Quote →

> Learn continually - there's always "one more thing" to learn!
>
> — Steve Jobs —

Today's Content

→ pow (a, n) {

// three ways.

}

→ pow (a, n, p)

→ T.C of recursive codes

Q) Given a, n. find a^n using recursion. (No overflow), $N \geq 0$.

Eg:

a	n	a^n
2	5	32
3	4	81

Assumption: Calculate & return the value of a^n.

```
int pow (a, n){
    if(n == 0) return 1
    return (pow(a, n-1) * a ).
}
```

$$a^n = \underbrace{a * a * a * \cdots a * a}_{n\text{-times}}$$

$$a^n = a^{n-1} * a$$

$$[pow(a,n) = pow(a,n-1) * a]$$

```
int pow2 (a, n) {
    if(n == 0) return 1
    if (n%2 == 0) {
        return { pow2(a, n/2) * pow2(a, n/2) }
    }
    else {
        return {pow2(a, n/2) * pow2(a, n/2) * a }
    }
}
```

$$a^{10} = a^5 * a^5$$

$$a^{14} = a^7 * a^7$$

$$a^{11} = a^5 * a^6$$

$$= a^5 * a^5 * a$$

$$a^{13} = a^6 * a^6 * a.$$

if n is even.

$$pow(a,n) = pow(a, n/2) * pow(a, n/2)$$

if n is odd.

$$pow(a,n) = pow(a, n/2) * pow(a, n/2) * a.$$

Observation: Same problems, calling only once.

```
int power3 ( a , n){
    if(n == 0) return 1
    int p = pow3(a, n/2);
    if (n%2 == 0){
        return p*p;
    }
    else {
        return p*p*a;
    }
}
```

Tracing

```
int power3 ( a=2, n=9 ){
    if(n==0) return 1
    int p = pow3(a, n/2);
    if (n%2 ==0){
        return p*p;
    }
    else {
        return p*p*a;
    }
}
```

16*16*2

512

16.

```
int power3 ( a=2 , n=4 ){
    if(n==0) return 1
    int p = pow3(a, n/2);
    if (n%2 ==0){
        return p*p;
    }
    else {
        return p*p*a;
    }
}
```

4.

```
int power3 ( a=2, n=2) {
    if(n==0) return 1
    int p = pow3(a, n/2);
    if (n%2 ==0){
        return p*p;
    }
    else {
        return p*p*a;
    }
}
```

2.

```
int power3 ( a=2, n=1 ){
    if(n==0) return 1
    int p = pow3(a, n/2);
    if (n%2 ==0){
        return p*p;
    }
    else {
        return p*p*a;
    }
}
```

1

```
int power3 (a=2, n=0){
    if(n==0) return 1
    int p = pow3(a, n/2);
    if (n%2 ==0){
        return p*p;
    }
    else {
        return p*p*a;
    }
}
```

Q4 Given a, n, m. Calculate $a^n \% m$.

Note → Take care of overflows.

Constraints → $\begin{bmatrix} 1 <= a <= 10^9 \\ 1 <= n <= 10^9 \\ 2 <= m <= 10^9 \end{bmatrix}$

Ass^m: Calculate $a^n \% m$ & return.

$$a^n = a^{n/2} * a^{n/2}$$

$$a^n \% m = [a^{n/2} * a^{n/2}] \% m$$

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

powmod(a, n/2, m)

```
powmod ( int a , int n, int m){
    if (n == 0) return 1

    long p = powmod(a, n/2, m)
    if (n % 2 == 0){
        return ((p * p) % m)
    }
    else {
        return (p * p * a) % m
    }
}
```

$p = [\underline{a^{n/2} \% m}]$ → [No case of overflow. It will further break down]

max value of $p = m-1$

$p \approx 10^9$

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

$= 10^9 * 10^9 * 10^9$

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

return $(((p * p) \% m * a \% m) \% m$.

```
powmod ( a , n , m ) {   // Assumption → calculate & return a^n % m
    if (n == 0) return 1
    long p = powmod( a, n/2 , m) ;
    if ( n % 2 == 0) {
        return (p * p) % m
    }
    else {
        return [(p * p) % m * a] % m
    }
}
```

$a^n \% m$

[Fast exponentiation]

Wrong ⟹ $\rightarrow (p \% m) * (p \% m) * (a \% m)$

$(m-1) * (m-1) * (m-1)$

$(\underbrace{10^9 * 10^9 * 10^9}_{\text{overflow.}}) \% m$

T.C for Recursive Code Using Recursive Relation :-

① int sum (N){
    if (N==1){return 1 }
    return $(\underbrace{\text{sum}(N-1)}_{f(N-1)} + N)$
}

// Time taken to calculate sum(N) = f(N)

Time " to " sum(N−1) = f(N−1)

$$f(N) = f(N-1) + 1$$
$$f(N-1) = f(N-2) + 1$$
$$= f(N-2) + 2$$
$$f(N-2) = f(N-3) + 1$$
$$= f(N-3) + 3$$
$$f(N-3) = f(N-4) + 1$$
$$= f(N-4) + 4$$

After K-steps.

$$f(n) = f(n-K) + K \quad , \quad f(1) = 1$$

// $n - K = 1 \Rightarrow K = n-1.$

$$f(n) = f(n-(n-1)) + n-1$$
$$= f(\cancel{n}-\cancel{n}+1) + n-1$$
$$= f(1) + n - 1. = \cancel{1}+n\cancel{-1} = n.$$

$\boxed{T.C \rightarrow O(N)}$

② int fact (N){

if (N == 1) {return 1 }

return ($\underbrace{\text{fact}(N-1)}_{f(N-1)}$ * N)

}

Time taken to calculate fact (N) = f(N)

" " " " fact(N-1) = f(N-1)

$f(N) = f(\cancel{N-1}) + 1$

$f(\cancel{N-1}) = f(\cancel{N-2}) + 1$

$f(\cancel{N-2}) = f(\cancel{N-3}) + 1$

$f(\cancel{N-3}) = f(\cancel{N-4}) + 1$

| | |

| | |

$f(\cancel{1}) = f(\cancel{1-1}) + 1$

$f(N) = \underbrace{1+1+1+1+1 ---- 1}_{N}$

$f(N) = N$ $\boxed{T.C \to O(N)}$

task1 → 20 min

task2 → 30 min

total time taken = 50 min

③

```
int pow1 (a, n){
    if (N==0){return 1 }
    return pow(a,n-1) * a
}
```

$\underbrace{pow(a,n-1)}_{f(n-1)}$

Time taken to calculate $pow1(a,n) = f(n)$

" " " " $pow1(a,n-1) = f(n-1)$

$f(n) = f(n-1) + 1$, $f(0) = 1$

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

$f(n) = f(n-2) + 2$

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

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

$f(n-3) = f(n-4) + 1$

$f(n) = f(n-4) + 4$

After k^{th} step, base cond^n occurs

$[f(n) = f(n-k) + k]$

$n - k = 0 \Rightarrow \boxed{k = n}$

$f(n) = f(n-n) + n$

$f(n) = 1 + n$

$\boxed{T.C \to O(N)}$

(5)
```
int pow3(a, n){
    if (n==0){return 1}
    p = pow3(a, n/2)
    if (n%2 == 0){return p*p}
    else {return p*p*a}
}
```

// Time taken to calculate pow3(a,n) = $f(n)$

" " " " pow3(a, n/2) = $f(n/2)$

multiplication will take $O(1)$

→ $p*p \to O(1)$

$p*p*a \to O(1)$

$f(n) = f(n/2^1) + 1$, $f(1) = 1$

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

$f(n) = f(n/2^2) + 2$

$f(n/4) = f(n/8) + 1$

$f(n) = f(n/2^3) + 3$

$f(n/8) = f(n/16) + 1$

$f(n) = f(n/2^4) + 4$

After K steps, base condition

$f(n) = f(n/2^K) + K$

// $\frac{n}{2^K} = 0$ ✗ $\quad \frac{n}{2^K} = 1 \Rightarrow n = 2^K$

$\boxed{K = \log_2 n}$

$f(n) = f\left(\frac{\cancel{n}}{\cancel{n}}\right) + \log_2 n$

$\boxed{T.C \to O(\log_2 n)}$

```
int powmod(a, n, m){
    if (n==0){return 1}
    long p = powmod(a, n/2, m)
    if (n%2 == 0) {return (p*p)%m}
    else {return [(p*p)%m * a]%m}
}
```

$\boxed{T.C \to O(\log_2 N)}$

(4)

```
int pow2 (a, n) {
    if (n == 0) return 1
    if (n % 2 == 0) {
        return {pow2(a, n/2) * pow2(a, n/2)}
    }
    else {
        return {pow2(a, n/2) * pow2(a, n/2) * a}
    }
}
```

// Time taken to calculate pow2(a, n) = f(n)

" " " " pow(a, n/2) = f(n/2)

$f(n/2) + f(n/2) + 1$

$f(n/2) + f(n/2) + 1$

Doubt

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

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

$$\frac{n}{2} * \frac{1}{2} = n/4$$

$$f(n) = 2f(n/2) + 1 \quad , f(1) = 1$$

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

$$f(n) = 2\left[2f(n/4) + 1\right] + 1$$

$$f(n) = 2^2 f(n/2^2) + 2^2 - 1$$

$$f(n/4) = 2f(n/8) + 1$$

$$f(n) = 2^2\left[2f(n/8) + 1\right] + 3$$

$$= 2^3 f(n/2^3) + 2^3 - 1$$

After K steps,

$$f(n) = 2^k f(n/2^k) + 2^k - 1$$

$$// \quad n/2^k = 1 \qquad n = 2^k \Rightarrow \boxed{k = \log_2 n}$$

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

$$= n \cdot f(1) + n - 1$$

$$= n + n - 1 = 2n - 1$$

$$\boxed{T.C \rightarrow O(N)}$$