Calculate $a^n \% p$.

* Constraints :- $1 \le (a, n, p) <= 10^9$

```
     int  int  int
pow (a , n , p ) {
    if (n==0) { return 1 }
    long he = pow (a, n/2, p ) % p.
    long ha = (he % p * he % p ) % p.
    if (n%2==0 ) {
            return ha
    } else {
        return (ha % p * a % p ) % p.
    }
```

he * he
$10^9$ $10^9$

9:58 am – 10:08 am

Time → T(N).

```
— Sum( N) {
   if (N==1) return 1
   return Sum (N-1) + N
  }
```

→ n

int a = n+y;

$$\boxed{T(N) = T(N-1) + 1}$$

$T(N-1) = T(N-2) + 1$

$T(N) = T(N-2) + 2$

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

$T(N) = T(N-3) + 3$

$\vdots$

// generalize

$T(N) = T(N-k) + \underline{k}$.

$T(1) = O(1)$.

what if $K = \underline{N}$.

$T(N) = T(0) + N$

$T(N) = 1 + N$

$\underline{T(N) = O(N)}$.

```
— fact (n) {  ⟶ T(n)
    if (n==1) {return 1}
      return n * fact(n-1)
  }
```

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

↳ Recurrence Relation.

```
*            ↱ T(n)
pow (a, n) {
    if (n==0) {return 1}
    long ha = pow (a, n/2)
    long ha = ha * ha
    if (n%2 = = 0) {
          return ha
    } else {
          return ha * a
     }
```

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

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + 1$$

$$T(n) = T\left(\frac{n}{4}\right) + 2$$

$$T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + 1$$

$$T(n) = T\left(\frac{n}{8}\right) + 3$$

-: generalise :-

$$T(n) = T\left(\frac{n}{2^k}\right) + k$$

$$T(0) = 1 \rightarrow T(1) = 1$$

$\frac{n}{2^k} = 0$ X

$$\frac{n}{2^k} = 1 \Rightarrow n = 2^k$$

$$\log_2 n = k \log_2 2$$

$$k = \log_2 n$$

$$T(n) = T(1) + \log_2 n$$

$$T(n) = 1 + \log_2 n \Rightarrow T.C \Rightarrow O(\log_2 n)$$

# < Solve equations >

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

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + 1$$

$$T(n) = 2\left(2T\left(\frac{n}{4}\right) + 1\right) + 1$$

$$T(n) = 4T\left(\frac{n}{4}\right) + 3.$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + 1$$

$$T(n) = 8T\left(\frac{n}{8}\right) + 7$$

$$T(n) = 16\,T\left(\frac{n}{16}\right) + 15$$

// generalized equation :-

$$T(n) = 2^k\,T\left(\frac{n}{2^k}\right) + 2^k - 1$$

// Lets say $T(1) = 1$.

$$\frac{N}{2^k} = 1 \Rightarrow k = \log_2 N.$$

$$T(N) = 2^{\log_2 N} T(1) + 2^{\log_2 N} - 1$$

$$T(N) = N(1) + N - 1$$

$$T(N) = 2N - 1.$$

T.C $\Rightarrow$ $O(N)$.

Ques) $T(N) = 2T(N-1) + 1$

$T(N-1) = 2T(N-2) + 1$

$$T(N) = 4T(N-2) + 3$$

$$T(N) = 8T(N-3) + 7$$

$$T(N) = 16T(N-4) + 15$$

// Generalized equation :-

$$T(N) = 2^k T(N-k) + 2^k - 1$$

$$T(0) = 1$$

$$N - k = 0$$

$$k = N$$

$$T(N) = 2^N T(0) + 2^N - 1$$

$$T(N) = 2 \cdot 2^N - 1 \approx O(2^N)$$

```
int fib(N) {
    if(N==1 || N==2) { return 1 }
    return fib(N-1) + fib(N-2)
}
```

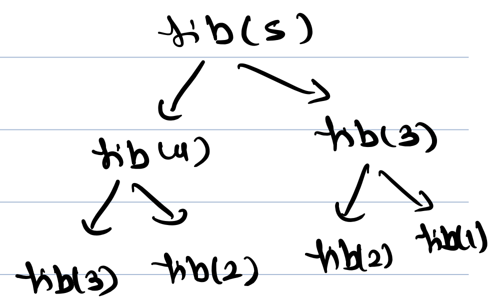
$$T(N) = T(N-1) + T(N-2) + 1.$$

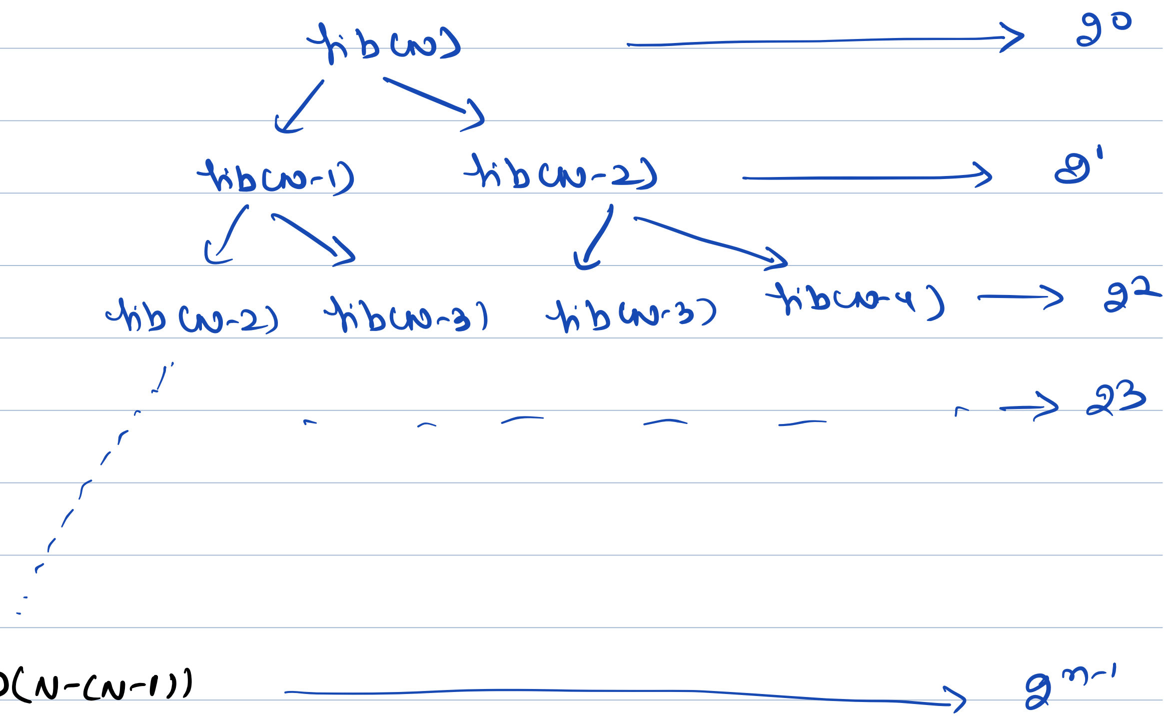
$$\left\{\begin{array}{l} T(N-1) = T(N-2) + T(N-3) + 1 \\ T(N-2) = T(N-3) + T(N-4) + 1 \end{array}\right.$$

$$= T(N-2) + 2T(N-3) + T(N-4) + 3$$

> Note:- If more than 1 time func is present, better to go with recursive method.

```
int fib(N) {
    if(N==1 || N==2) {return 1 }
    return fib(N-1) + fib(N-2)
}
```



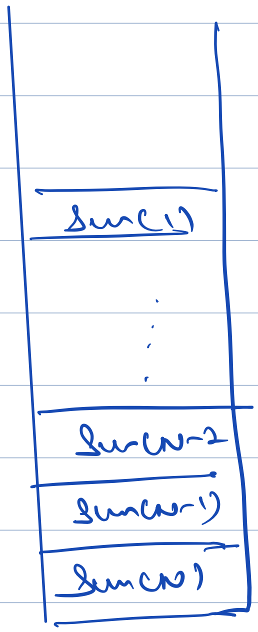




Total → $2^0 + 2^1 + 2^2 + \dots 2^{n-1}$

T.C → $O(2^n)$

11:08 – 11:11 pm

* S.C → Max stack size at any given point.




sum(N)

↳ sum(N-1)

↳ sum(N-2)

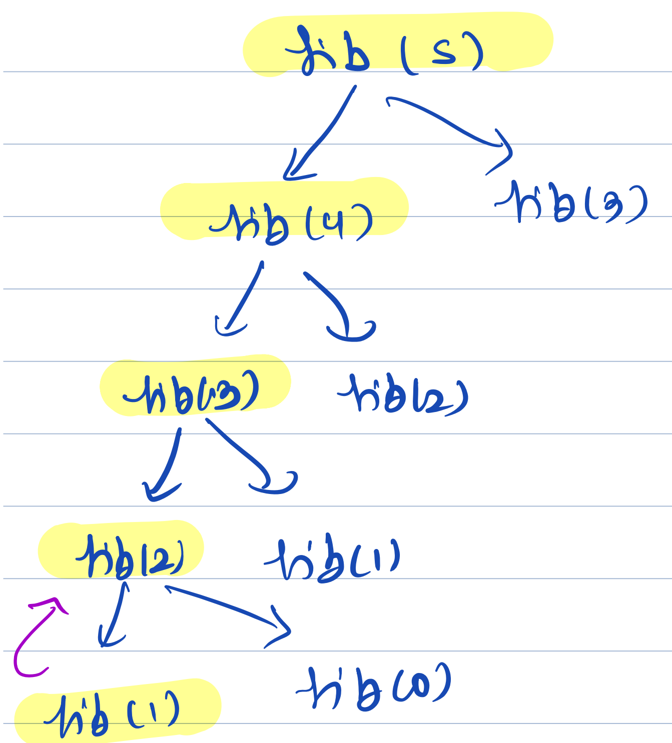
↳ sum(N-3)

↳ sum(1)

S.C → O(N).

S.C → O(N).

```
int fib(N) {
    if(N==1 || N==2) {return 1 }
    return fib(N-1) + fib(N-2)
}
```





Advance → ME.

7 day break → at Max.

Sunday → Comparators.

↓

Solve Problems.

1-3pm.

* Doubts :-

```
public int fun(int x, int n) {
    if (n == 0)
        return 1;
    else if (n % 2 == 0)
        return fun(x * x, n / 2);
    else
        return x * fun(x * x, (n - 1) / 2);
}
public void main() {
    int ans = fun(2, 10);
    System.out.println(ans);
}
```