1) Drop in PSP

pow(a, n)

pow(a, n, p)

TC for recursive codes

SC for recursive codes

Q1) Given a, n find a^n using recursion $n > 0$

a	n	a^n
2	5	$2^5 = 32$
3	4	$3^4 = 81$
5	3	$5^3 = 125$

Assumption: Given a, n calc & return a^n

$A^N = A \times A^{N-1}$ pow(A,N) = pow(A,N-1) * A

```
int pow(a, n) {                                   ✓
    Base case:   if (n==1) { return a }   // if (n==0) { return 1 }   ✓
    Main logic:  return pow(a, n-1) * a
}
```

pow(a,5) → pow(a,4) → pow(a,3) → pow(a,2) → pow(a,1)

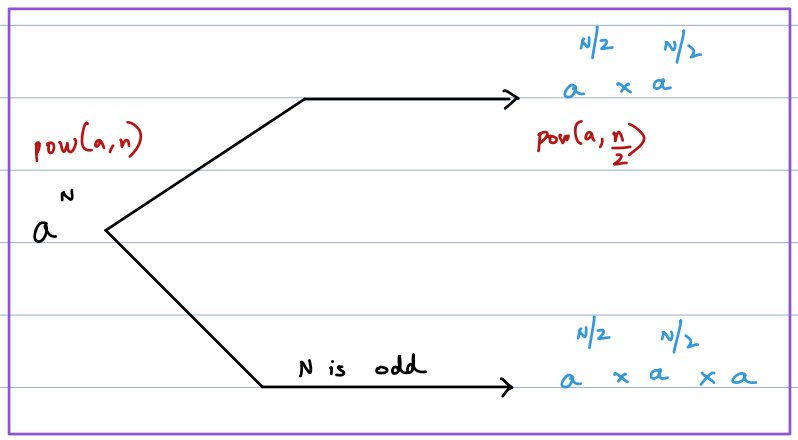
pow(a, N) TC: O(N)

$a^8 = a^7 \times a^1$ ⟶ $a^8 = a^4 \times a^4$

$a^{10} = a^5 \times a^5$

$a^{13} = a^6 \times a^6 \times a$

$a^{15} = a^7 \times a^7 \times a$



$\frac{N-1}{2}$

N is odd

N = 13 $\quad \frac{N-1}{2} = 6 \quad \frac{N}{2} = 6$

Fast exponentiation

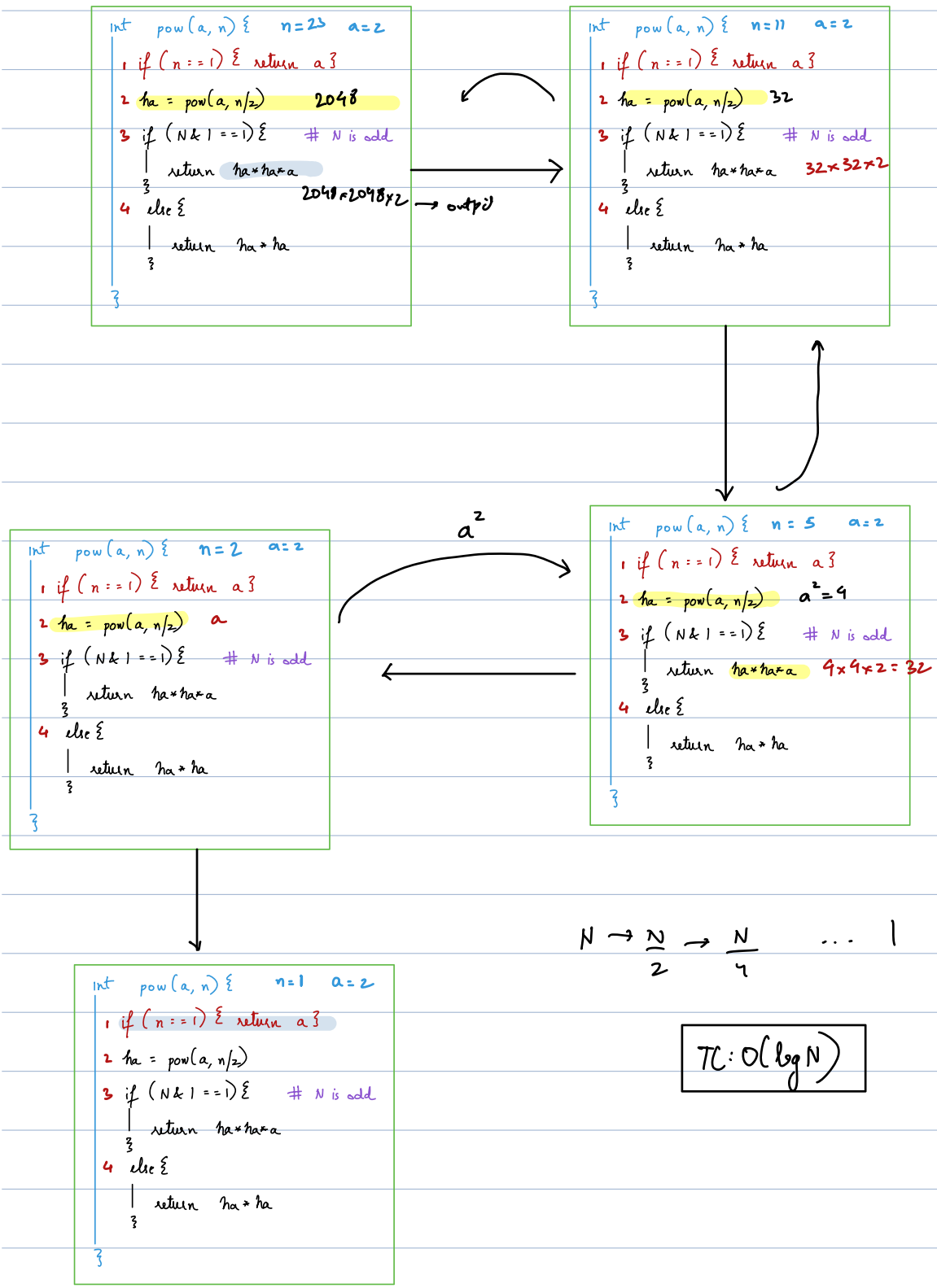
math.pow(a,n)

Main logic :

```
int pow(a, n) {
    if (n == 1) { return a }
    ha = pow(a, n/2)
    if (N & 1 == 1) {          # N is odd
        return ha*ha*a
    }
    else {
        return ha * ha
    }
}
```

a ** 1000

a^{1000}



$N \rightarrow \frac{N}{2} \rightarrow \frac{N}{4} \quad \ldots \quad 1$

TC: O(log N)

Q3) Given a, n, m calculate $a^n \% m$

$\hookrightarrow [0, m-1]$

$1 \le a \le 10^9$

$1 \le n \le 10^9$

$1 \le m \le 10^9$

HW: Update this method to take care of overflow

10^{1000}

```
int pow(a, n) {
    if (n == 1) { return a }
    long ha = pow(a, n/2)
    long hp = ((ha%.m) * ha%.m) % m
    if (N & 1 == 1) {        # N is odd
        return (hp%.m * a%.m) %.m
    }                          (10^9)
    else {
        return hp
    }
}
```

1) Modular arithmatic class revise

9:53 - 10:08

1) Do a dry run

Substitution method for TC calculation

TC for recursive codes using recursive solution

N = 1 $\quad\quad$ f(1)

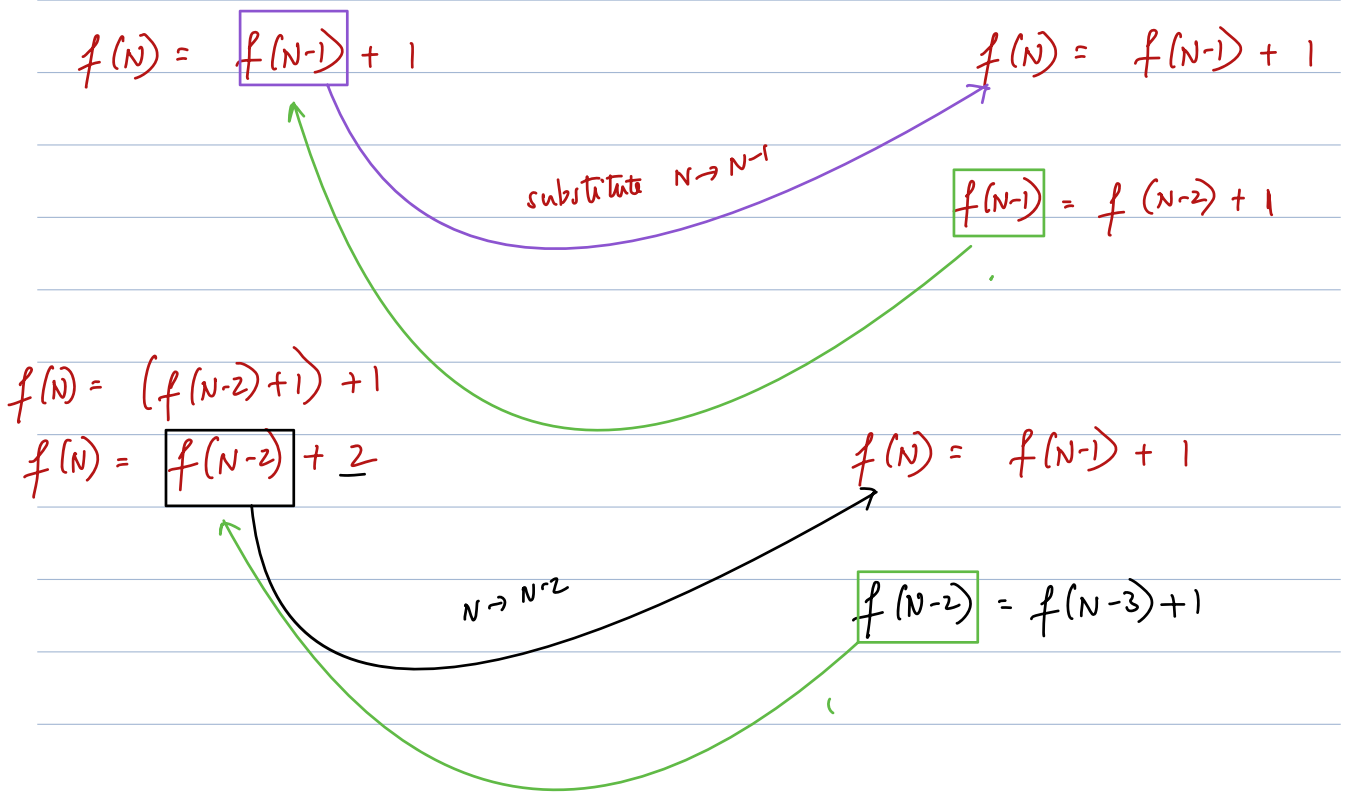
```
int sum(N){
    if(N==1){ return 1 }
    return sum(N-1) + N
}
```

TC → f(N)

$f(N) = O(1) + f(N-1)$

1) Find recursive time equation
2) Find general solution after k substitution
3) Equate unknown with base condition and find TC

$f(N) = f(N-1) + 1$ Recursive time equation



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

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

$\vdots$

$f(N) = f(N-k) + k$

$f(N)$ → final solution

$f(N-k)$ → unknown term

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

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

$f(N) = N$

$f(1) = 1$

$f(N-k) = f(1)$

$N-k = 1$

$k = N-1$

$f(1) = 1$

```
int pow(a, n) {
    if (n == 1) { return a }
    ha = pow(a, n/2)
    if (N & 1 == 1) {        # N is odd
        return ha*ha*a
    }
    else {
        return ha * ha
    }
}
```

step 1)

TC → $f(N)$

$$f(N) = O(1) + f(N/2)$$

$$f(N) = f(N/2) + 1$$

$N \to N/2$

$$f(N) = f(N/2) + 1$$

$$f(N/2) = f(N/4) + 1$$

$$f(N) = \left(f\left(\frac{N}{4}\right) + 1\right) + 1 = f\left(\frac{N}{4}\right) + 2$$

$$f(N) = f\left(\frac{N}{4}\right) + 2$$

$N \to N/4$

$$f(N) = f(N/2) + 1$$

$$f\left(\frac{N}{4}\right) = f\left(\frac{N}{8}\right) + 1$$

$f(N) = f\left(\frac{N}{8}\right) + 3$

1) $f(N) = f\left(\frac{N}{2}\right) + 1$

2) $f(N) = f\left(\frac{N}{4}\right) + 2$

3) $f(N) = f\left(\frac{N}{8}\right) + 3$

$\log_x x^n = n$

$f(N) = f\left(\frac{N}{2^k}\right) + k$

$f(N) = f(1) + \log_2 N$

$f(N) = 1 + \log_2 N$

$f(N) = O(\log N)$

$f\left(\frac{N}{2^k}\right) = f(1)$

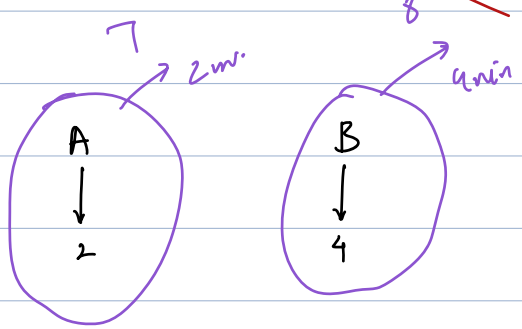
$\frac{N}{2^k} = 1$

$N = 2^k$

Taking $\log_2$ both sides

$\log_2 N = \log_2 2^k$

$\log_2 N = k$



A * B

4, 6, ~~8~~

Tim equation:

$$f(N) = 1 + f\left(\frac{N}{2}\right) + f\left(\frac{N}{2}\right)$$

```
int pow(a, n) {
    if (n == 1) { return a }
    ha = pow(a, n/2) x pow(a, n/2)
    if (N & 1 == 1) {        # N is odd
        return haxa
    }
    else {
        return a
    }
}
```

$f(N) = 2f\left(\frac{N}{2}\right) + 1$ $\qquad$ $f(N) = 2f\left(\frac{N}{2}\right) + 1$

$N \to N/2$

$f\left(\frac{N}{2}\right) = 2f\left(\frac{N}{4}\right) + 1$

$$f(N) = 2\left[2f\left(\frac{N}{4}\right) + 1\right] + 1$$

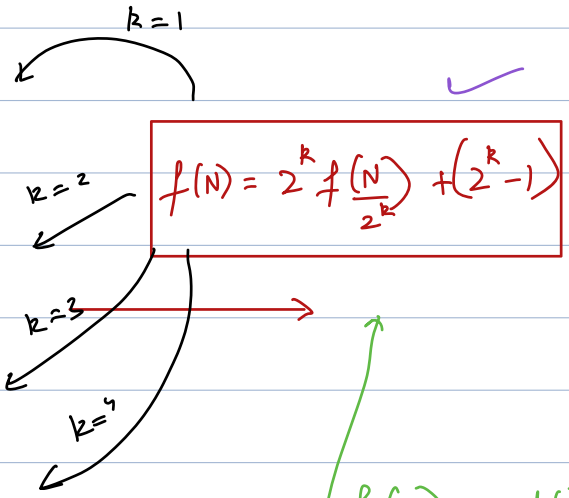
$f(N) = 4f\left(\frac{N}{4}\right) + 3$ $\qquad$ $f(N) = 2f\left(\frac{N}{2}\right) + 1$

$N \to N/4$

$f\left(\frac{N}{4}\right) = 2f\left(\frac{N}{8}\right) + 1$

$$f(N) = 4\left[2f\left(\frac{N}{8}\right) + 1\right] + 3$$

$f(N) = 8f\left(\frac{N}{8}\right) + 7$



$f(N) = 2f\left(\frac{N}{2}\right) + (2-1)$

$f(N) = 4f\left(\frac{N}{4}\right) + (4-1)$

$f(N) = 8f\left(\frac{N}{8}\right) + (8-1)$

$f(N) = 16f\left(\frac{N}{16}\right) + (16-1)$

$f(N) = Nf(1) + N - 1$

$= 2N - 1$

$f(N) = O(N)$

$f(1) = O(1)$

$f\left(\frac{N}{2^k}\right) = 1$

$N = 2^k$

HW

$f(N) = 4f\left(\frac{N}{2}\right) + 1 \qquad f(1) = N$

$f(N) = f(N-1) + N \qquad f(0) = 1$

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

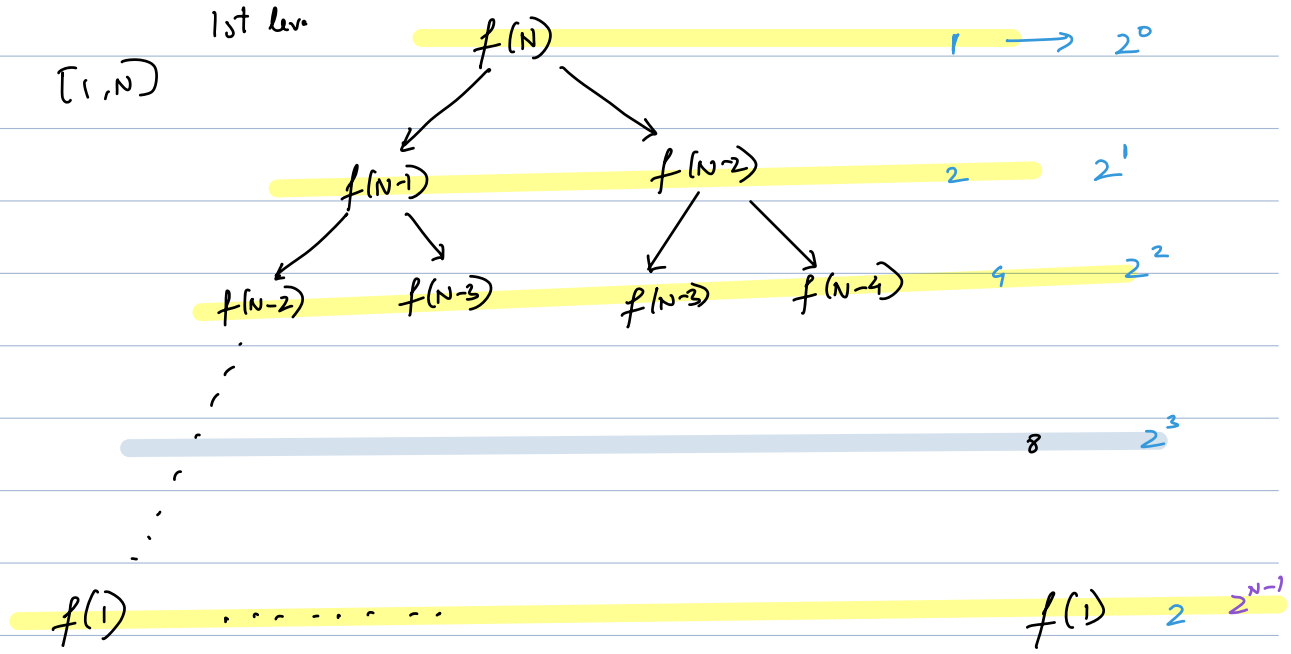
Right side more than one recursive term

substitution fails

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

for(i=0; i<N; i++)

Time equation: $f(N) = f(N-1) + f(N-2) + 1$



N levels

TC: (Total No. of function calls) × Time taken for every function call

$1 + 2 + 4 + 8 \ldots 2^{N-1}$

sum of GP $a\left(\frac{r^n - 1}{r-1}\right)$

$r = 2 \quad n = N$

$a = 1$

TC: $O(2^N)$

Space complexity

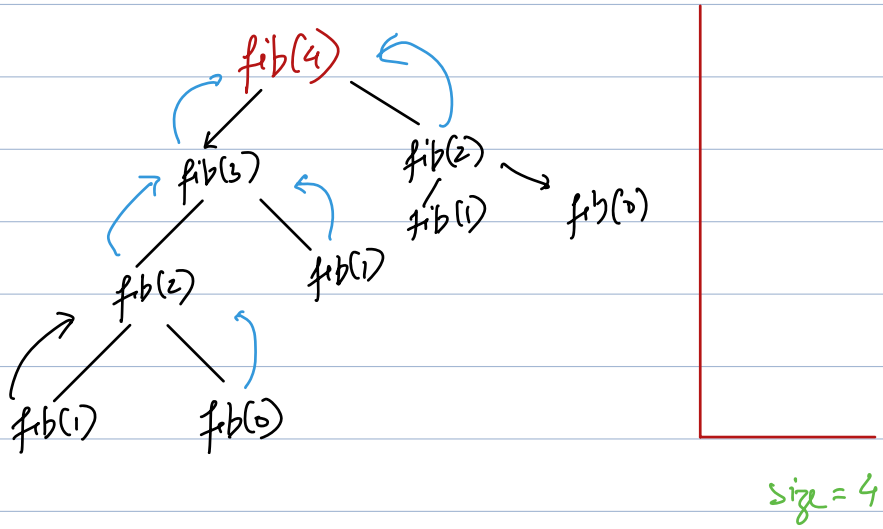
Dry run

Max no. of entries in function call stack

fib(4)

TC: $O(2^N)$

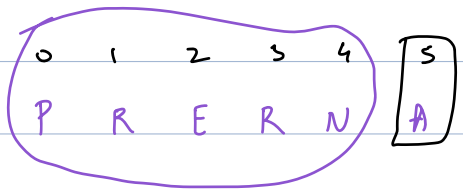
SC: $O(2^N)\alpha$



size = 4

SC: $O(N)$

Print string in reverse order using recursion



[0:5]

Assumption: rev will print the string in rev order

```
void rev(s, i) {
    if (i == -1) { return }
    print(s[i])
    rev(s, i-1)
}
```

i = len-1