Today's agenda

- Pow(a, n)
- TC & SC of Recursion

Q) Given a and N, calculate a^n.

Ex:

a	n	
2	3	→ 8
3	5	→ 243

```
int Pow (int a, int n) {
1 if (n == 1) { return a; }
2 int temp = Pow (a, n-1);
3 return temp * a;
}
```

Faith: Given a and n, calculate and return a^n.

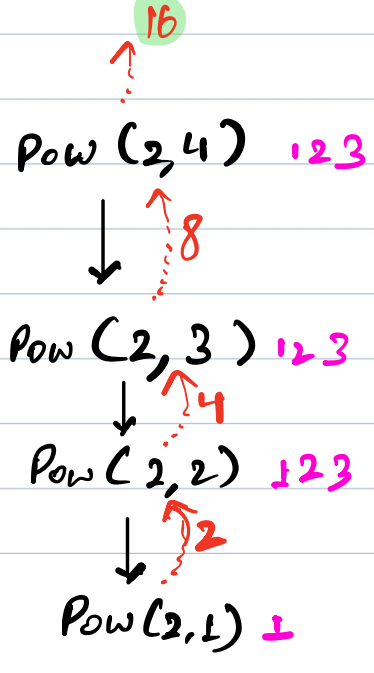
Main logic:

$a^n \rightarrow temp * a$

$a^{n-1} \rightarrow temp$

Base case:

```
if (n == 1) { return a; }
```



T.C. of 1 function: O(1) → x

No. of function: N. → y

T.C of recursion: x * y = O(1) * N = O(N)

S.C of recursion: O(1) * N = O(N)

// do something better than O(n).

→ $a^n = a^{n-1} * a$

↳ $a^n = a^{n/2} * a^{n/2}$ if n is even

ex: $2^8 = 2^4 * 2^4$

$2^9 = 2^4 * 2^4 * x$

↳ $a^n = a^{n/2} * a^{n/2} * a$ if n is odd

$2^9 = 2^4 * 2^4 * 2$


```
int Pow(int a, int n){
  if(n==1){return a;}
```

Faith: Given a and n, calculate and return a^n.

Main logic:

a^n → n is even = temp * temp

a^n → n is odd = temp * temp * a

$a^{n/2}$ → temp

```
  int temp = Pow(a, n/2);
  if(n%2==0){return temp * temp;}
  else {return temp * temp * a;}
}
```

Base case:

```
if(n==1){return a;}
```

```
int Pow (int a, int n){
1 if (n==1) { return a; }

2 int temp = Pow(a, n/2);

3 { if (n%2==0){return temp * temp;}
  else {return temp * temp * a;} }

}
```

a = 2 N = 37

Pow(2, 37) 123 — 262144 * 262144 * 2

↓ 262144

Pow(2, 18) 123

↓ 512

Pow(2, 9) 123

↓ 16

Pow(2, 4) 123

↓ 4

Pow(2, 2) 123

↓ 2

Pow(2, 1) 1

overall T.C:

T.C of 1 function: O(1)

No. of function: $O(\log_2 N)$

$\simeq O(\log N)$

overall S.C: $O(1) * \log N \simeq O(\log N)$

if n is negative

$2^{-37} \Rightarrow 2^{37}$ ⇝ recursive soln → ans

⇓

return $\frac{1}{ans}$

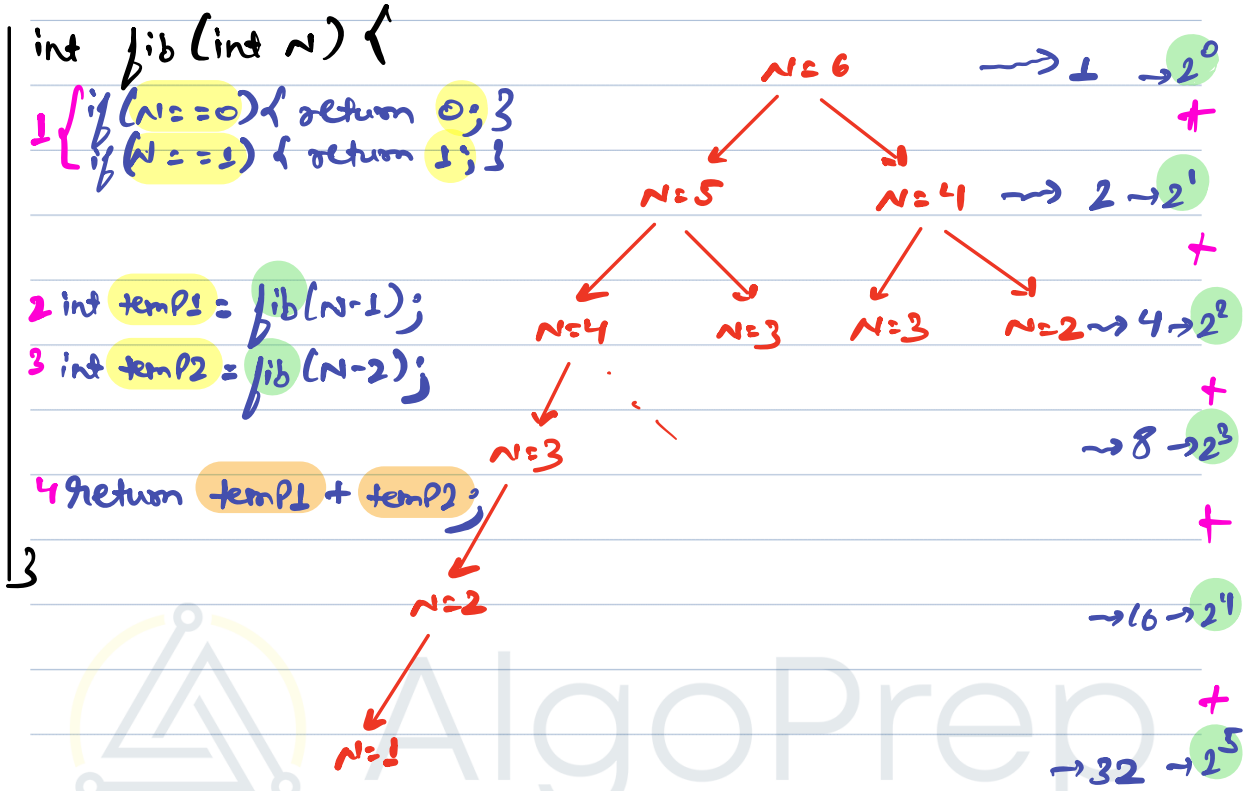
→ $a^n = a^{n/2} * a^{n/2}$ → $O(\log N)$ → most optimal here.

$a^n = a^{n/3} * a^{n/3} * a^{n/3}$

$a^n = a^{n/4} * a^{n/4} * a^{n/4} * a^{n/4}$

$a^n = a^{n/5} * a^{n/5} * a^{n/5} * a^{n/5} * a^{n/5}$

$a^n = a^{n/n} * a^{n/n} * a^{n/n} * a^{n/n} \ldots\ldots a^{n/2}$ → $O(N)$ → ideal



$2^0 + 2^1 + 2^2 + 2^3 \; - \; - \; - \; - \; + 2^{N-1}$

(*2, *2, *2)

Sum of G.P: $a * \frac{(r^n - 1)}{r-1}$

$= 1 * \frac{2^N - 1}{2-1} = 2^N - 1$

Overall T.C: $O(1) * 2^N \approx O(2^N)$

No. of Calls = x

Count of levels = n

no. of functions = x^n

Break till 3:49 pm.

Q) Given an array, check if it is Palindrome or not?
(recursion)

Ex: MALAYALAM → true

Ex: a a b b a → false

//idea 1

→ MALAYALAM
→ MALAYALAM → true

a a b b a → false
a b b a a

//idea 2

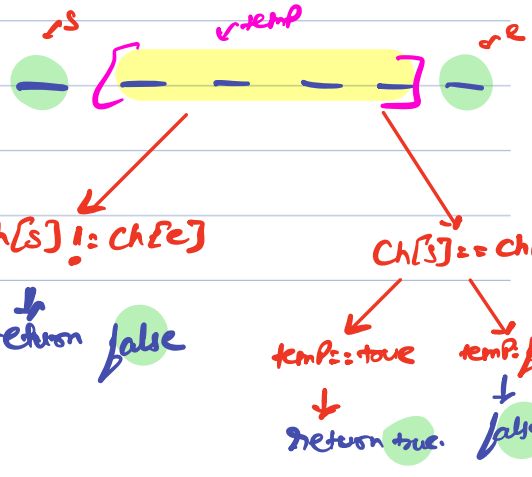
T.C: O(1) * N/2 ≈ O(N)
S.C: O(1) * N/2 ≈ O(N)

```
boolean isPalindrom(char[] ch, int s, int e){
    if (s == e) {return true;}
    if (s > e) {return true;}

    if (ch[s] == ch[e]){
        boolean temp = isPalindrome(ch, s+1, e-1);
        return temp;
    }
    else {return false;}
}
```

Faith: Check and return whether ch array is Palindrome betn s (0) to e (n-1).

main logic:



base case:

```
boolean isPalindrom(char[] ch, int s, int e){
1  if(s==e){return true;}
   if(s>e){return true;}
   if(s>=e){return true;}
   if(ch[s] == ch[e]){
2    boolean temp = isPalindrome(ch,s+1,e-1);
     return temp;
   }
3  else{return false;}
}
```

Ch: M A L A Y A L A M (0 1 2 3 4 5 6 7 8)

isPalindro(0 8) ↓ 2a b — temp: true — ↑ true

isPalindrome(1 7) ↓ 2a b — ↑ true

isPalindrom(2 6) ↓ 2a b — ↑ true

isPalindrome(3,5) ↓ 2a b — ↑ true

isPalindrome(4,4) ↓ 1

e

s

Ch: M A L A A L A M (0 1 2 3 4 5 6 7)

isPalin(0,7) → true

isPalind(1,6) → true

isPalind(2,5) → true

isPalin(3,4) → true

isPalin(4,3)

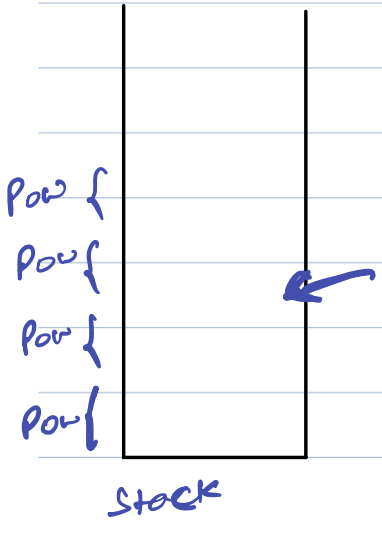
Ch: M A L A A L B M
0 1 2 3 4 5 6 7

isPalin(0,7) → false
↓ ↑ false
isPalin(1,6)

write iterative code → for (int i=0; i<n; i++) {

3

write recursive code: → within 1 function space complexity is like iteration.
↳ you create multiple instance of same function.



space will be considered in space complexity.

S.C: (Space used in 1 function) & (max no. of function in stack at any point of time.)