Recursion Output Questions

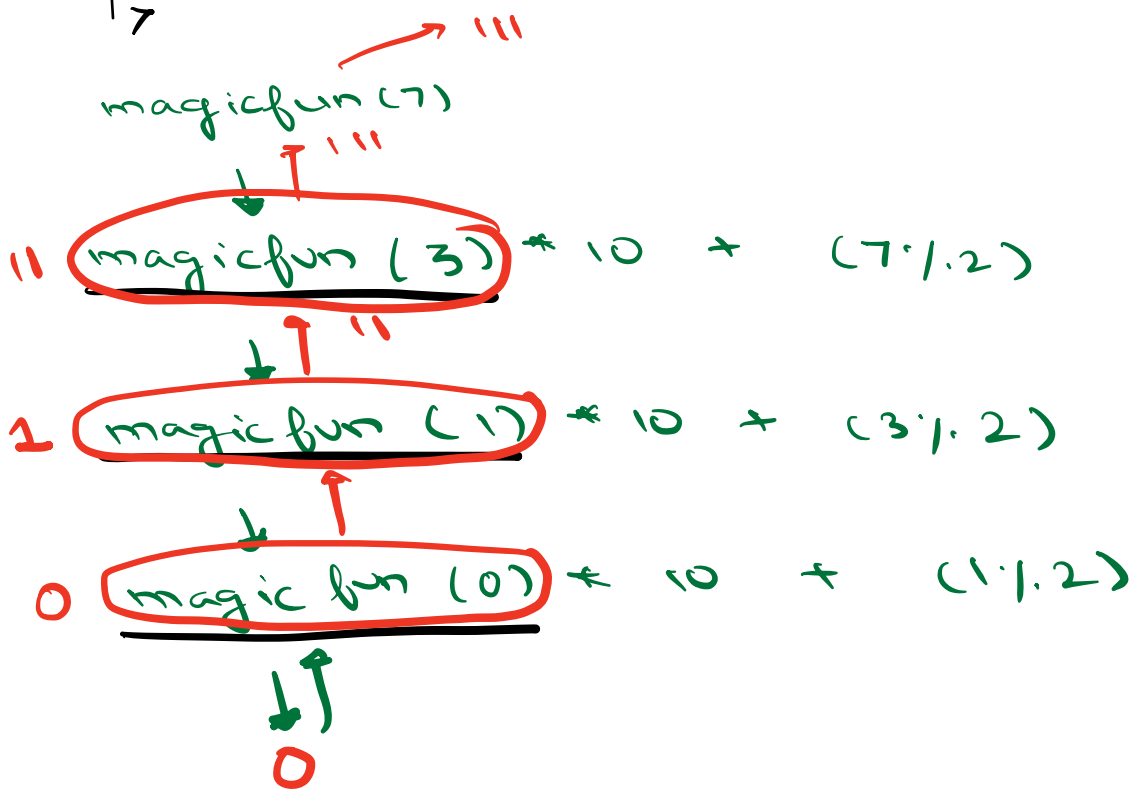
Print all subsets

Generate N digit numbers

Find K^{th} symbol

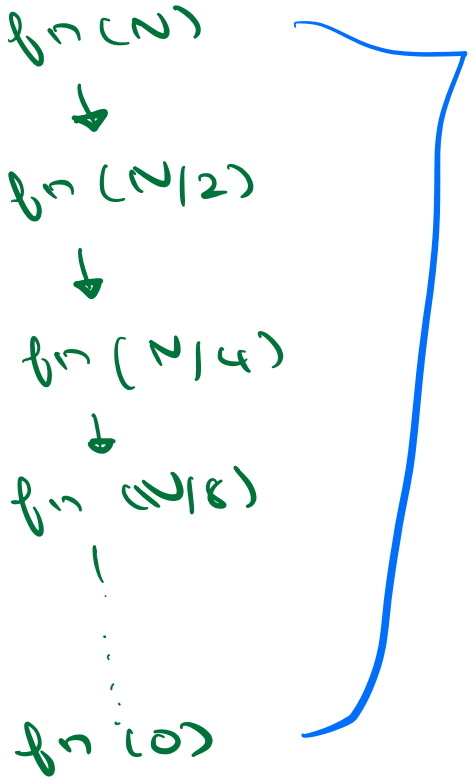
```
int magicfun(int N) {
    if (N == 0)
        return 0
    else
        return magicfun(N/2) * 10 + (N%2)
}
```

ans → 111



Rec : Q5 foo bar

$f(N) = f(N/2) * 10 + N\%2$



log N x O(1)

TC : O(log N)

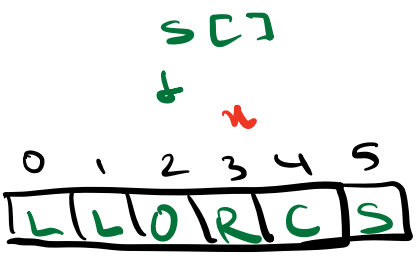
SC : O(log N)

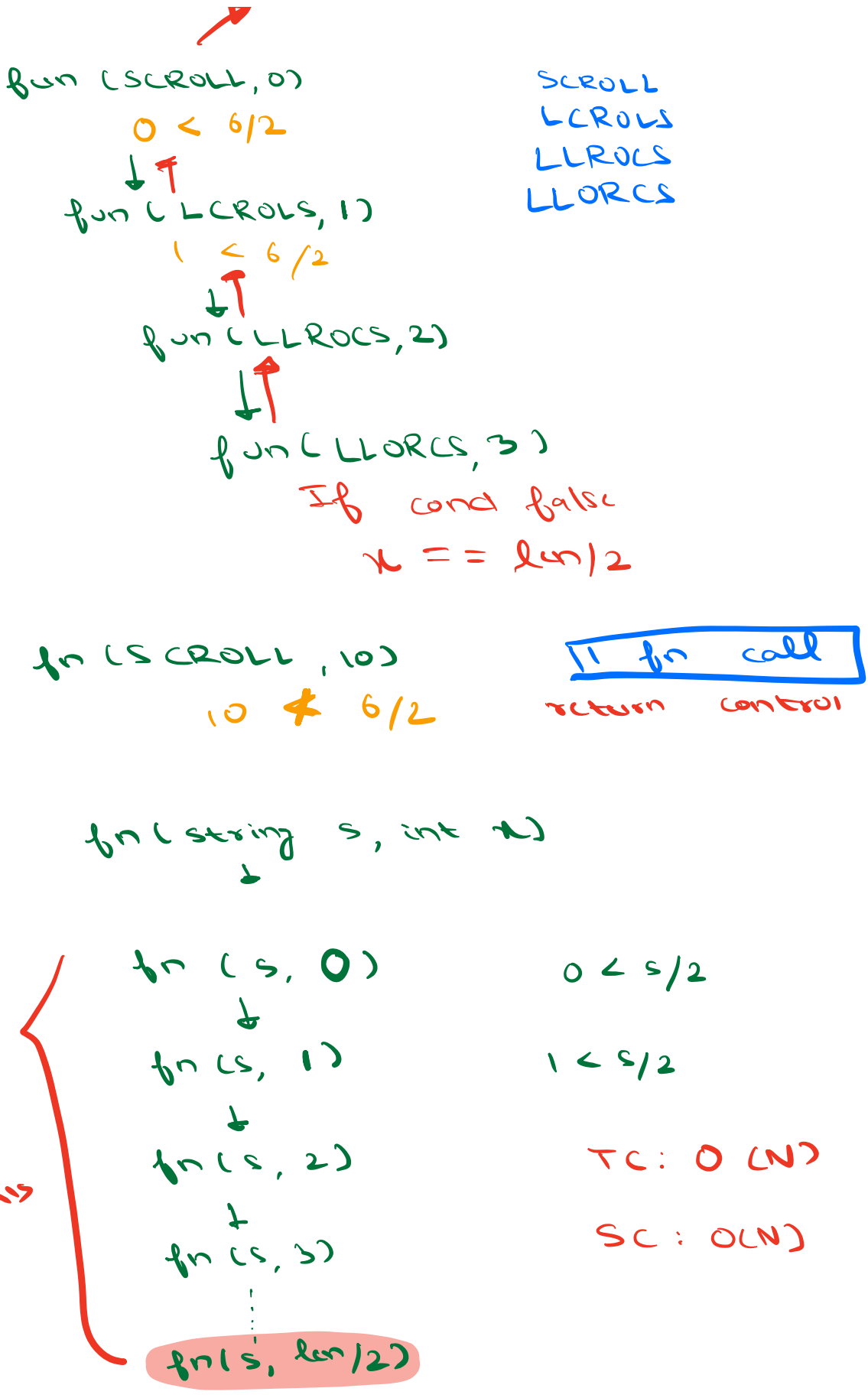
---

① Pass by reference

② Global

```
void fun (char s[], int x) {
    print (s)
    char temp
    if (x < s.length /2) {
        temp = s[x]
        s[x] = s[s.length - x - 1]
        s[s.length - x - 1] = temp
        fun (s, x+1)
    }
}
```




fun (SCROLL, 0)
0 < 6/2
fun (LCROLS, 1)
1 < 6/2
fun (LLROCS, 2)
fun (LLORCS, 3)
If cond false
x == len/2
SCROLL
LCROLS
LLROCS
LLORCS
fn (SCROLL, 10)
10 < 6/2
1 fn call
return control
fn (string s, int x)
fn (s, 0)
fn (s, 1)
fn (s, 2)
fn (s, 3)
fn(s, len/2)
0 < s/2
1 < s/2
N/2 fn calls
TC: O (N)
SC: O(N)

1. Given an array with distinct integers, print all subsets using recursion.

Input : [4, 6, 8] (0, 1, 2)

Subarrays

Continuous part

↓

Adjacent elements

(s, e)

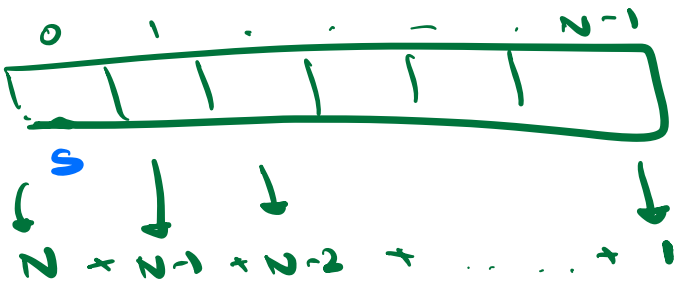
0, 0 1, 1

0, 1 1, 2

0, 2 2, 2

[4, 8] Not a subarray

No. of subarrays (Ar → N) $= \frac{N(N+1)}{2}$



Subset

Pick continuous / non continuous elements

All elements have a choice to be in subset or not

(No ordering in subset)

Can't use s-e to uniquely represent a subset

4, 6, 8 — len → 3

✓ ✗ ✓ ✗ ✓ ✗

			subsets
✗	✗	✗	⟨⟩
✓	✗	✗	⟨4⟩
✗	✓	✗	⟨6⟩
✗	✗	✓	⟨8⟩
✓	✓	✗	⟨4,6⟩
✓	✗	✓	⟨4,8⟩
✗	✓	✓	⟨6,8⟩
✓	✓	✓	⟨4,6,8⟩

$8 = 2^3$

len → N (2^N subsets)

$$\underbrace{2 \times 2 \times 2 \cdots \times 2}_{\text{N times}} = 2^N$$

x y — N = 2 — $2^2 = 4$ subsets

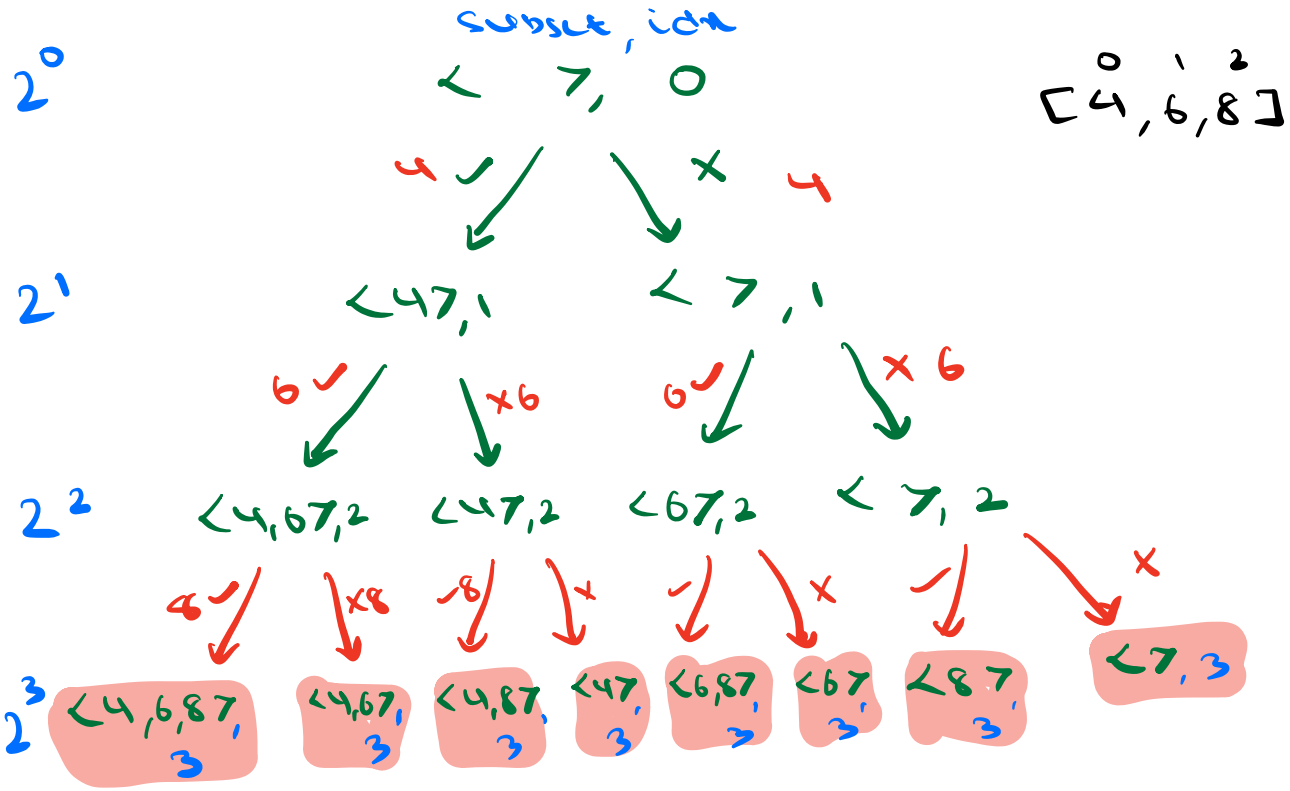
2 2

✓ ✓ → ⟨x y⟩

✗ ✗ ⟨x⟩

⟨y⟩

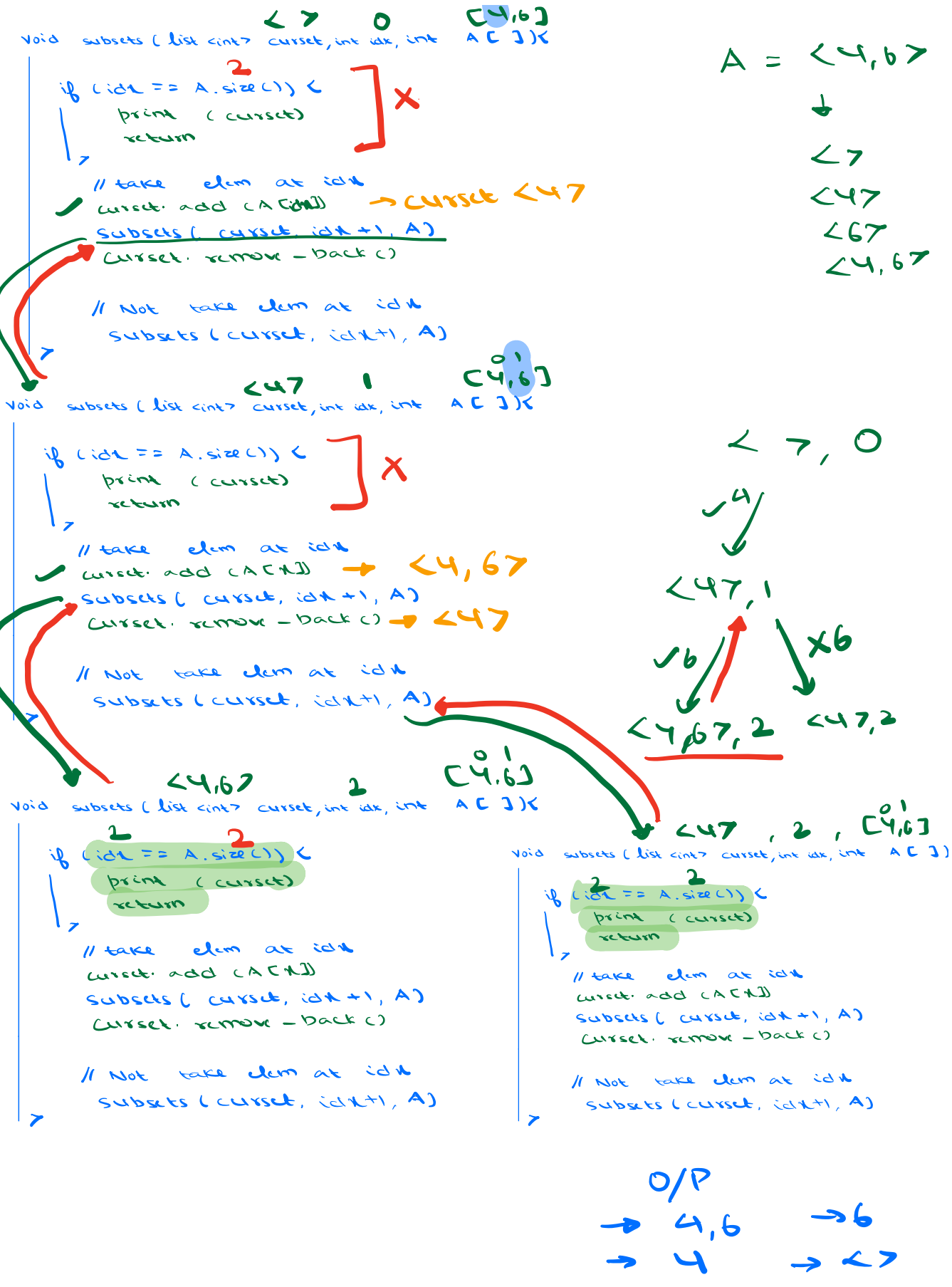
⟨⟩

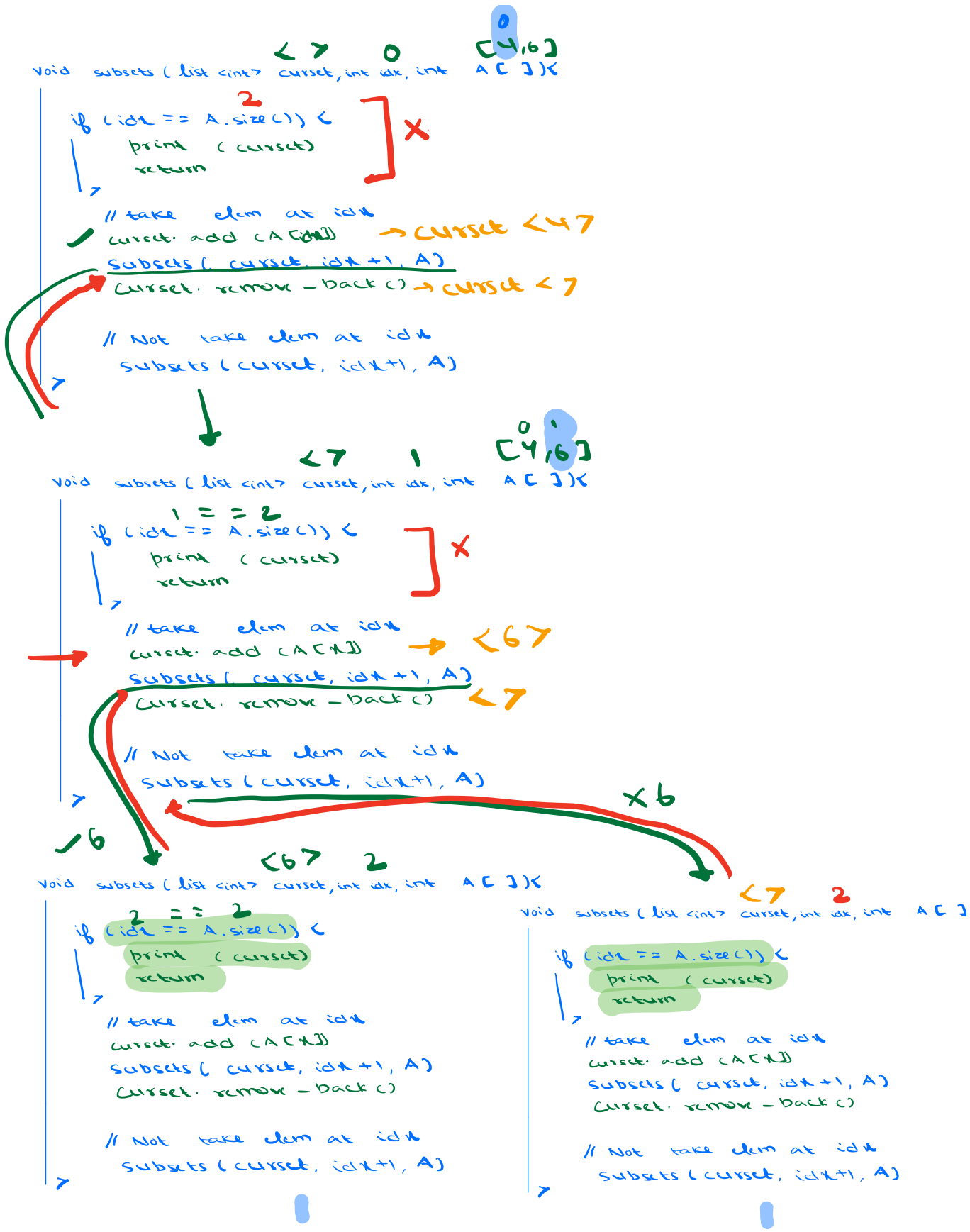


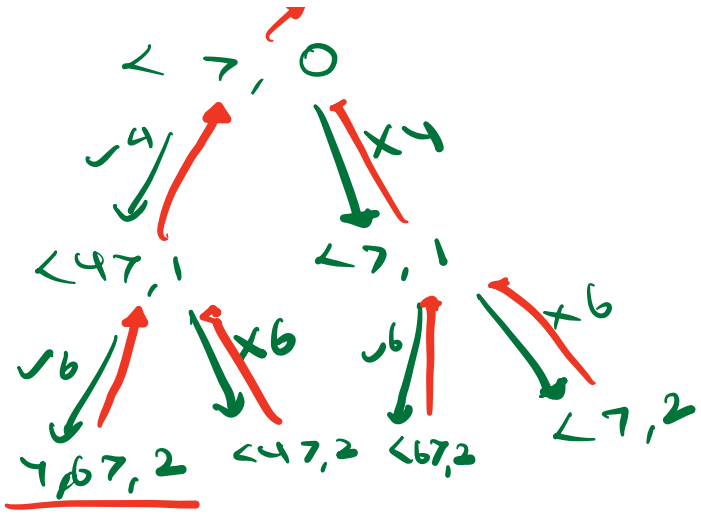
< 7 , 0

```
void subsets ( list <int> curset, int idx, int A[ ])<
    if (idx == A.size()) {
        print (curset)
        return
    }
    // take elem at idx
    curset. add (A[x])
    subsets ( curset, idx+1, A)
    curset. remove_back ()

    // Not take elem at idx
    subsets ( curset, idx+1, A)
}
```


A = <4,6>
<>
<7
<47
<67
<4,67
<47,1
<47,2
<4,67,2
O/P
4,6
6
4
<>


void subsets (list <int> curset, int idx, int A [ ])
< > 0
[4,6]
if (idx == A.size()) {
print (curset)
return
}
// take elem at idx
curset. add (A[idx])
subsets (curset, idx+1, A)
curset. remove – back ()
// Not take elem at idx
subsets (curset, idx+1, A)
→ curset <4,7>
→ curset <7>
<7> 1
1 == 2
→ <6>
<6> 2
2 == 2
<7> 2
x
x6
6



TC : $O(2^0 + 2^1 + 2^2 + \ldots + 2^n)$

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

TC : $O(2^N)$

SC : $O(N)$

10:58

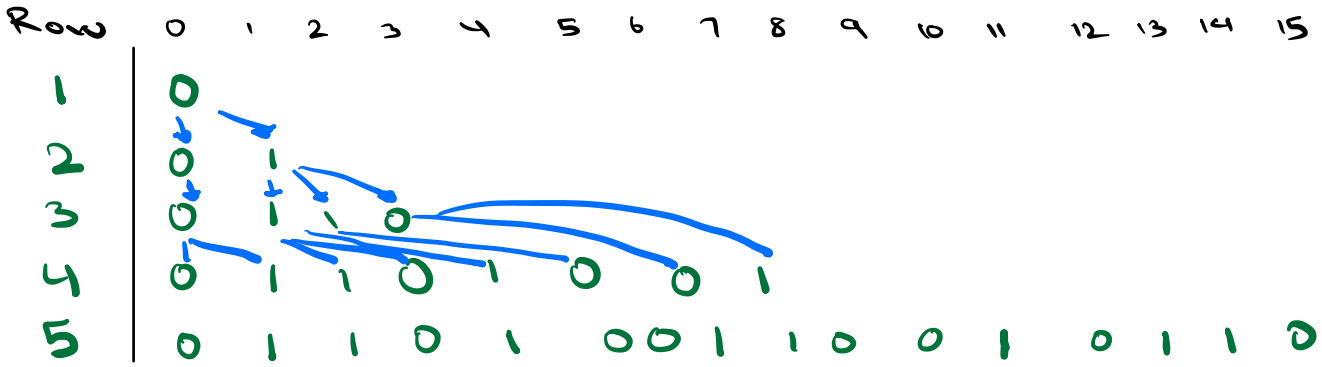
2. Find K^{th} character in a magical sequence of rows.

Seq starts with 1st row containing a 0.

Next row is generated from prev row

0 is replaced with 01

1 is replaced with 10



Find out K^{th} ele in N^{th} row

N, K	ans	
4, 5	0	
5, 11	1	
4, 9	Not	possible
5, 9	0	

BF: Generate all rows from 1st till N^{th} row

Maintain a cur row (start with 0),

Iterate in cur row and generate new row. ⇒ new row becomes cur row

1st → $1 = 2^0$ → 0

2nd → $2 = 2^1$ → 0 1

3rd → $4 = 2^2$

4th → $8 = 2^3$

2^{N-1} elements

$2^N - 1$ elements $\approx 2^N$ iterations

TC : $O(2^N)$

SC : $O(2^N)$ current

Constraints

$1 \le N \le 10^5$

$0 \le K \le 10^{18}$

For N value,

2^N will work

$2^N < 10^7 - 10^8$ iterations

$N = 20 - 25$

$2^{10} = 10^3$

$2^{20} = 10^3$

✓ $2^{20} = 10^6 < 10^7 - 10^8$

$2^{30} = 10^9$ iter X

$N = 30$

Optimized Approach : obs 1

Par [idx]

0

1

3

child [idx]

0, 1

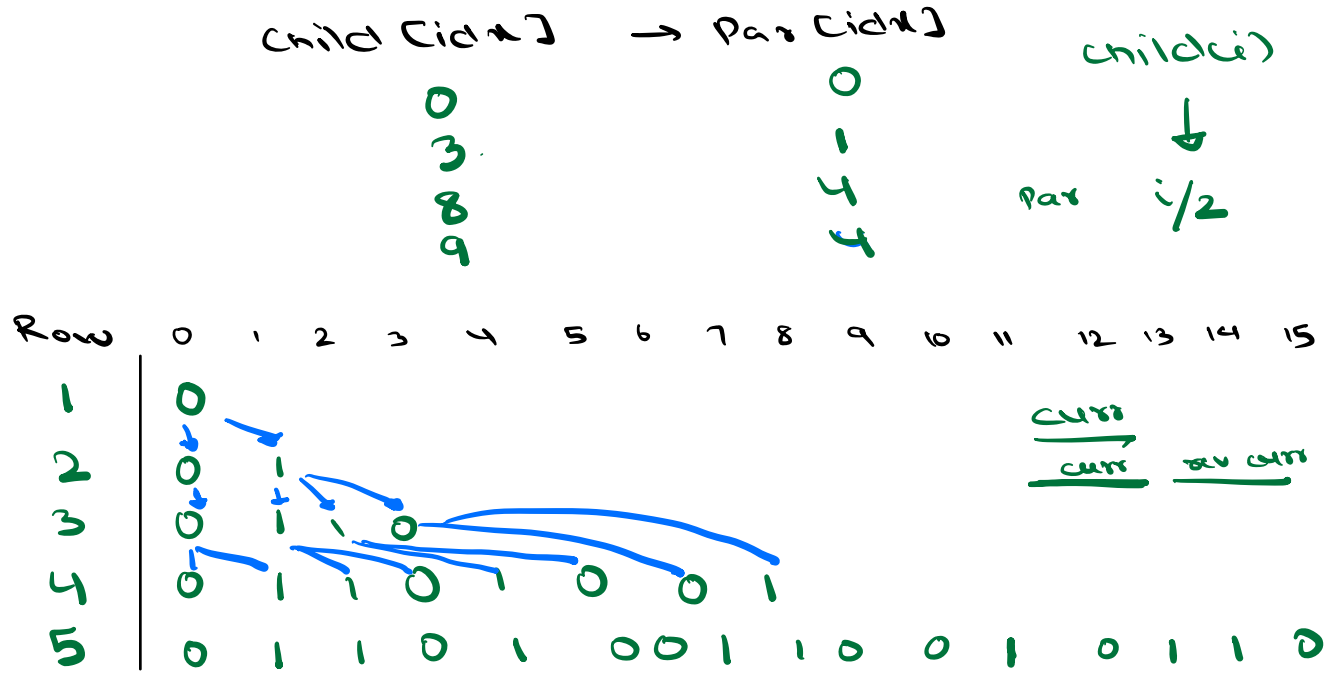
2, 3

6, 7

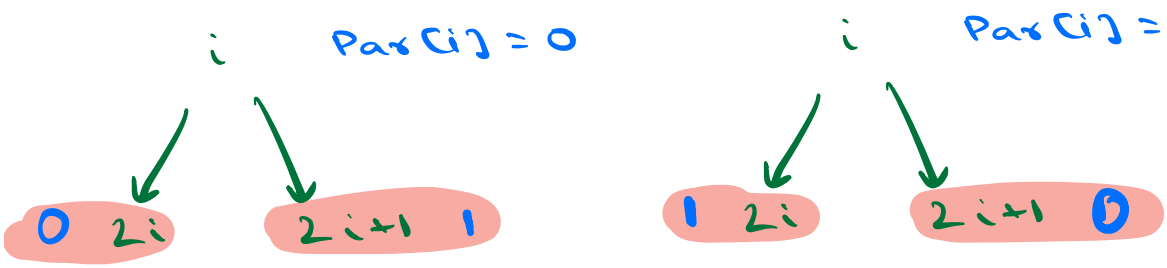
Par [i]

2i 2i+1

Obs 2



Obs 3



If idx of child is even, its data is same as par

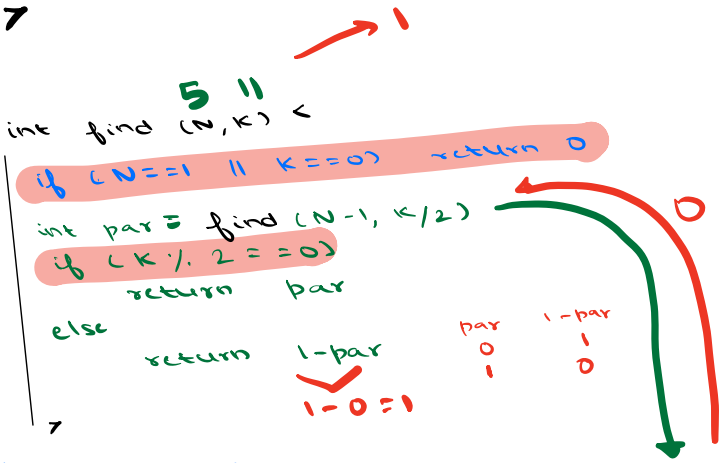
If idx of child is odd, its data is opp as par

N^{th} row, K^{th} idx

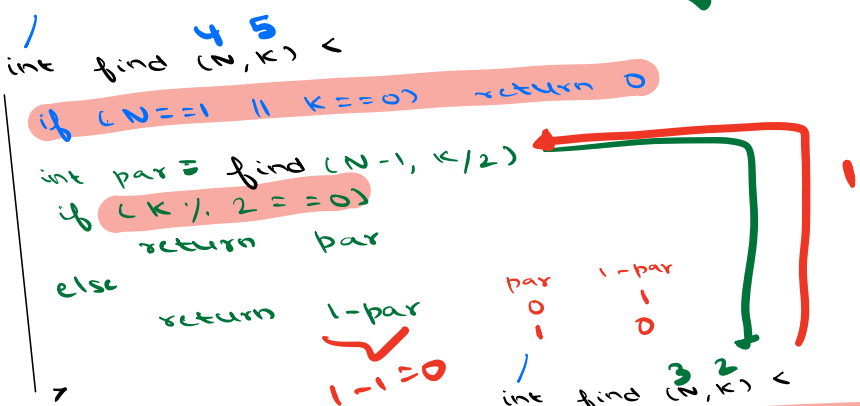
$N-1^{th}$ row, $K/2^{th}$ idx

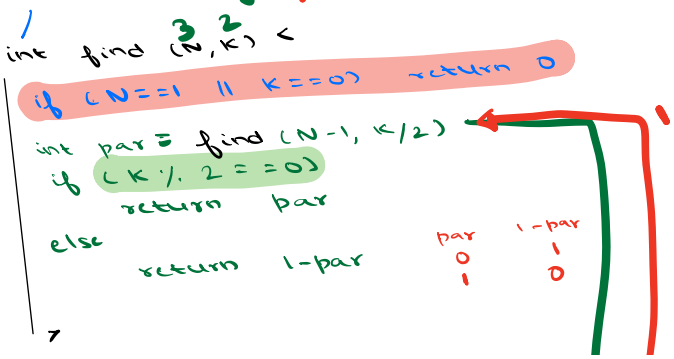
```
// return Kth char in Nth row
int find (N,K) {
    if (N==1 || K==0) return 0
    int par = find (N-1, K/2)
    if (K %. 2 == 0)
        return par
    else
        return 1-par
}
```

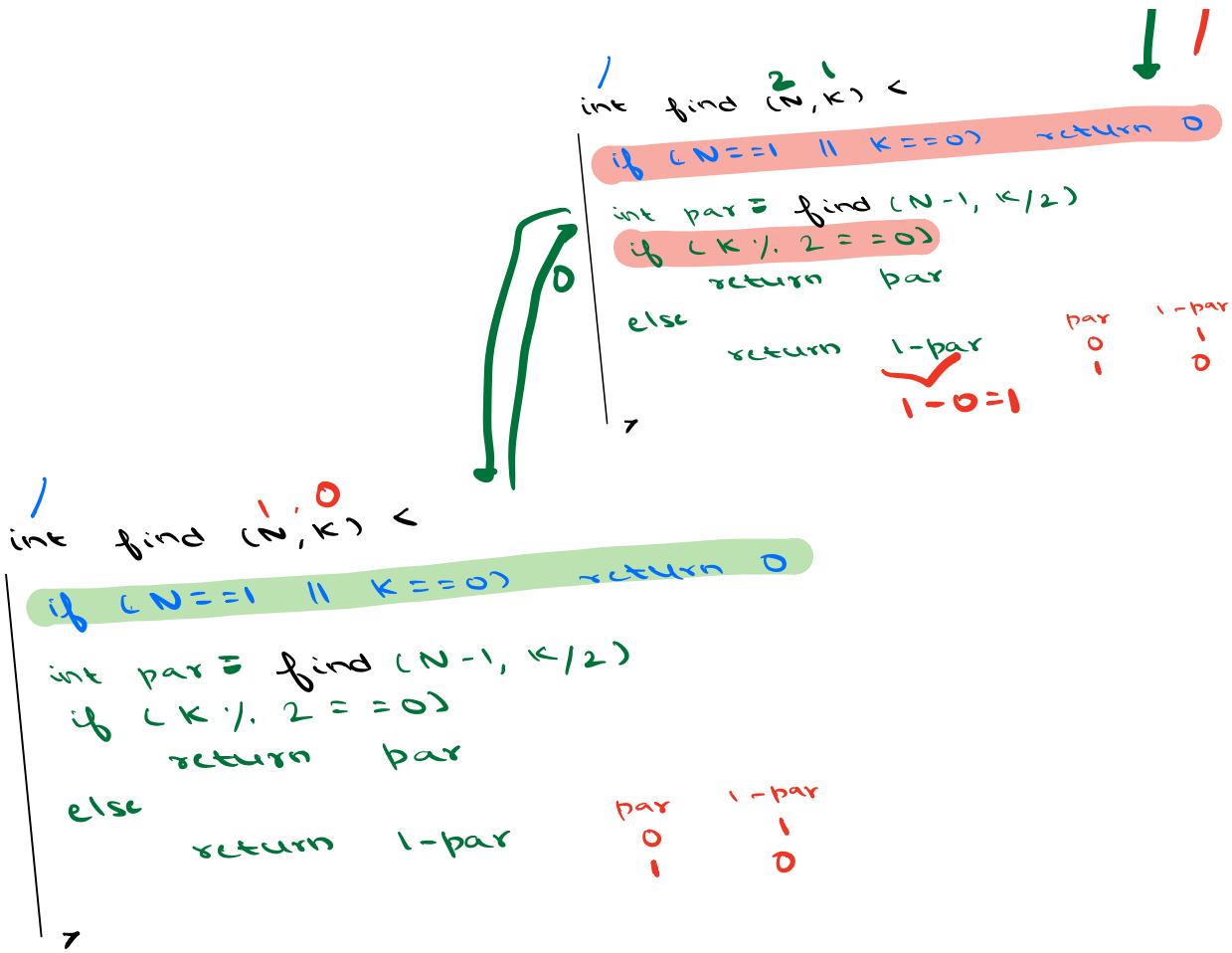
par	1-par
0	1
1	0



N	K	ans
5	11	1







N, K

↓

N-1, K/2

↓

N-2, K/4

↓

⋮

1, 0

Min (N, log K)

N → 1

N fn calls

K → K/2 → K/4 → ... 0

log K fn calls

TC: O(log₂ K)

K = 2^(n-1)

↓

O(n)

SC: O(log₂ K)

↓

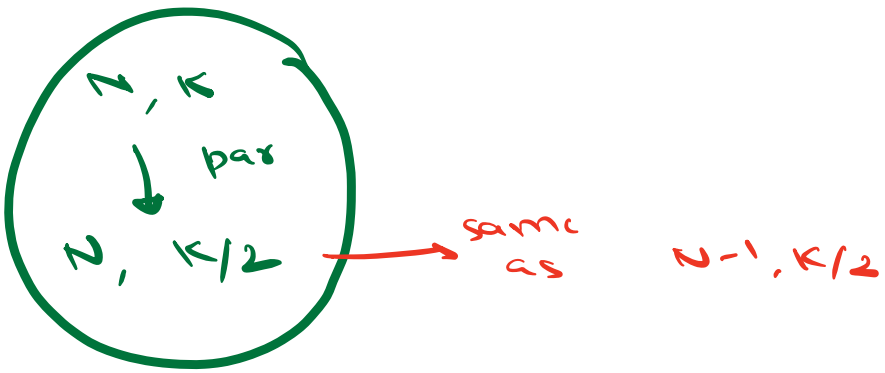
O(n)

K → idx in Nth row

$N^{th} \rightarrow 2^{N-1}$ symbols

$K = 2^{N-1}$

Obs




```
// return kth char in a row
int find (K) {
    if (K == 0)        return 0
    int par = find (K/2)
    if (K % 2 == 0)
        return par
    else
        return 1-par
}
```

$1 \le N \le 10^5$

$1 \le K \le 10^{18}$

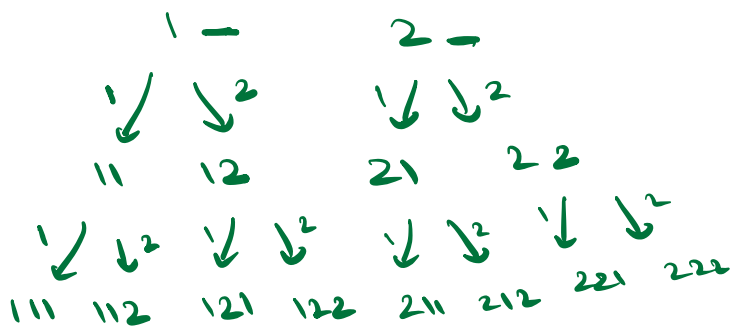
Given N, print all N digit nos. formed using 1 and 2 in increasing order

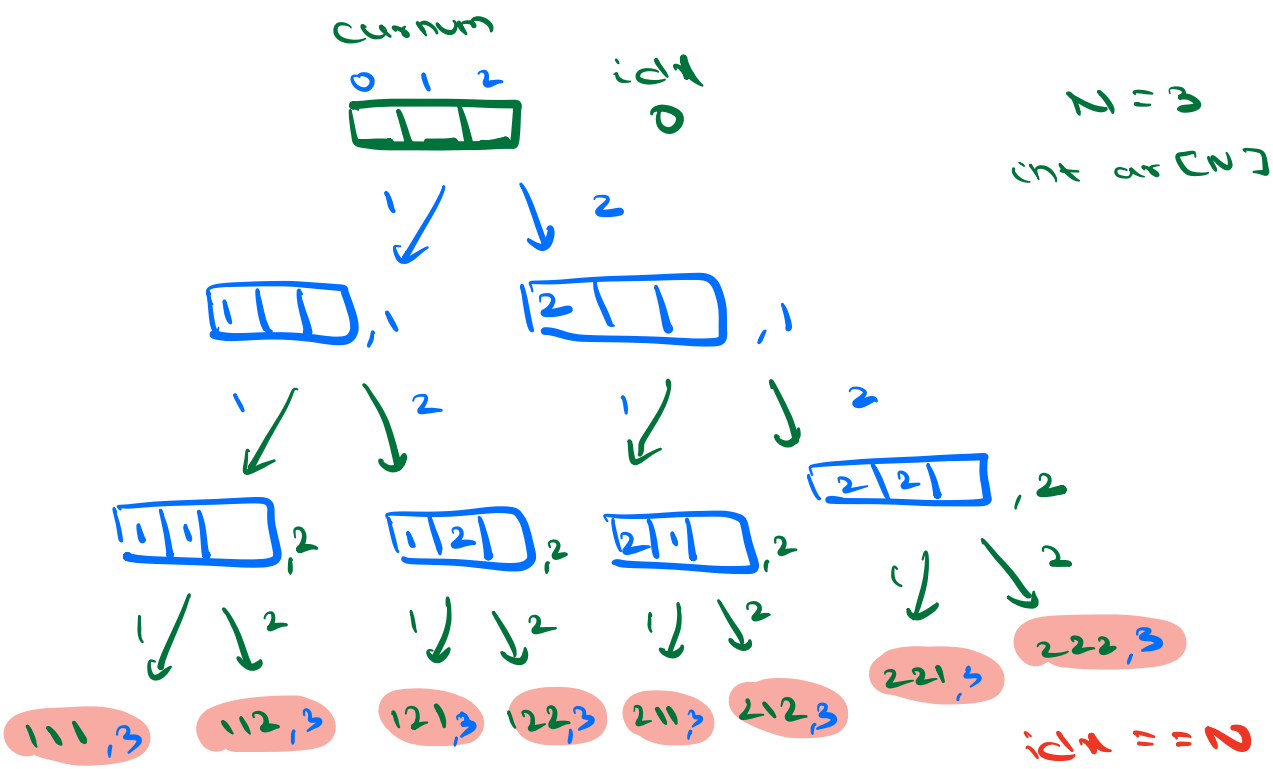
ans

N = 1 → 1, 2

N = 2 → 11, 12, 21, 22

N = 3 → 111, 112, 121, 122, 211, 212, 221, 222



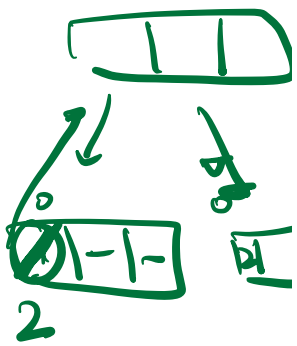


0 N

```
void print (int cur[], int idx, int N){
    if (idx == N){
        print (cur)
        return
    }

    // put 1
    cur[idx] = 1
    print (cur, idx+1, N)

    // put 2
    cur[idx] = 2
    print (cur, idx+1, N)
}
```



TC : $O(2^N)$

SC : $O(N)$

Bit Manipulation 2
HW Q4 Sum of XOR of all Pairs

3 0 1 1
4 1 0 0
① 1 1 1

4 1 0 0
2 0 1 0
⑥ 1 1 0

Contribution

3 0 1 1
2 0 1 0
① 0 0 1

2

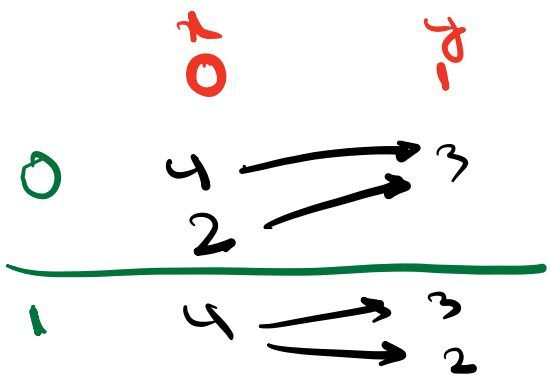
7+6+1 = 14

	2	1	0
3	0	1	1
4	1	0	0
2	0	1	0
XOR	1	1	1
pairs	2	2	2
contri	8	4	2

More and more set bits in XOR

	2	1	0
x	1	1	0
y	0	0	1
		1	

2^1

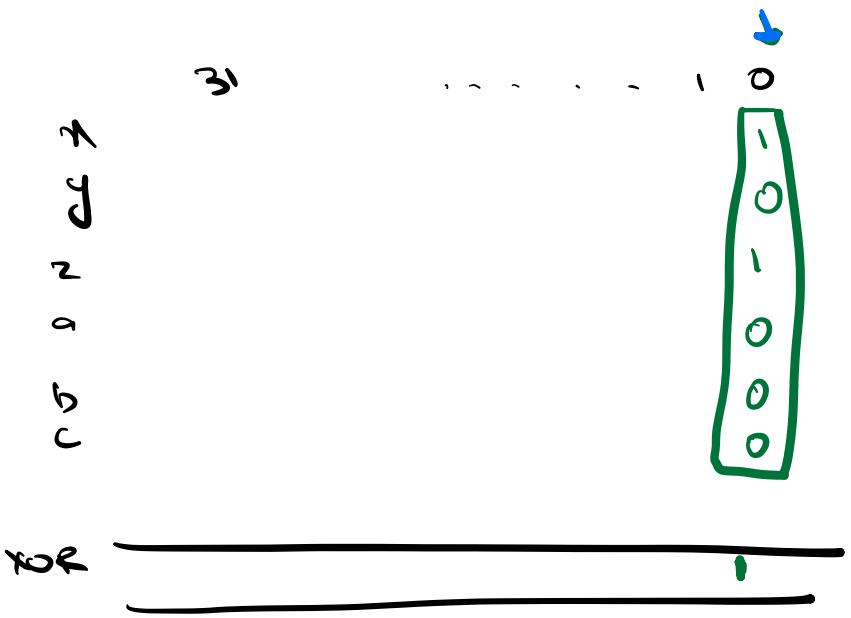


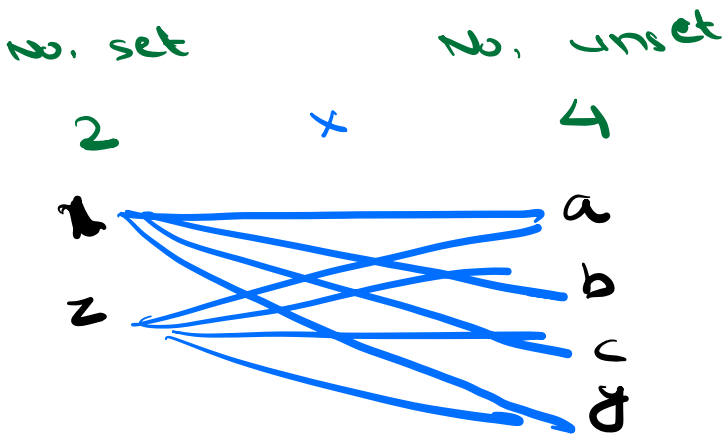
contri → $2^0 \times 2$
= 2

contri → $2^1 \times 2$
= 4

2

contri = $2^2 \times 2$
= 8





i^{th} bit → cnt no. which have
i^{th} bit set → cntset

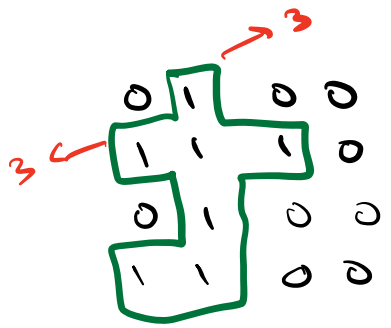
no. which i^{th} bit unset → cnt unset

Total pairs = cntset * cntunset

contri of ith bit = 2^i * total pairs

$i \rightarrow [0, 31]$

gtotal += contri of curr ith bit



$1 \leq N, M \leq 10^5$

NxM