Todays Content:

a) a^n:

1) $a^n = a^{n-1} * a$
2) $a^n = a^{n/2} * a^{n/2}$
3) Special way

Recursion: Function calling itself

Solving a problem using Smaller Input of Same Problem: SubProblem

Sum(5) = 1 + 2 + 3 + 4 + 5

Sum(5) = Sum(4) + 5

↳ SubProblem

Sum(N) = 1 + 2 + 3 + .. N-1 + N

Sum(N) = Sum(N-1) + N

Steps to Recursive Code:

Assumption A: Decide what your function does & assume it works

Hint: Question itself Assumption

Main Logic M: Solving asumption using SubProblems

Base Conditions B: Input, for which we want to stop recursion

↳ Why? In Stopping Recursion

Note: Dont use post increment operators, while writing recursion.

Pow way: 1

Note: Don't worry about overflows

Pow(a,5) = a * a * a * a * a

Pow(a,5) = Pow(a,4) * a // SubProblem = Pow(a,4)

Pow(a,n) = a * a * a ... * a * a] n times = a^n

Pow(a,n) = Pow(a,n-1) * a // SubProblem = Pow(a,n-1)

```
// a > 0          // n >= 0
long Pow(int a, int n){   Ass: Given a & n, calculate & return a^n
    if(n==0){return 1}
    return Pow(a,n-1) * a
}
```

Base: Pow(a,n) = Pow(a,n-1) * a

Pow(a,0) = Pow(a,-1) * a *

Pow(a,1) = Pow(a,0) * a ✓

Tracing:

```
long Pow(int a=5, int n=3){ a          → 125
    x1. if(n==0){return 1}
    ✓2. return Pow(a,n-1) * a
                          25 * 5
}
```

```
long Pow(int a=5, int n=2){ b
    x1. if(n==0){return 1}
    ✓2. return Pow(a,n-1) * a
                        5 * 5
}
```

```
long Pow(int a=5, int n=1){ c
    x1. if(n==0){return 1}
    ✓2. return Pow(a,n-1) * a
                        1 * 5
}
```

```
long Pow(int a=5, int n=0){ d
    1. if(n==0){return 1}
    2. return Pow(a,n-1) * a
}
```

Way:2 Note: Don't worry about overflows

Pow(a, 6) = a * a * a * a * a * a ⇒ Pow(a,5) * a

Pow(a, 6) = Pow(a,3) * Pow(a,3)

Pow(a, 10) = Pow(a,5) * Pow(a,5)

Pow(a, 9) = Pow(a,4) * Pow(a,4) * a

Pow(a, 11) = Pow(a,5) * Pow(a,5) * a

Pow(a, n) = if n is even

Pow(a,n) = Pow(a, n/2) * Pow(a, n/2)

else

Pow(a, n) = Pow(a, n/2) * Pow(a, n/2) * a

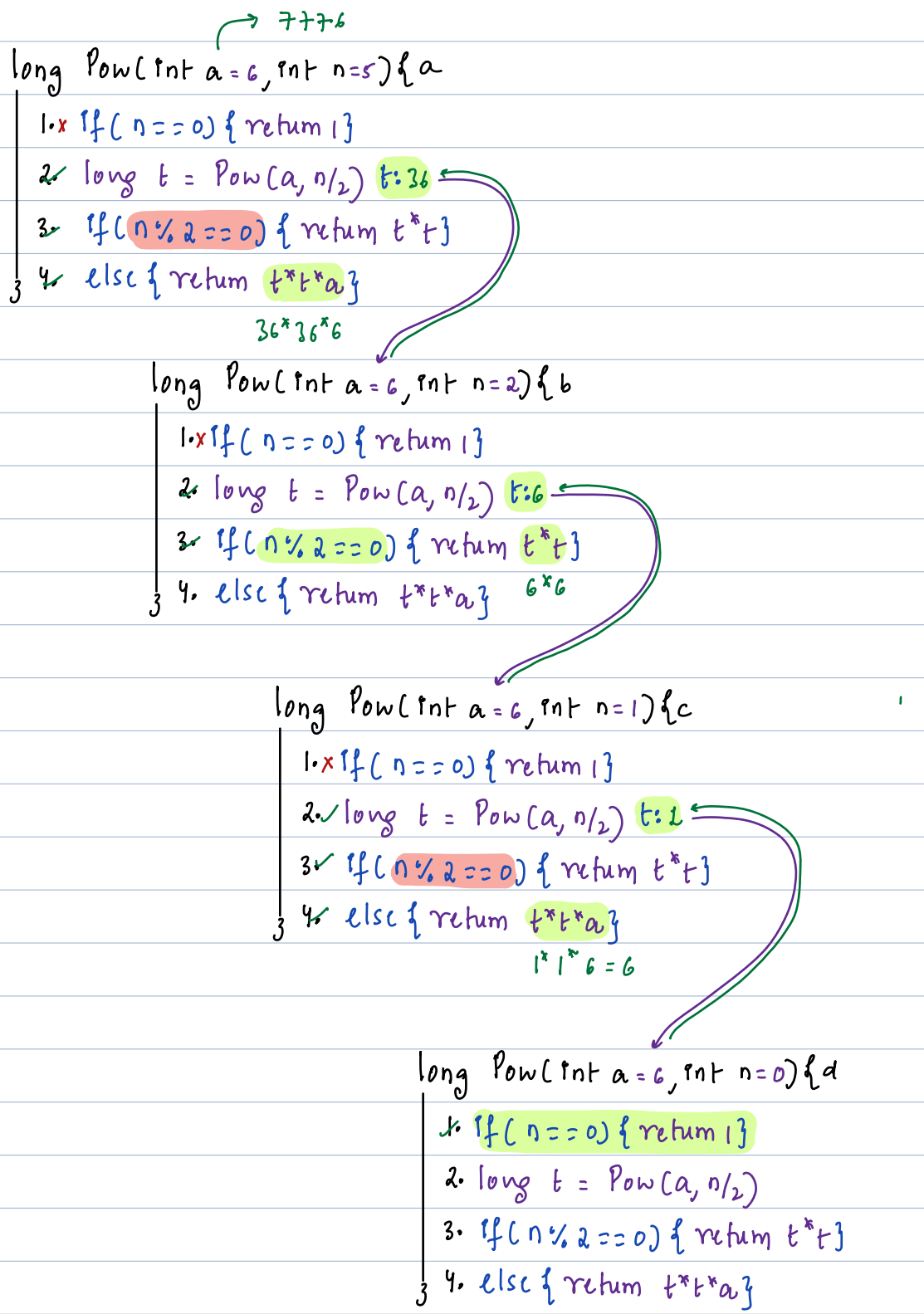
```
long Pow(int a, int n){   Ass: Given a, n calculate & return a^n
    if(n==0){return 1}
    if(n%2==0){
        return Pow(a, n/2) * Pow(a, n/2)
    }else{
        return Pow(a, n/2) * Pow(a, n/2) * a
    }
}
```

Note: Pow(a, n/2) is SubProblem, which is calculated multiple times

Calculate once, store ans in variable & re-use it.

```
// Way:3 Note: Dont worry about overflows
long Pow(int a, int n){
    if(n==0){return 1}
    long t = Pow(a, n/2)
    if(n%2==0){return t*t}
    else{return t*t*a}
}
```

// Tracing


7776
long Pow(int a=6, int n=5){a
1.x if(n==0){return 1}
2. long t = Pow(a, n/2) t:36
3. if(n%2==0){return t*t}
4. else{return t*t*a}
36*36*6
long Pow(int a=6, int n=2){b
1.x if(n==0){return 1}
2. long t = Pow(a, n/2) t:6
3. if(n%2==0){return t*t}
4. else{return t*t*a}
6*6
long Pow(int a=6, int n=1){c
1.x if(n==0){return 1}
2. long t = Pow(a, n/2) t:1
3. if(n%2==0){return t*t}
4. else{return t*t*a}
1*1*6=6
long Pow(int a=6, int n=0){d
1. if(n==0){return 1}
2. long t = Pow(a, n/2)
3. if(n%2==0){return t*t}
4. else{return t*t*a}

Pow with modules Note: Take care of overflows:

Q: Given a, n, m, calculate & return $a^n \% m$

Constraints:

$1 <= a <= 10^9$

$0 <= n <= 10^9$

$2 <= m <= 10^9+1$

Formula:

$ele \% m = \{0 \ldots m-1\}$

$(a*b) \% m = \{a\%m * b\%m\} \% m$

$Pow(a, n, m) = a^n \% m$ $\quad Pow(a, n/2, m) = a^{n/2} \% m$

if n is even:

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

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

$a^n \% m = (Pow(a, n/2, m) * Pow(a, n/2, m)) \% m$

if n is odd:

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

$a^n \% m = (Pow(a, n/2, m) * Pow(a, n/2, m) * a) \% m$

$Pow(a, n, m)$ = if n is even:

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

if n is odd:

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

Note: $Pow(a, n/2, m)$ is SubProblem, which is calculated multiple times

Calculate once, store ans in variable & re-use it

Final exp:

$Pow(a, n, m)$ = $t = Pow(a, n/2, m)$

if n is even = $(t*t) \% m$

if n is odd = $(t*t*a) \% m$

Examples:

pow(4, 8, 7) = Calculate = $4^8 \% 7$, expected ans = 2

Using exp:

t = pow(4, 4, 7); $t = 4^4 \% 7 = 4$

n is even: return $[t*t] \% m = [4*4] \% 7 = [16 \% 7] = 2$

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

```
long powmod(int a, int 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*a)%m}
}
```

Constraints & Overflow:

$1 <= a <= 10^9$ int $\approx 10^9$ long $\approx 10^{18}$

$0 <= n <= 10^9$ max m = 10^9+1

$2 <= m <= 10^9+1$ ele % m = [0 m-1] max = 10^9

Doubts: max $m = 10^9+1$

ele % m = [0.. m-1]

= [0.. 10^9+1-1]

= 10^9

```
long powmod(int a, int n, int m){
    if(n==0){return 1}
    long t = pow(a, n/2, m) // t = [a^(n/2)] % m, max t ≈ 10^9
    if(n%2==0){return (t*t)%m} // (t*t)%m = [10^9 * 10^9 ≈ 10^18]%m ≈ 10^9
    else{
        return (t*t*a)%m // t*t*a]%m = [10^9 * 10^9 * 10^9 ≈ 10^27]%m = 10^9
                         // overflow:
        return [ (t*t)%m * a%m ]%m // no overflows.
                // [ [10^9 * 10^9 = 10^18]%m = 10^9  * 10^9 = 10^18]%m ≈ 10^9 ]
    }
}
```

Final Code:

if a < 0 =

Calculate a%m In Java? = (a%m + m)%m ?

Constraints:

$-10^9 <= a <= 10^9$

$0 <= n <= 10^9$

$2 <= m <= 10^9+1$

```
long powmod(int a, int n, int m){
1. if(n==0){ return 1}
2. long t = [powmod(a, n/2, m) + m] %m   // It works for -ve as well
3. if( n%2==0){ return t*t%m }            <-> no need in python
4. else{
5. } return {[(t*t)%m * a%m] %m + m} %m
}
```

Calculate $-2^5 \% 10 = -32 \% 10 = 8$.

8:

```
long powmod(int a=-2, int n=5, int m=10){
✗ if(n==0){ return 1}
2. long t = [powmod(a, n/2, m) + m] %m : T=[4+10]%10 = 4
✗ if( n%2==0){ return t*t%m }
4. else{
5. } return [(t*t)%m * a%m] %m + m} %m
}
```

```
long powmod(int a=-2, int n=2, int m=10){
  1. if(n==0){ return 1;}
  2. long t = [powmod(a, n/2, m) * m] %m
  3. if( n%2==0){ return t*t%m }
  4. else{
  5.   return [(t*t)%m * a%m] %m * m }%m
}
```

8

t : [8 * 10] % 10 * t = 8

= 8*8%10 = 4

```
long powmod(int a=-2, int n=1, int m=10){
  1. if(n==0){ return 1;}
  2. long t = [powmod(a, n/2, m) * m] %m
  3. if( n%2==0){ return t*t%m }
  4. else{
  5.   return [(t*t)%m * a%m] %m * m }%m
}
```

1

t : [1 * 10] % 10 = 1

= [1*1 %10 * -2%10] %10 * 10 }%10 = 8

```
long powmod(int a=-2, int n=0, int m=10){
  1. if(n==0){ return 1;}
  2. long t = [powmod(a, n/2, m) * m] %m
  3. if( n%2==0){ return t*t%m }
  4. else{
  5.   return [(t*t)%m * a%m] %m * m }%m
}
```