Recursion - 2

Today's Agenda:

- Fast Exponentiation
- Recursion on Arrays
- Check Palindrome
- Problems

Bhargav

- Google (2+ yrs)
- Scaler (1+ yrs)

Question: Given 2 integers a & n, find a^n using recursion

Ex: a = 2 n = 3

O/P = 8

```
for i -> 1 to n:        } Iteration
    res *= a
```

Recurrence relation:

pow(a, n) = pow(a, n-1) * a

Revision

1) Assume the smaller problem is solved.
2) Use smaller problem to solve bigger problem
3) Base Case

```
func pow(a, n) {
    // Base case
    if (n == 0) return 1;
    return pow(a, n-1) * a;
}
```

pow(2, 3) = 8
↓
2 * pow(2, 2) = 4
↓
2 * pow(2, 1) = 2
↓
2 * pow(2, 0) = 1
↓
return 1

TC = O(# func call * TC of pow)
= O(N * 1) = O(N)

SC = O(N)

Optimised Approach

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

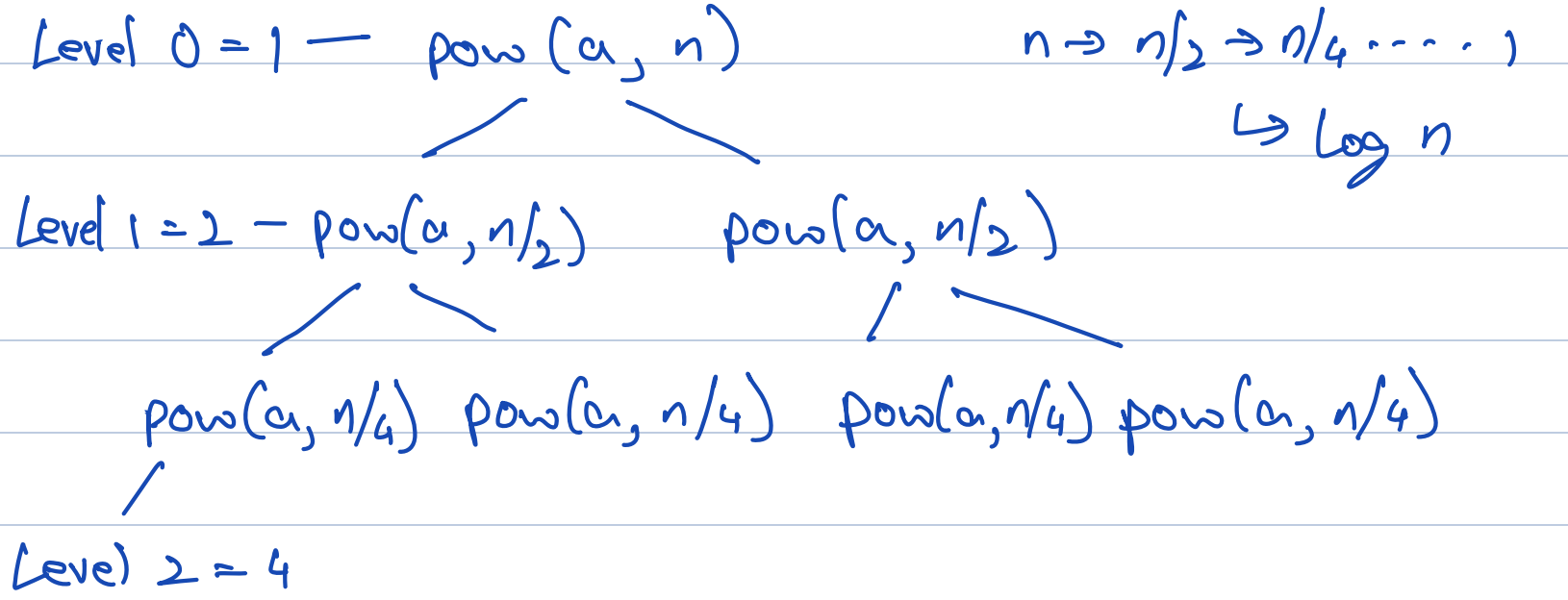
$pow(a, n) = pow(a, n-1) * a$

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

(or) $a^n = a^{n/2} * a^{n/2}$ (Even)

$a^n = a^{n/2} * a^{n/2} * a$ (Odd)

```
func pow(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
}
```



nth level = 2^n Func^n calls.

Total levels = $\log n + 1 \approx \log n$

Calculate total func^n calls

$$\overbrace{1 + 2 + 4 + \cdots + 2^{\log n}}^{\log n}$$

$$2^{\log n} = n$$

$1 + 2 + 4 \cdots n$

Sum = $\frac{a*(r^n - 1)}{r-1}$, $a = 1$, $r = 2$, $n = \log n$

func calls = $\frac{1*(2^{\log n} - 1)}{2-1} = n - 1$

T.C = O(# func calls * TC of each func)

= O((n-1)*1)

T.C = O(n)

S.C = O(Depth of recursive stack + SC of each func call).

= O(log n + 1)

S.C = O(log n)

Fast Exponentiation

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

pow(a, n)
↓
pow(a, n/2)
↓
pow(a, n/4)
↓
pow(a, n/8)
⋮
↓
pow(a, 1)
↓
pow(a, 0)

Total levels = $\log n + 1 \approx \log n$

Total func calls = $\log n$

T.C = O(# func calls * TC of each func call)

= O(log n * 1)

T.C = O(log n)

S.C = O(Depth + S.C of each)

= O(log n + 1)

S.C = O(log n)

Recursion on Arrays

Question: Given an array of integers, write a recursive function to print all elements of the array.

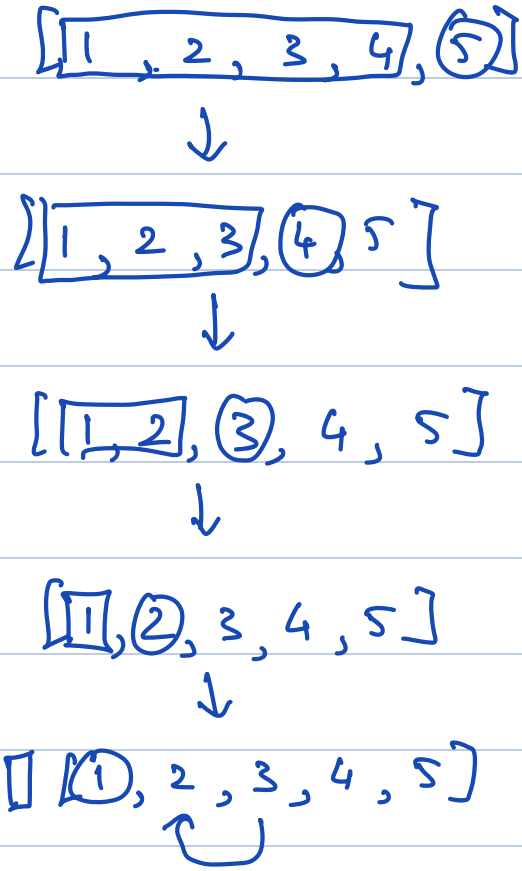
Recurrence relation

printN(N) = printN(N-1) , print(arr[N])

↳ index to be printed

```
func printN(int[] arr, int N){
    // Base case
    if(N<0) return;
    printN(arr, N-1);
    print(arr[N]);
}

main(){
    arr = {1, 2, 3, 4, 5};
    printN(arr, arr.length-1);
}
```



T.C = O(N) S.C = O(N)

Question: Given an array of integers, write a recursive function to find the maximum element of the array.

arr = [5 , 8 , 2 , 10 , 3]

Recurrence relation

max_arr(arr, N) = max(max_arr(arr, N-1), arr[N])

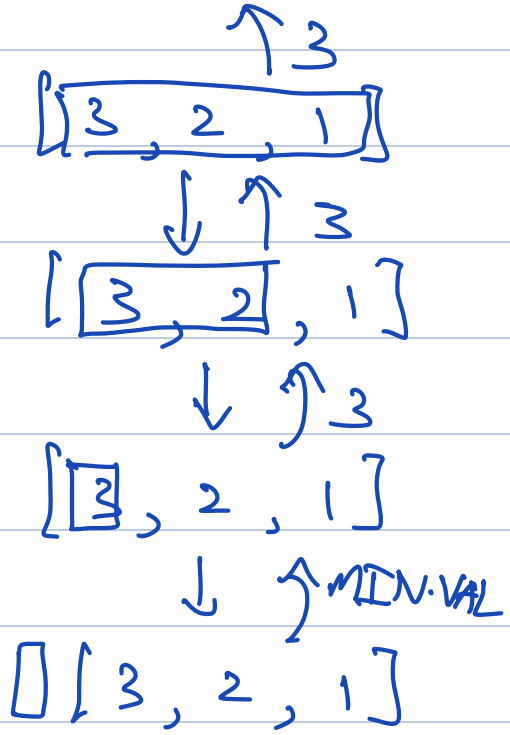
index ↵

```
func max_arr(int[] arr, int N){
    // Base case
    if(N<0) return Integer.MIN_VAL;
    return max(max_arr(arr, N-1), arr[N]);
}
```

T.C = O(N)

S.C = O(N)

Python – sys.minint → -10**18



Quiz 1: Sum of array elements

```
func sum_arr(int[] arr, int N){
    // Base case
    if(N<0) return 0;
    return sum_arr(arr, N-1) + arr[N];
}
```

Question: Given an array of N integers and a target B, find all the indices at which B occurs in the array.

```
      0  1  2  3  4  5  6
ex:  A = [4, 5, 3, 1, 5, 4, 5]
     B = 5
     O/P = [1, 4, 6]
```

```
          0  1  2  3  4
Quiz 2:  A = [1, 2, 3, 1, 1]    B = 1
         O/P = [0, 3, 4]
```

Recurrence relation

allidx(N) = allidx(N-1) + if(arr[N] == B)

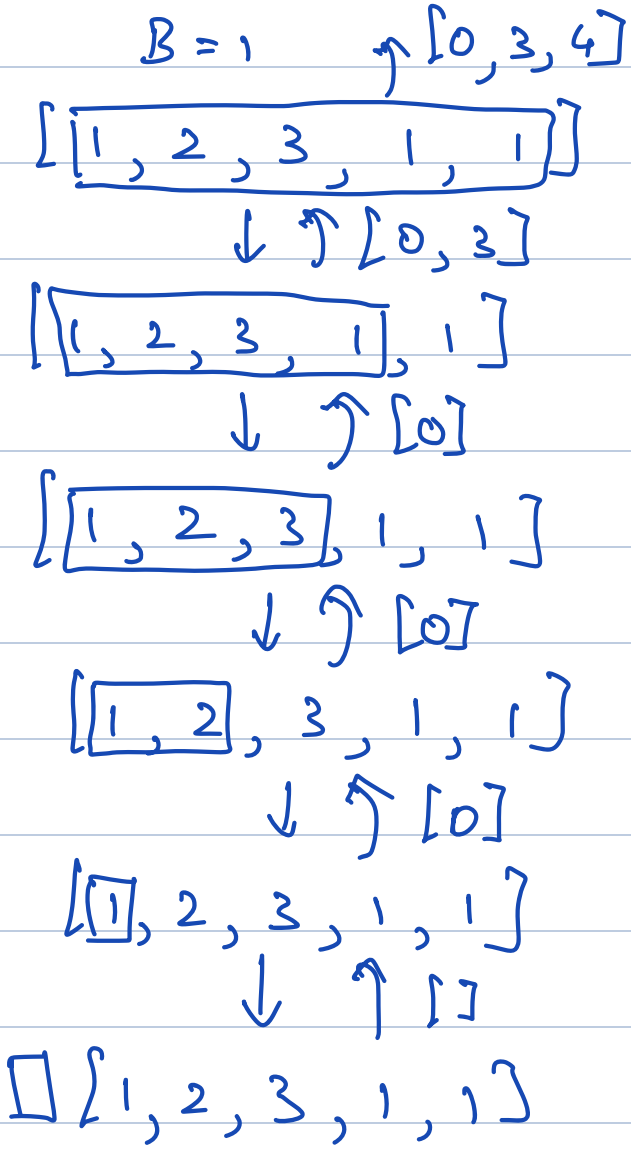
```
func allidx(arr, N, B) {
    // Base case
    if(N<0) : return [];
    ans = allidx(arr, N-1, B);
    if(arr[N] == B) : ans.add(N);
    return ans;
}
```

T.C = O(N)

S.C = O(Depth + S.C of each)

= O(N + N)

S.C = O(N)



Question: Given a string, write a recursive function to check if it is a palindrome.

ex: s = "radar"

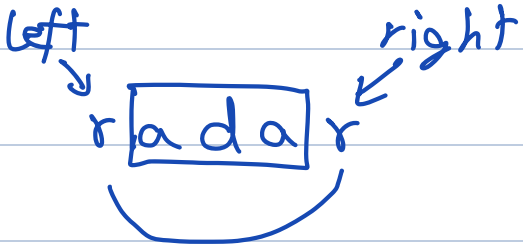
o/p = true

TC = O(N)

S.C = O(N)

s = "area"

o/p = false.



if (left == right): return ispalin(left+1, right-1)

right left

radar → Break = left > right condition ↳ return true

```
func isPalin(String str, left, right){
    // Base case
    if(left > right): return true;
    if(str[left] == str[right]):
        return isPalin(str, left+1, right-1);
    }
    return false;
}
```

Quiz 3:

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

N = 3

↓

N = 2

↓

N = 1

↓

N = 0

O/P

3 2 1 1 2 3

Quiz 4:

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

N = -3

↓

N = -4

↓

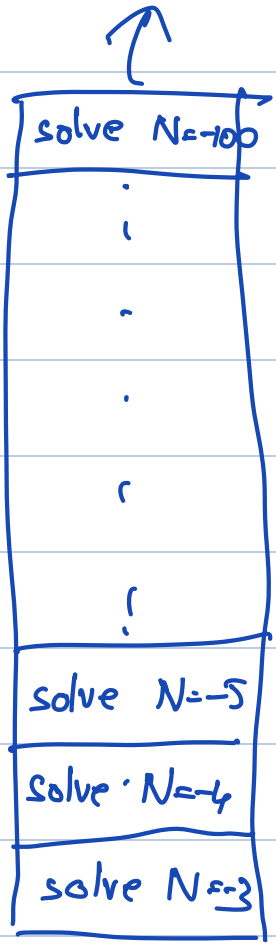
N = -5

↓

. . .

O/P

-3 -4 -5 . . .



01 101 11

[1, 5, 3]

PF = [01 101 111]

sum = 0

for i → 0 to N:

res = 0

for j → i to N:

res |= arr[j]

sum += res

1 5 3

001 101 011

↓

Subarrays it is part ((i+1)*(n-i))

Sum = 0

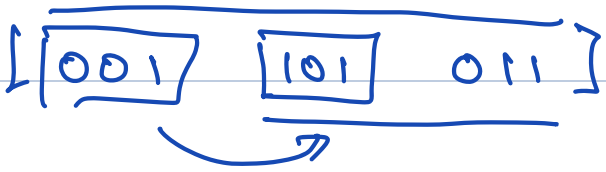
for i → 0 to N-1:

for j → 0 to 31:

if (checkSetBit (arr[N], j)):

indi = 2^j

sum += indi * (i+1) * (N-i)


001
101
011