Todays Content

1. Sum of Digits of N
2. Print 1 2 2 3 3 3 4 4 4 4
3. Fibanacci..
4. Power Function
5. Fast exponentiation with % arithmetic.

Steps for Recursion:

Assumption: Decide what your function does # {input, does, return}

Main logic: Solve problem using Subproblems # Recursive step

have a believe that subproblem will work as per assumption.

Base Condition: Input for which recursion needs to stop

Q: Given N return sum of digits using recursion.

<u>Note</u>: N > 0

Ex: Sum(239) = 2+3+9 = 14

Sum(7864) = 7+8+6+4 = 25

Sum(7864) = sum(786) + 4

Ex: N = $\boxed{d_1\ d_2\ ..\ d_{y-1}}\ \boxed{d_y}$ → N%10 = d_y

N/10   N%10   ↳ N/10 = $d_1\ d_2\ ..\ d_{y-1}$

Sum(N) = Sum(N/10) + N%10

Ass: Given N, calculate & return sum of digits of N.

```
int Sum(int N){
    if(N==0){return 0;}
    return sum(N/10) + N%10
}
```

$\boxed{d_1\ d_2\ ..\ d_{y-1}}$ d_y

Trace:

```
int Sum(N=365) : a
    if(N==0){return 0}
    return Sum(N/10) + N%10;
}
```

Q8: Given N print below pattern 1 2 2 3 3 3 ....

Pat(4) : 1 2 2 3 3 3 4 4 4 4

Pat(5) : 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5

Pat(4)

loop 5 time

print(5)

Pat(N) : 1 2 2 3 3 3 .. N-1 N-1 .. N-1 N N.... N N

Pat(N-1)

loop N times

print(N)

Ass: Given N, print the pattern & return nothing.

```
void Pat(int N){
    if(N==0){ return;}
    Pat(N-1);
    for(int i=1; i<= N; i++){
        print(N);
    }
}
```

Fibonacci:

0	1	2	3	4	5	6	7	8	9	10

Series = 0 1 1 2 3 5 8 13 21 34 55

Note: $N >= 0$

Fib(6) = Fib(5) + Fib(4)

Fib(10) = Fib(9) + Fib(8)

Fib(N) = Fib(N-1) + Fib(N-2)

Ass: Given N, calculate & return N^{th} fibonacci number

```
int Fib(int N){
    if(N==0){ return 0; }
    if(N==1){ return 1; }      if(N==0|| N==1){ return N; }
                               if( N<=1){ return N;}
    return Fib(N-1) + Fib(N-2)
}
```

How to get base conditions ?

Input for which subproblems will fail

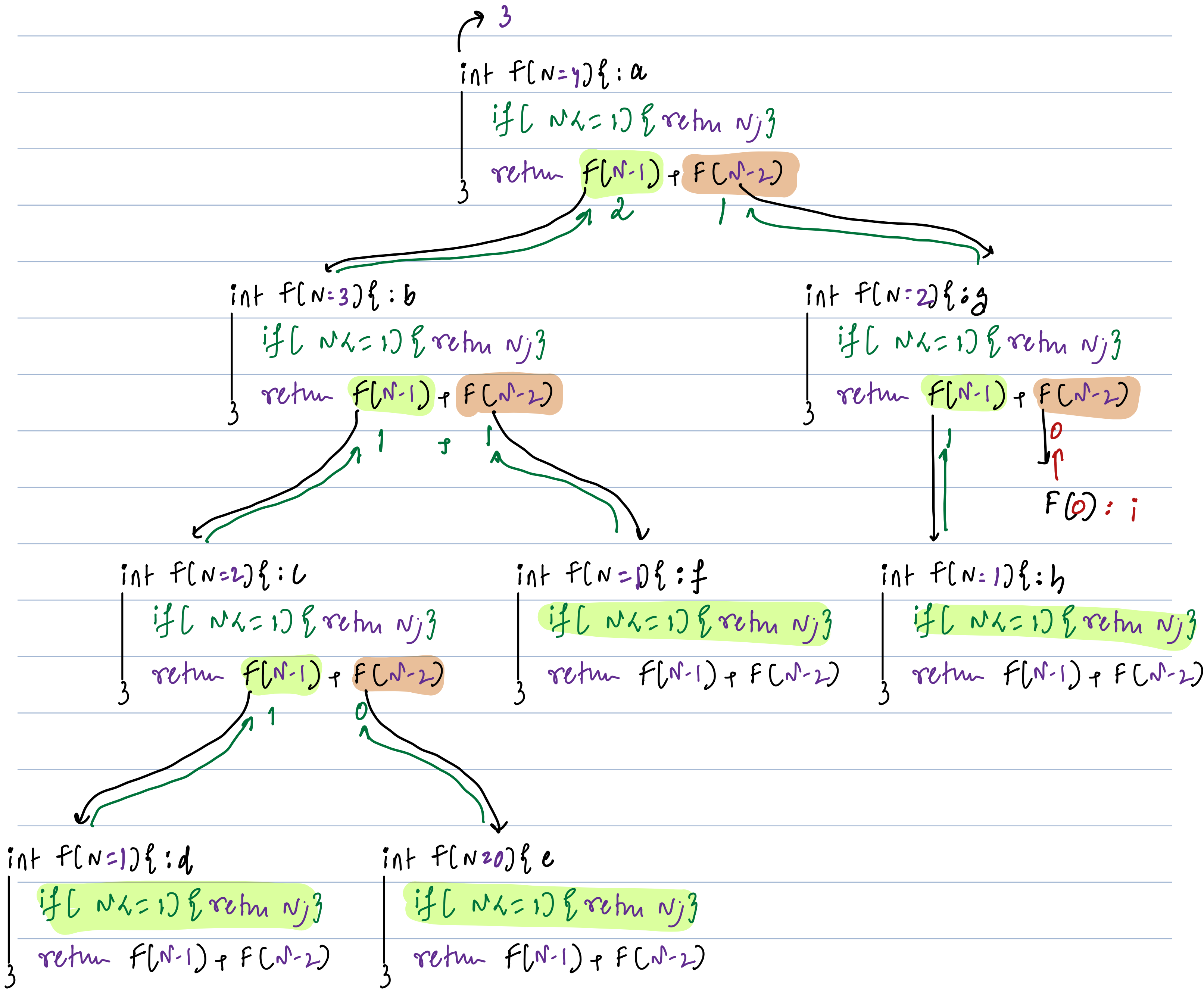
Ex:

Fib(N) = Fib(N-1) + Fib(N-2)

Fib(2) = Fib(1) + Fib(0)

* Fib(1) = Fib(0) + Fib(-1)

* Fib(0) = Fib(-1) + Fib(-2)


3
int f(N=4){ : a
if(N<=1){return N;}
return f(N-1) + f(N-2)
}
2
1
int f(N=3){ : b
if(N<=1){return N;}
return f(N-1) + f(N-2)
}
1
+
1
int f(N=2){ : g
if(N<=1){return N;}
return f(N-1) + f(N-2)
}
0
F(0) : i
int f(N=2){ : c
if(N<=1){return N;}
return f(N-1) + f(N-2)
}
1
0
int f(N=1){ : f
if(N<=1){return N;}
return f(N-1) + f(N-2)
}
int f(N=1){ : h
if(N<=1){return N;}
return f(N-1) + f(N-2)
}
int f(N=1){ : d
if(N<=1){return N;}
return f(N-1) + f(N-2)
}
int f(N=0){ e
if(N<=1){return N;}
return f(N-1) + f(N-2)
}

Q2: pow(a,n) : Calculate & return a^n

pow(a,5) = a * a * a * a * a;

pow(a,5) = pow(a,4) * a

pow(a,n) = a * a * -- a * a

pow(a,n) = pow(a,n-1) * a

Ass: Given a, n calculate a^n & return it

```
long pow(a, n){   TC: O(N)   SC: O(N)
    if(n==0){return 1;}
    return pow(a,n-1) * a
}
```

Other ways to solve a problem with Subproblems:

Note: Can break a problem at corners or at center.

pow(a,8) = $a^7 * a$

pow(a,8) = $a^4 * a^4$

pow(a,4) * pow(a,4)

pow(a,9) = $a^4 * a^4 * a$

pow(a,4) * pow(a,4) * a

Ass: Given a, n calculate & return a^n.

```
long pow(a, n){   TC: O(N)   SC: O(logN)
    if(n==0){return 1;}
    if(n%2==0){
        return pow(a,n/2) * pow(a,n/2)
    }
    else{
        return pow(a,n/2) * pow(a,n/2) * a;
    }
}
```

Assumption: Given a, n calculate & return a^n.

```
long pow(long a, long n){ TC: O(logN)   SC: O(logN)
    if(n==0){return 1;}
    long t = pow(a, n/2); # t = a^(n/2)
    if(n%2==0){
        return t*t;      #even
    }
    else{
        return t*t*a;    #odd
    }
}
```

5Q Given a, N, M calculate and return $a^N \% M$

pow(a, n, m): $a^n \% m$

Constraints

$1 <= a <= 10^9$

$1 <= N <= 10^{18}$

$m = 10^9+7$

a n m

Ex: pow(3, 4, 4) = $(3^4) \% 4 = 81 \% 4 = 1$

$\% M = \{0..M-1\}$ at max $= M-1 \approx 10^9+7-1 = 10^9+6 \approx 10^9$

Ex: pow(a=2, b=1000, M=10^9+7) : $(2^{1000}) \% [10^9+7]$

pow(a=10, b=1000, M=10^9+7) = $(10^{1000}) \% [10^9+7]$

Issue: We cannot calculate a^n first & apply % m later ? $a^n >>$ long.

Hint: Apply % arithmetic

Ex: pow(a, n, m) = $(a^n) \% m$ # t = pow(a, n/2, m) $\approx 10^9$

if(n % 2 == 0){

$(a^{n/2} * a^{n/2}) \% m$ # $(a*b) \% m = (a\%m * b\%m) \% m$

$(a^{n/2} \% m * a^{n/2} \% m) \% m$

(pow(a, n/2, m) * pow(a, n/2, m)) % m

(t * t) % m

}

else{ # t = pow(a, n/2, m)

$(a^{n/2} * a^{n/2} * a) \% m$

$((a^{n/2} * a^{n/2}) \% m * a\%m) \% m$

$((a^{n/2} \% m * a^{n/2} \% m) \% m * a\%m) \% m$

((pow(a, n/2, m) * pow(a, n/2, m)) % m * a%m) % m

((t * t) % m * a % m) % m

}

$((10^9 * 10^9 = 10^{18}) \% m = 10^9 * 10^9 = 10^{18}) \% m = 10^9$

Ass: Given a, n, m calculate & return $(a^n)\%m$

```
long pow(int a, long n, int m){
    if(n==0){return 1;}

    long t = pow(a, n/2, m);
    if(n%2==0){
        return (t*t)%m
    }
    else{
        return ((t*t)%m * a%m)%m
    }
}
```

What if ?

$pow(a,n,m) = (a^n)\%m$

if(n%2==0){

$(a^{n/2} * a^{n/2})\%m$ # $(a*b)\%m = (a\%m * b\%m)\%m$

$(a^{n/2}\%m * a^{n/2}\%m)\%m$

$(pow(a,n/2,m) * pow(a,n/2,m))\%m$

}

$t = pow(a, n/2, m) = (a^{n/2})\%m \approx 10^9$

else{

$(a^{n/2} * a^{n/2} * a)\%m$ # $(a*b*c)\%m = (a\%m * b\%m * c\%m)\%m$

$(a^{n/2}\%m * a^{n/2}\%m * a\%m)\%m$

$(pow(a,n/2,m) * pow(a,n/2,m) * a\%m)\%m$ 2:25 break

$(t * t * a\%m)\%m$

$(10^9 * 10^9 * 10^9)\%m$

Can exceed long range get's overflow

}

$(10^{27})\%m$