Recursion3

[Quiz1](#_a8qjnckufzwn)

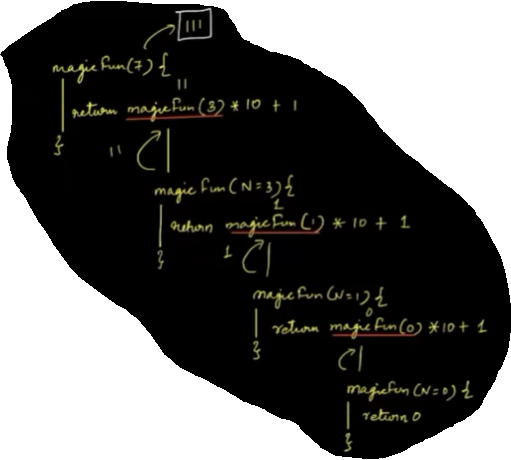
[Quiz 2](#_kfmeb3b42cmd)

[Problem 1:](#_o6hxfqmbs6y5)

[Problem 2:](#_ds62433hyauf)

[Problem 3](#_tkbd0ickou0r)

# Quiz1



Find the output of below

Int magicFun(int N){

if(N == 0){

return 0;

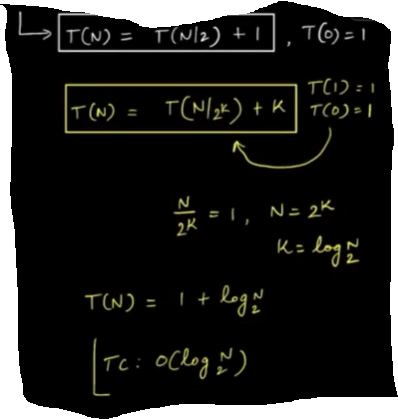
}else{

Return magicFun(N/2)\*10 + (N%2);

}

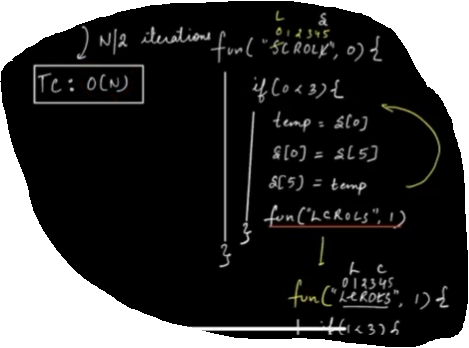
}

For N = 7



T.c:

# Quiz 2



Find the output for fun(“SCROLL”)

Void fun(char s[], int x){

print(s);

Char temp;

if(x < s.len/2){

Temp = s[x]

S[x] = s[s.len-x-1]

S[s.len-x-1] = temp;

fun(s, x+1)

}

}

T.c = O(N)

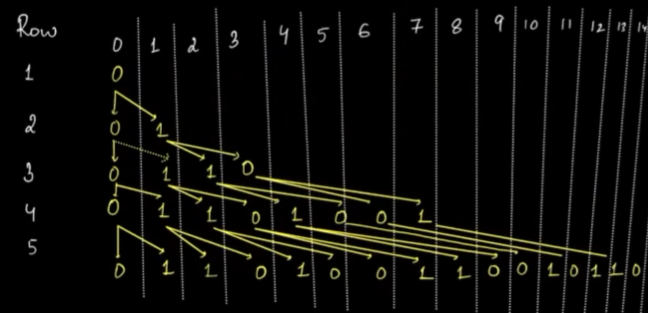
# Problem 1:

Kth Symbol

Each row is generated by replacing all elements of the previous row.

1 -> 1 0

0 -> 0 1



Input : N and K f{find the value at Kth index in Nth row.}

4 5 -> return 0

5 11 -> return 1

4 9 -> Not possible

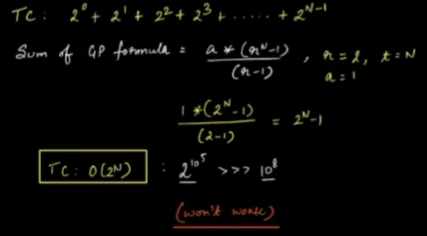
**Brute Force:**

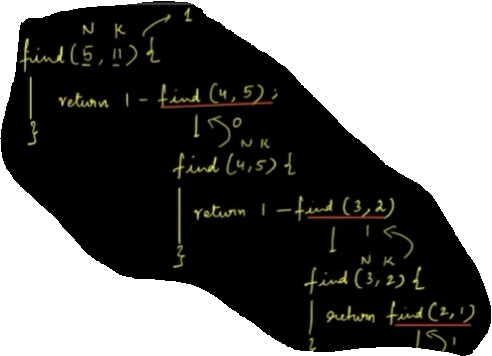
Construct Nth Row and find Kth Element.

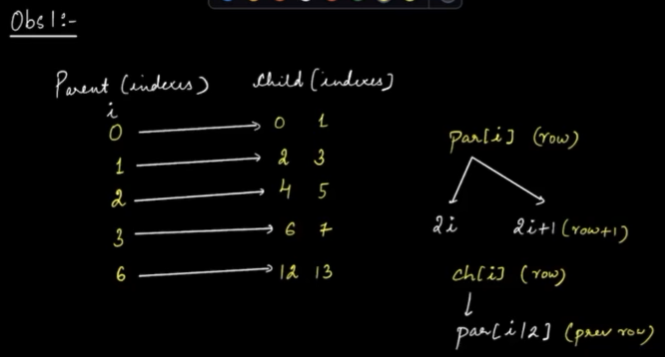
Nth -> N-1th -> N-2nd -> ….. -> 3 -> 2 -> 1

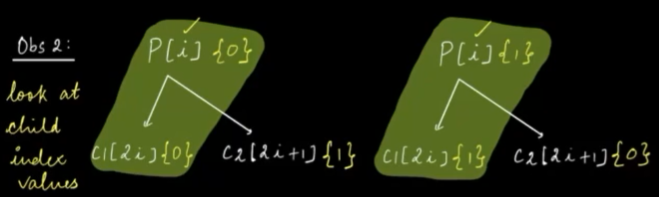
2^2 2^1 0

For any ith Row = 2^(i-1)



**Optimised Approach** 

****

****

**Note:**

**From above two observation:**

**For any child,** if the index is even : value will be the same as parent.

If the child index is odd : value will be opposite of parent.

Now, To find element at: Nth Row and Kth Column

Pseudocode:

*int find(int N, int K){*

*if(k==0){*

*return 0;*

*}*

*if(k%2 == 0){*

*return find(N-1, k/2);*

*}else{*

*return 1 - find(N-1, k/2); // 1 - What is returned from the parent=Opp value*

*}*

*}*

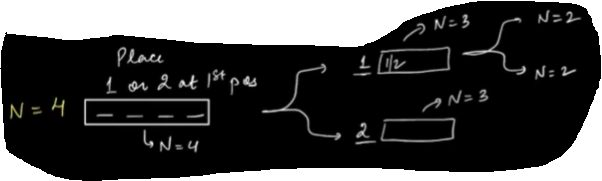
*T.C : T(K) = T(K/2) +1 | T.C : O(log2K)*

# Problem 2:

Given N digits Print all N digits formed by only 1 and 2 in increasing order of numbers.

N = 2

1 1

1 2

2 1

2 2

N = 3

1 1 1

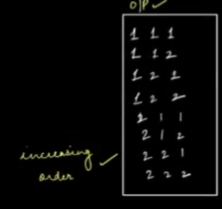
1 1 2

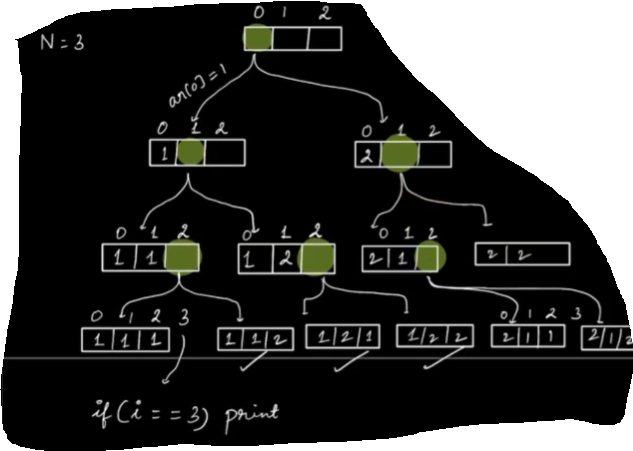
1 2 1

1 2 2

……..

Idea:- Generate all N digit numbers using Recursion.





*Void printAll (int arr[], int i, int N){*

*if(i == N){*

*Iterate and print array // and.push(arr);*

*return;*

*}*

*// At ith index i’ve two choices*

*Arr[i] = 1 // Choice 1*

*printAll(arr, i+1, N);*

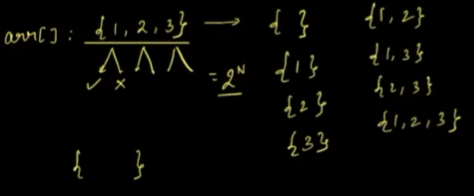
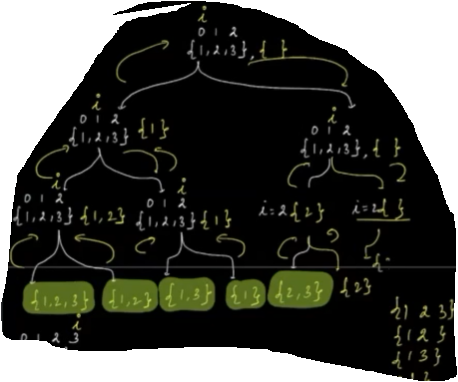
*Arr[i] =2 // Choice 2*

*printAll(arr, i+1, N);*

*}*

# Problem 3

Given an array. Print all the subsets using recursion

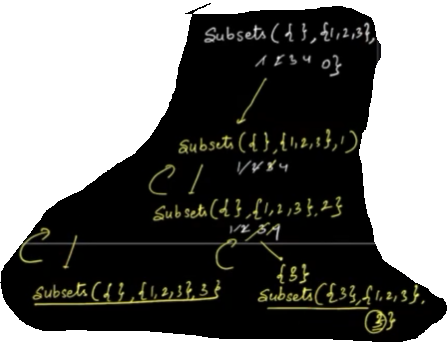
 

Every element in the array has 2 options

1. Considered in the subset
2. Not considered in the subset.

if(i == N){ // done forming all the subsets}

list<list<int> ans;

void subsets(list<int> curr, list<int> arr, int i){

if(i == arr.length){

ans.add(curr);

return;

}

subsets(curr, arr, i+1);

curr.push(arr[i]);

subsets(curr, arr, i+1);

}