

## T.C. of TO H

$$T(N) = \underline{2T(N-1)} + 1 \quad \text{--- (1)}$$

put  $N = N-1$  in eq (1)

$$\underline{T(N-1)} = 2T(N-2) + 1$$

$$\begin{aligned} T(N) &= 2 \left( 2T(N-2) + 1 \right) + 1 \\ &= 4T(N-2) + 2 + 1 \end{aligned}$$

put  $N = N-2$  in eq (1)

$$\underline{T(N-2)} = 2T(N-3) + 1$$

$$\begin{aligned} T(N) &= 4 \left( 2T(N-3) + 1 \right) + 2 + 1 \\ &= 8T(N-3) + 4 + 2 + 1 \end{aligned}$$

put  $N = N-3$  in eq (1)

$$\underline{T(N-3)} = 2T(N-4) + 1$$

$$T(N) = 8 \left( 2T(N-4) + 1 \right) + 4 + 2 + 1$$

$$T(N) = 16T(N-4) + 8 + 4 + 2 + 1$$

$$= 2^4 T(N-4) + 2^3 + 2^2 + 2^1 + 2^0$$

$\downarrow$   
 $\vdots$   
 $\vdots$   
K times

$$T(N) = 2^K T \left( \underbrace{N-K}_0 \right) + 2^{K-1} + 2^{K-2} + 2^{K-3} \dots 2^0$$

$\stackrel{=}{\circlearrowleft}$   
 $K = N$

$$T(N) = 2^N T \left( \cancel{N-1}^1 \right) + 2^{N-1} + 2^{N-2} \dots 2^0$$

$$= 2^N + 2^{N-1} \dots 2^0$$
$$= \underbrace{2^0 + 2^1 + 2^2 \dots}_{N+1 \text{ terms}} \underbrace{2^{N-1} + 2^N}_{2^N}$$

$N+1$  terms

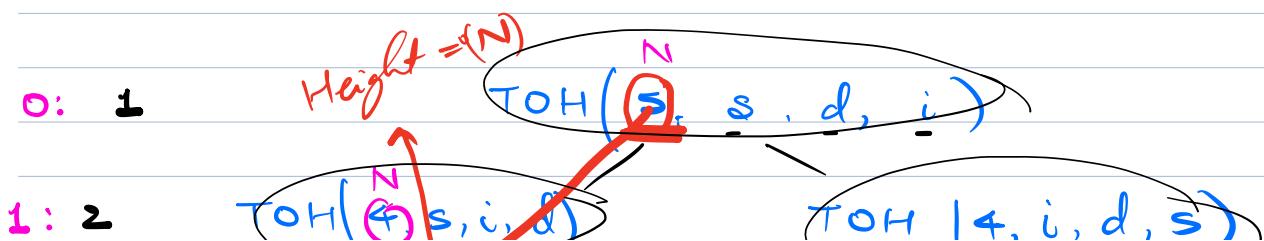
$$a = 1$$

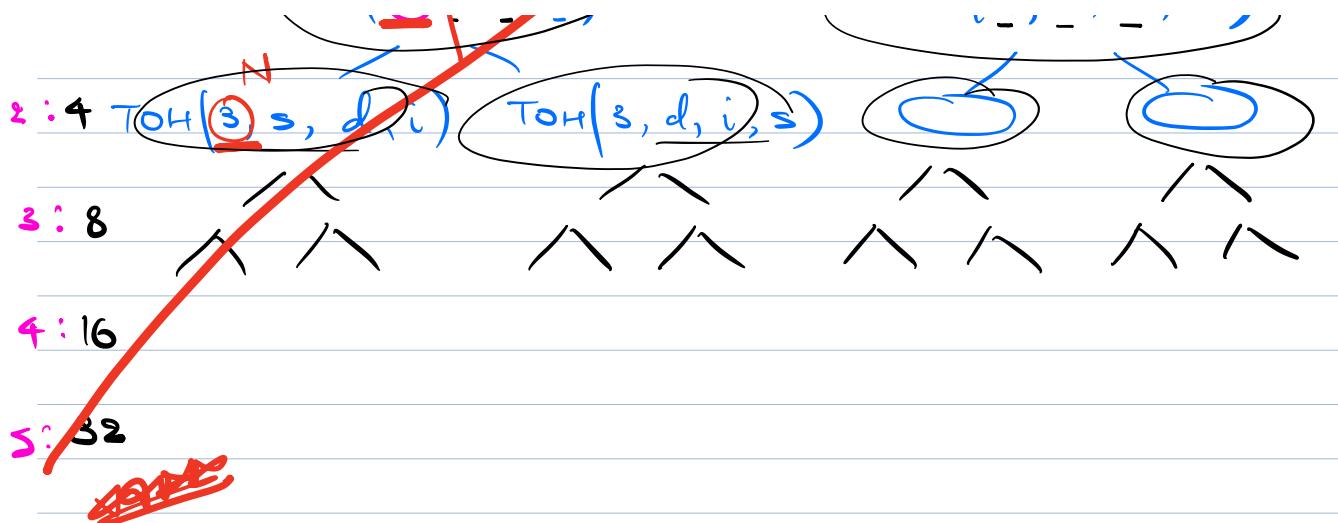
$$r = 2$$

$$T(N) = \frac{1}{r} \left( \frac{2^{(N+1)} - 1}{2 - 1} \right) \approx \underline{\underline{2^{N+1}}}$$

$$T.C. = O(\underline{\underline{2^N}})$$

$$2^{N+1} = \frac{2 \times 2^N}{x}$$





Dime Complexity of any recursive function =  $\left( \frac{\text{No. of f^n calls}}{\text{No. of nodes in the recursive tree}} \right) \times \text{T.C. of every recursive call.}$

$$\leq \left( \frac{2^0}{1} + \frac{2^1}{2} + \frac{2^2}{4} + \frac{2^3}{8} + \dots + \frac{2^{N-1}}{2^N} \right) \times O(1)$$

$$= \frac{(2^{N-1})}{2-1} = \underline{2^N} \times O(1)$$

$$\underline{\text{T.C.}} = O(2^N)$$

void printTOH (N, src, dest, int) {

```
if (N == 1) {
    print ("Ring 1 : src → dest");
}
```

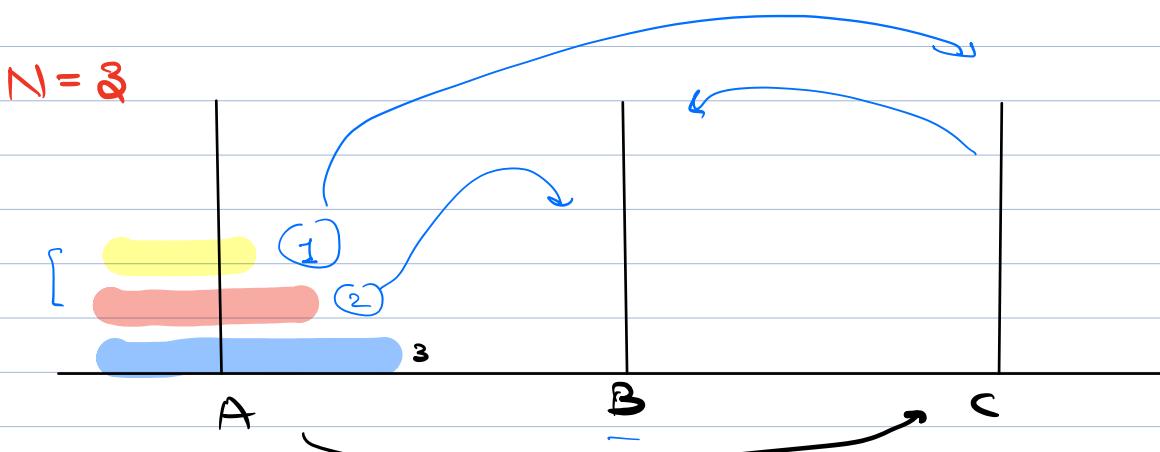
4  
`return'; ..

print TOH( N-1, src, int, dest );

print (Ring N: src  $\rightarrow$  dest); ✓

print TOH( N-1, int, dest, src );

5



print TOH( 3, A, C, B )

print( 2, A, B, C )

print

print TOH( 2, B, C, A )

TOH( 1, A, C, B ) print: TOH( 1, C, B, A )

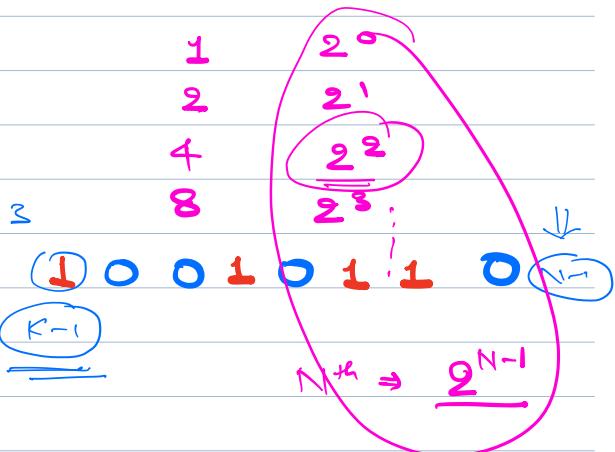
A  $\rightarrow$  B

Q2

Given  $N \neq K$ . Return the value at the  $(K+1)^{th}$  index in  $N^{th}$  row.  
(Assume that the input is always valid)

for the next row, change 0 to 01 & 1 to 10  
in the previous row.

<u>N</u>															
<u>1</u> :	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<u>2</u> :	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
<u>3</u> :	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
<u>4</u> :	0	1	1	0	1	0	0	1	0	0	1	0	0	1	0
<u>5</u> :	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14



## Brute force

→ generate  $N$  rows  $\Rightarrow$

→ Return  $A [N-1] \{K-1\}$

$$\begin{aligned} T.C. &= \underline{\underline{2^0}} + \underline{\underline{2^1}} + \underline{\underline{2^2}} \dots \dots \dots \underline{\underline{2^{N-1}}} \\ &= \underline{\underline{2^N - 1}} \end{aligned}$$

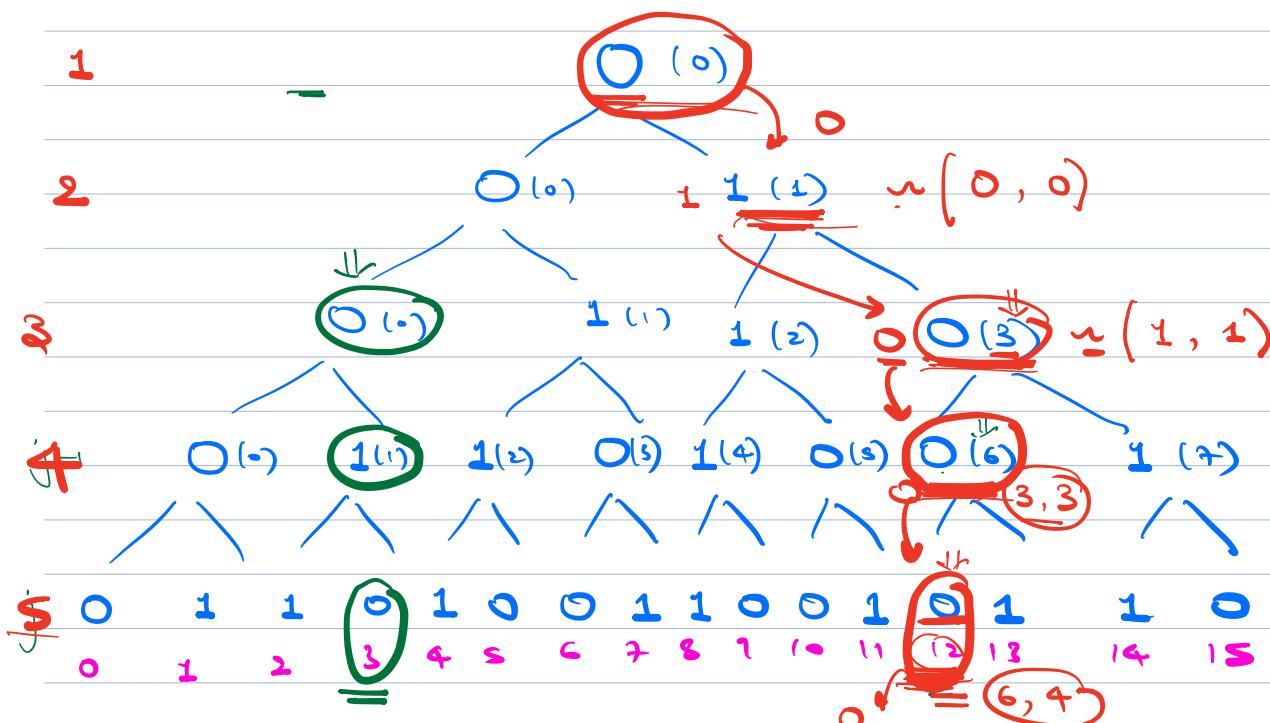
<u>N</u>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<u>1</u> :	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<u>2</u> :	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
<u>3</u> :	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
<u>4</u> :	0	1	1	0	1	0	0	1	0	0	1	0	0	1	0	0
<u>5</u> :	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



1)  $i^{\text{th}}$  index of row  $j$  generates

$\underline{2^i}, (2^{i+1})^{\text{th}}$  index of row  $j+1$

2)  $i^{\text{th}}$  index of  $j^{\text{th}}$  row can be generated of  $(j-1)^{\text{th}}$  row using  $(i/2)^{\text{th}}$  element



0(i) (parent)

$$\frac{12}{2} = 6$$
$$\frac{13}{2} = 6$$

0( $2i$ )  
Even  
=  $\frac{1}{2}(2i+1)$  (child)  
Odd

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

find ( $\underline{\underline{5}}, \underline{\underline{12}}$ )  
find ( $\underline{\underline{4}}, \underline{\underline{6}}$ )

$$= \begin{cases} 1 & (\text{even}) \\ 0 & (\text{odd}) \end{cases}$$

$\curvearrowleft$  find(3, 3)  
 $\curvearrowleft$  find(2, 1)

int find ( N, K ) {  $\curvearrowleft$   $O(\log \underline{K})$  }

if ( $K == 0$ )  $\curvearrowleft$  return 0;

int parentNode = find ( N-1, K/2 );

if ( $K \% 2 == 0$ )  $\curvearrowleft$

return parentNode;

$\curvearrowleft$  else  $\curvearrowleft$

return (1 - parentNode);

$\curvearrowleft$

$$\begin{aligned} 0 &\rightarrow 1 - 0 = 1 \\ 1 &\rightarrow 1 - 1 = \end{aligned} \Rightarrow$$

$\curvearrowleft$

Any element of  $N^{th}$  row  
 $\Downarrow$

at max  $N$  recursive calls

T.C. =  $\underline{O(N)}$

Gray Code

$\sim = \Delta$   $\equiv$

Different binary sequence where consecutive no. only differ by 1 bit

$\underline{\underline{N=3}}$

0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

$$\begin{aligned} & 2 \times 2 \times 2 \\ & 0 \quad - \quad 1 \\ & 2^3 = 8 \end{aligned}$$

Given  $N$ , generate all gray code no's

of  $N$  bits

$\underline{\underline{N=1}}$

0  
1

$\underline{\underline{N=2}}$

0	0
0	1
1	0
1	1

$\underline{\underline{N=3}}$

000	001	011	010
000	001	010	011
000	011	010	001
000	010	001	011
001	011	010	001
001	010	001	011
011	001	010	001
010	011	001	011

$\rightarrow$

0 0 0 0 0 0 0  
0 0 0 0 0 0 1  
0 0 0 0 0 1 1  
0 0 0 0 1 1 0  
0 0 0 0 1 1 1  
0 0 0 1 1 1 1  
0 0 1 1 1 1 0  
0 0 1 1 1 1 1

1	1	0	0	
1	1	0	1	
1	1	1	1	
1	1	1	0	
1	0	1	0	
1	0	1	1	
1	0	0	1	
1	0	0	0	

not cleared though

## Code

Ass:  $\text{grayCode}(N)$  : return the list of possible gray codes using n sets

Main Logic :  $\text{grayCode}(N)$  uses  $\text{grayCode}(N-1)$

Base Case :  $N=1$  { "0", "1" }  
 $N=0$  { }

$T(N)$   
list < string >  $\text{grayCode}(N)$  {

O(1) if ( $N == 0$ ) { return { }; }

O(1) if ( $N == 1$ ) { return { '0', '1' }; }

O(1)  $\underline{\text{list}} < \text{string} >$  codes =  $\text{grayCode}(N-1)$

O(1) list < string > ans = new list<string>();

$O(2^N)$  [ for ( $i = 0$ ;  $i < \text{codes.length}()$ ;  $i++$ ) {  
 ans. add ('0' + codes[i]);  
 } ]

$O(2^{N-1})$  [ for ( $i = \text{codes.length}() - 1$ ;  $i > 0$ ;  $i--$ ) {  
 ans. add ('1' + codes[i]);  
 } ]

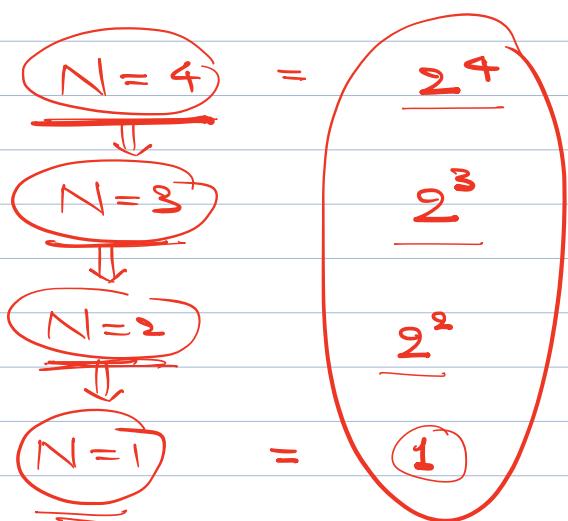
0() return ans;

b

T.C. if one recursive call

$$= 2^{N-1} + 2^{N-1}$$

$$\text{T.C.} = \underline{\underline{O(2^N)}}$$



$$N = 1 + 2^2 + 2^3 + 2^4 \dots \dots 2^N + 2^{1-2^1}$$

$$2^0 + 2^1 + 2^2 + \dots + 2^N - 2$$

↓

$$\left( \frac{2^{N+1} - 1}{2} \right) - 2$$

$$= 2^{N+1} - 3$$

$$\text{T.C.} = O(2^{N+1})$$

$$\sum GP = a \frac{\left(\frac{2^{N+1} - 1}{2} - 1\right)}{2 - 1}$$

PS session on Thursday

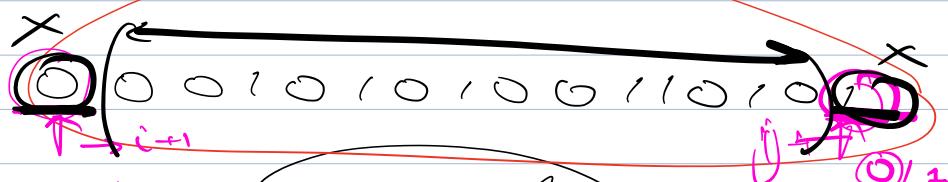
↳ Thread

↳ Topic, problem

1 problem

in 1 message

Doubts

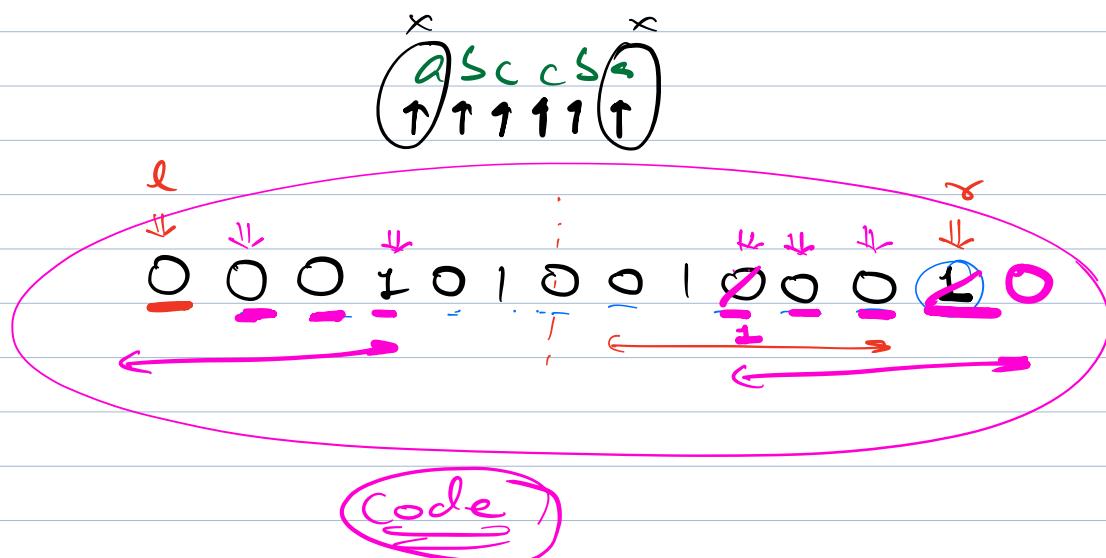
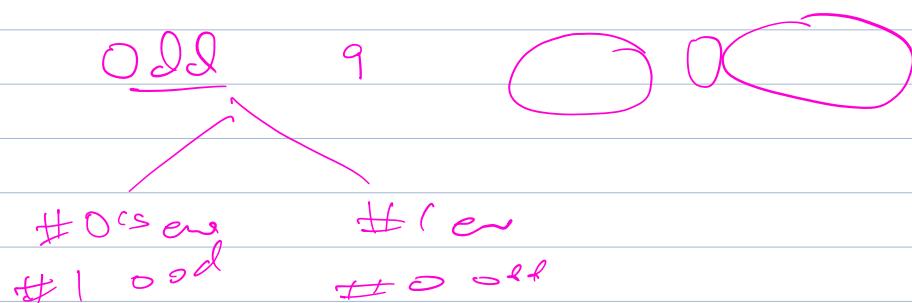
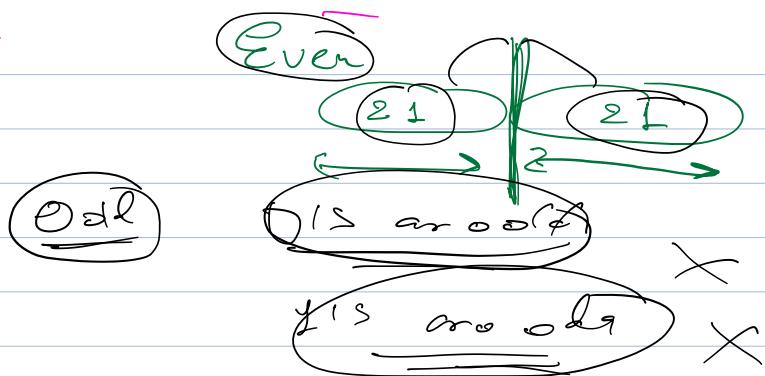


10  
r

min sum of 1's

$N = 8$

17



$$T(n) = T(n-1) + T(n-2) + O(1)$$

$$\underline{\underline{n-1 \approx n-2}}$$

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

$$\underline{\underline{2^n}}$$

