Recursion - 2

Question

Given 2 integers a & n, find a^n using recursion.

eg $a = 2$, $n = 3$ $\quad 2^3 = 2 \times 2 \times 2 = 8$

1. pow(a, n) → a * a * a * ..... * a (n times)
2. pow(a,n) = pow(a,n-1) * a
3. pow(a,0) = 1

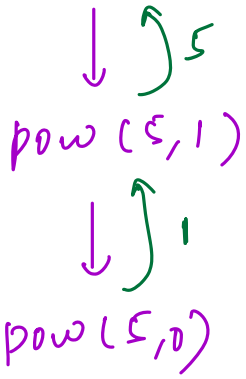
Code

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

pow(5,3) → pow(5,2) → ... (n times)

TC = O(N)

SC = O(N)



Alternative approach

$a^6 = a^3 \times a^3$

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

$a^7 = a^3 \times a^3 \times a$

$a^n = a^{n/2} \times a^{n/2}$ n is even

$= a^{n/2} \times a^{n/2} \times a$ n is odd

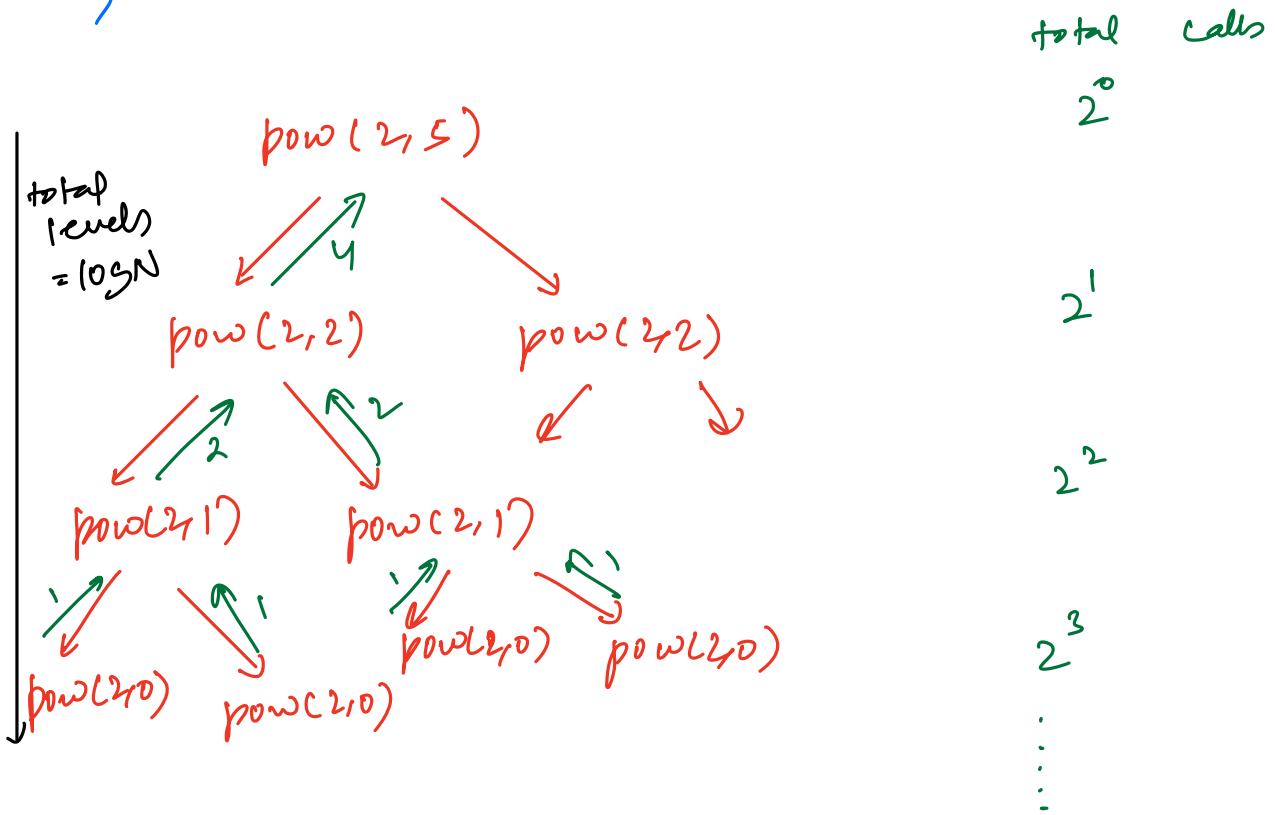
1. pow(a,n) $= a^n$

2. pow(a,n) =
   - pow(a,n/2) * pow(a,n/2) — n: even
   - pow(a,n/2) * pow(a,n/2) * a — n = odd

3. pow(a,0) = 1

Code

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



total calls

2^0

2^1

2^2

2^3

⋮

total calls = $2^0 + 2^1 + 2^2 + \ldots\ldots$ (log n times)

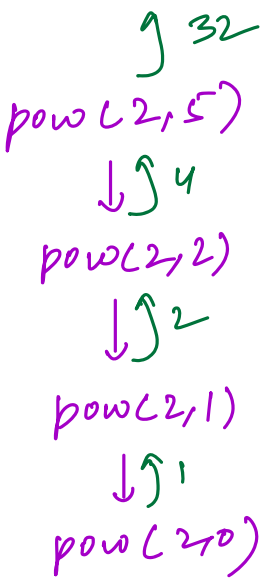
$= 2^0\left(\frac{2^{\log n} - 1}{2-1}\right) \rightarrow n-1$

TC = O(N)

SC = O(logN)

Optimize (fast Exponentiation)

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



TC = O(logN)

SC = O(logN)

Question

Given an array of integers. Write a recursive fn to print all elements.

A = [1 2 3 4 5]

output: 1 2 3 4 5

1. print (A , 0, n-1)

2. print (A , 0 , n-1) → print(A[0]) print (A, 1, n-1)

3. print (A , n-1, n-1) → print (A[n-1]) print (A , n, n-1)
do nothing

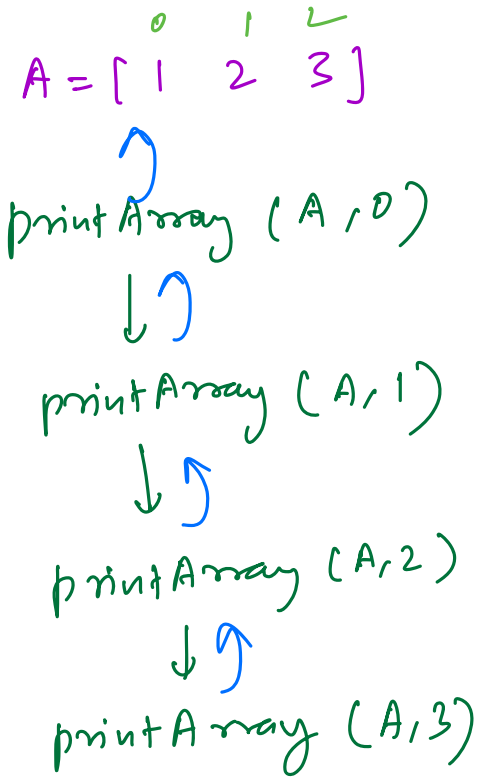
printArray (A,0)

Code

```
void printArray ( A , index) {
    if ( index == A.size()) {
        return
    }
    print( A[index])
    printArray ( A, index +1)
}
```

TC = O(N)

SC = O(N)



output: 1 2 3

Quiz

findSum (A, 0, n-1) = A[0] + findSum (A, 1, n-1)

```
int findSum ( A, index) {
    if ( index == A.size() ) {
        return 0
    }
    return A[index] + findSum ( A, index+1)
}
```

Question

Given an array of N elements & an integer B. find all the indices in array where element = B & return it.

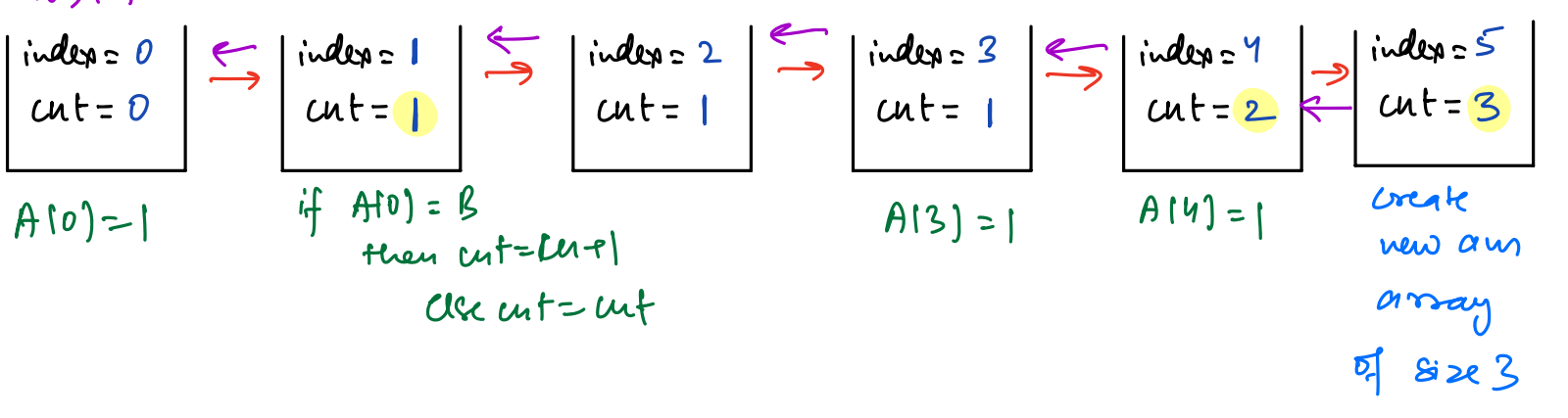
A = [4 5 3 1 5 4 5] (indices 0 1 2 3 4 5 6) B = 5

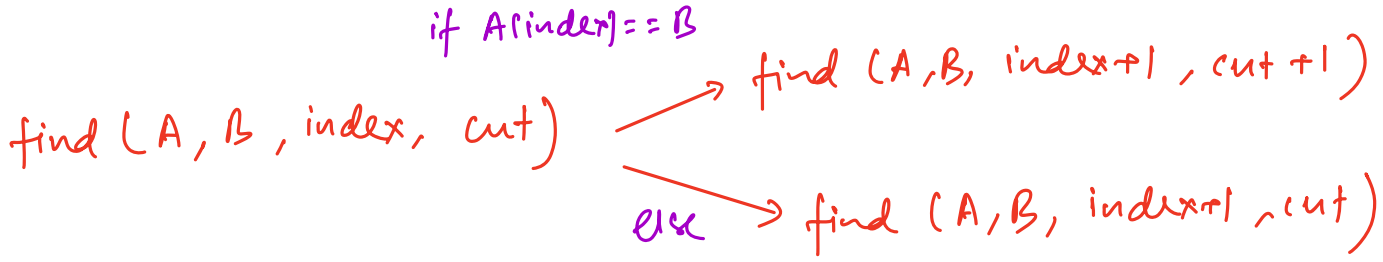
ans : [1 4 6]

A = [1 2 3 1 1] (indices 0 1 2 3 4) B = 1

ans : [0 3 4]

A = [1 2 3 1 1] (indices 0 1 2 3 4)






```
if ( index == A.size() ) {
    return new ans[cut]
}
```

find (A, B, 0, 0)

Code

```
int[] find ( int[] A , int B , int index , int cut ) {
    if ( index == A.size() ) {
        return new int[cut] ;
    }
    if ( A[index] == B ) {
        int[] ans = find ( A , B, index+1, cut+1 );
        ans[cut] = index
    }
    else {
        int[] ans = find ( A , B, index+1, cut)
    }
```

return ans

}

Question

Given a string, write a recursive function to check if its palindrome.

s = "radar" True

s = "area" false

isPalin(s, i, j) ⇒ if (s[i] != s[j]) return false

else if (isPalin(s, i+1, j-1)) return true

else return false

isPalin(s, i, j) ⇒ return true

base: i >= j

isPalin(s, 0, n-1)

Code

```
bool isPalin(s, i, j){
    if(i >= j) return true
    if(s[i] != s[j]) return false
    return isPalin(s, i+1, j-1)
}
```

TC = O(N)

SC = O(N)

0	1	2	3	4	5	6	7	8
r	a	c	a	d	a	c	a	r

isPalin(s,0,8) ↑ true
isPalin(s,1,7) ↑ true
isPalin(s,2,6) ↑ true
isPalin(s,3,5) ↑ true
isPalin(s,4,4)

0	1	2	3	4	5
r	a	x	b	a	r

isPalin(s,0,5) ↑ false
isPalin(s,1,4) ↑ false
isPalin(s,2,3) ↑ false

total levels = N/2

Quiz

```
solve (N) {
    if (N == 0) return
    print(N)
    solve (N-1)
    print(N)
}
```

solve(N=3)
↓↑
solve(N=2)
↓↑
solve(N=1)
↓↑
solve(N=0)

output: 3 2 1 1 2 3

Quiz

```
solve (N) {
    if (N == 0) return
    print(N)
    solve (N-1)
}
```

solve(N=-3)
↓
solve(N=-4)
↓
solve(N=-5)
↓
⋮ never become 0

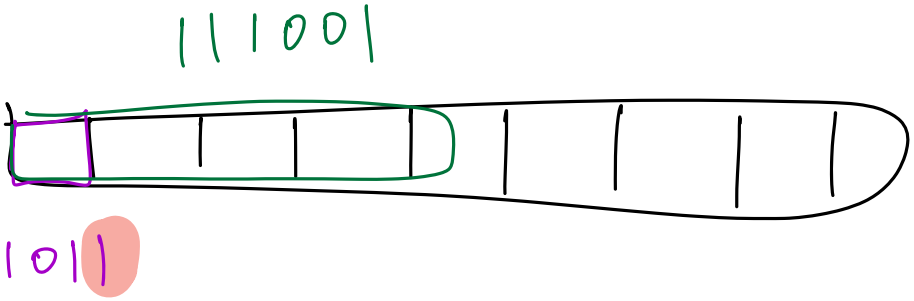
output: -3 -4 -5 -6 . . . . . .

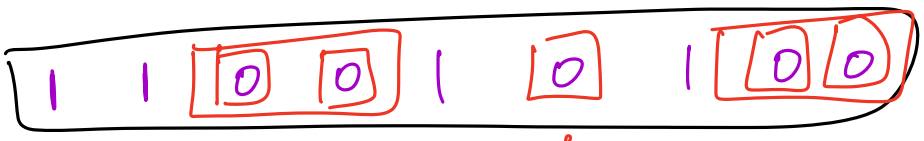
Stack overflow error

Doubt

32 bit

find sum of all subarray ORs.





2 1 2

$\frac{2(2+1)}{2}$ $\frac{1(1+1)}{2}$ $\frac{2(2+1)}{2}$

3 1 3

$\frac{n(n+1)}{2} - (3+1+3)$

$TC = O(32 \times N)$