Maths for DSA

Bitwise Operator.

(1.) AND Operator (&)

```
3 →   0 1 1
2 →   0 1 0
    &
-----------
      0 1 0
```

Table :-

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Notice :- if the number is "&" with all One (1's) then the same number is returned

```
7 →   1 1 1
    &
      1 1 1
-----------
      1 1 1
```
Same

```
10 →  1 0 1 0
    &
      1 1 1 1
-----------
      1 0 1 0
```
Same

② OR operator (|)

table :-

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

③ XOR operator (^)

- "if and only if"

table :-

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Imp points about XOR :-

① $a \wedge 1 = \bar{a}$

② $a \wedge 0 = a$

③ $a \wedge a = 0$

④ Complement operator (~)

Table :-

a	Y
0	1
1	0
a	$\bar{a}$

Application 1 :- Find if the given number is odd or even ?

Solution :- $(1010\textcircled{0})_{20}$ ← LSB

⇒ LSB tells you weather a number is even of odd

(Least Significant Bit)

How to find LSB ??

Ans → & with $(1)_2$ and it will give you 1 (odd) or 0 (even)

eg → (20) → 1010**0**

& 0000**1**

$(00000)_2 \rightarrow (0)_{10}$

even

(21) → 1010**1**

& 0000**1**

$(00001)_2 \rightarrow (1)_{10}$

odd

Ans → use AND operator

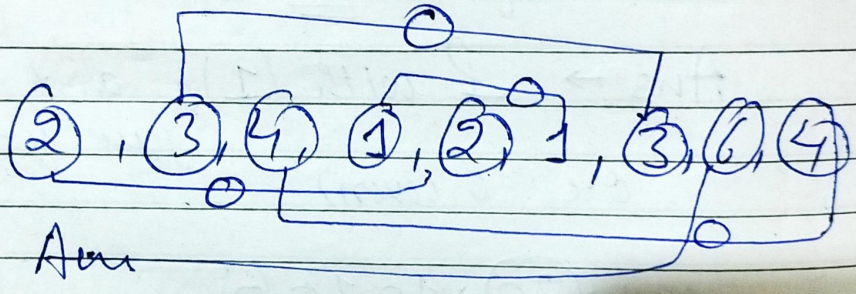
Application 2 :- ~~Remove~~ Find unique elements in the array of duplicates.

ars = { 2, 3, 4, 1, 2, 1, 3, 6, 4 }

Solution :-

(A) ⇒ We know that :- $a \wedge a = 0$ (XOR)

So,



ans = Ans

ans = 6

⇒ But How, we are not doing it at the same time like $2 \wedge 2$ or $3 \wedge 3$

we are doing it like $2 \wedge 3$

then ans $\wedge 4$

and so on.

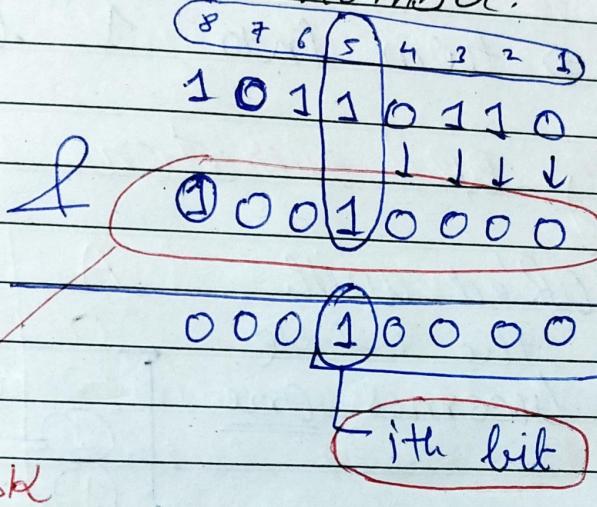
Eg → ars = { 2, 3, 4, 1, -2, -1, -3, 6, -4 }

if we were to add them:- regardless of order.

So ans would be (6).

Application 3:- Find i^{th} bit of a number.

Soln :-



mask

n → mask with n-1 zeros

→ This is a (1 << 4) (10000 — 5th)

So,

Ans :- n & (1 << (i-1))

Using mask and "AND" operator we can find the n^{th} bit.

Application - 4 :- Set and Reset a i^{th} bit.

(1) Set a i^{th} bit.

⇒ Set ⇒ turn into "1" bit

→ Using "OR" operator.

→ if OR'ed with 1 the whole bit becomes one

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

So,

eg → set 4th bit of 20

```
      4  3  2  1
1     0  1  0  0
OR(1) 0  1  0  0
-------------------
      1  1  1  0  0   (28)
```

So,

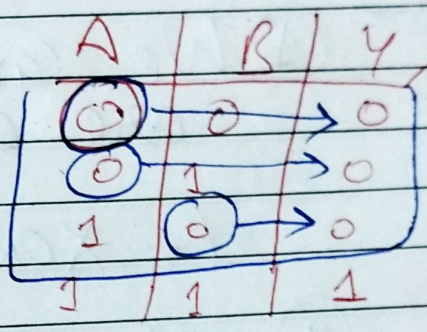
Ans ⇒ $n \mid (1 << (i-1))$

② Reset a bit

Reset a bit into "zero"

↳ Using "AND" Operator.

→ if AND with zero, entire bit becomes zero.



So, eg → Reset 4th bit to 0 of 28.

5	4	3	2	1
1	1	1	0	0
1	0	1	1	1

1 0 1 0 0

Ans) $n \& \sim (1 << (i-1))$

~ → Complement

Que.) find unique element in

arr = [7, 2, 1, 1, 2, 7, 2, 3, 1]

Soln

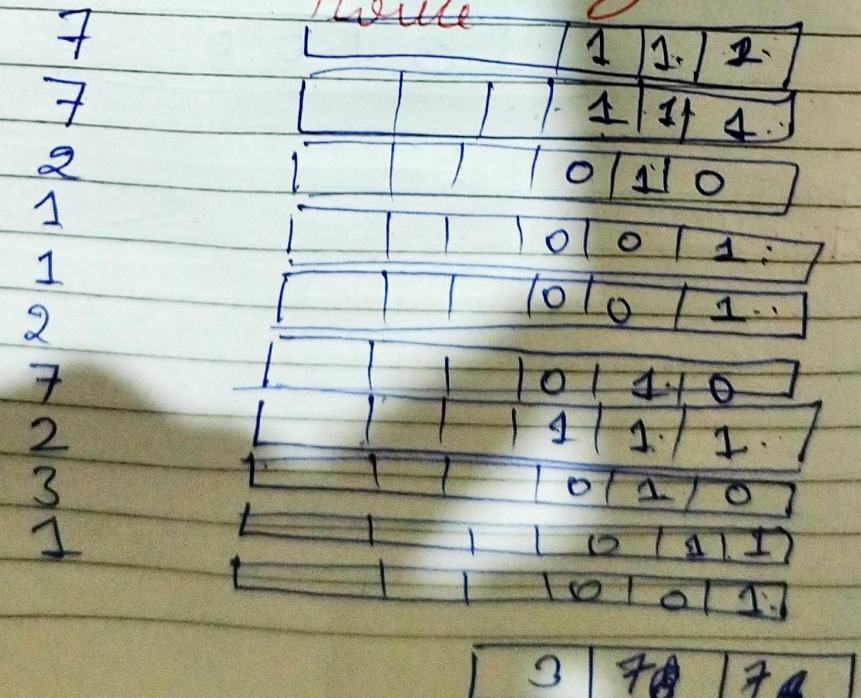
① We can see that all elements are appearing thrice

(∵ XOR doesn't work)

② Observe :-

① If we count (in binary) every elements (set) appear thrice

② There binary set appears thrice



③ That means every element set appears thrice

7 set = appears 3

2 set = appear 3

1 set = appear 3

3 set = will appear just once.

④ So, if we put sum of total bits and ideally if there are not unique elements then, bit sum

The total should be a multiple of 3

So,

bit sum % 3 == 0

And if not the unique element's binary will show itself

> In this case,

3	7	7

%3

0	1	1

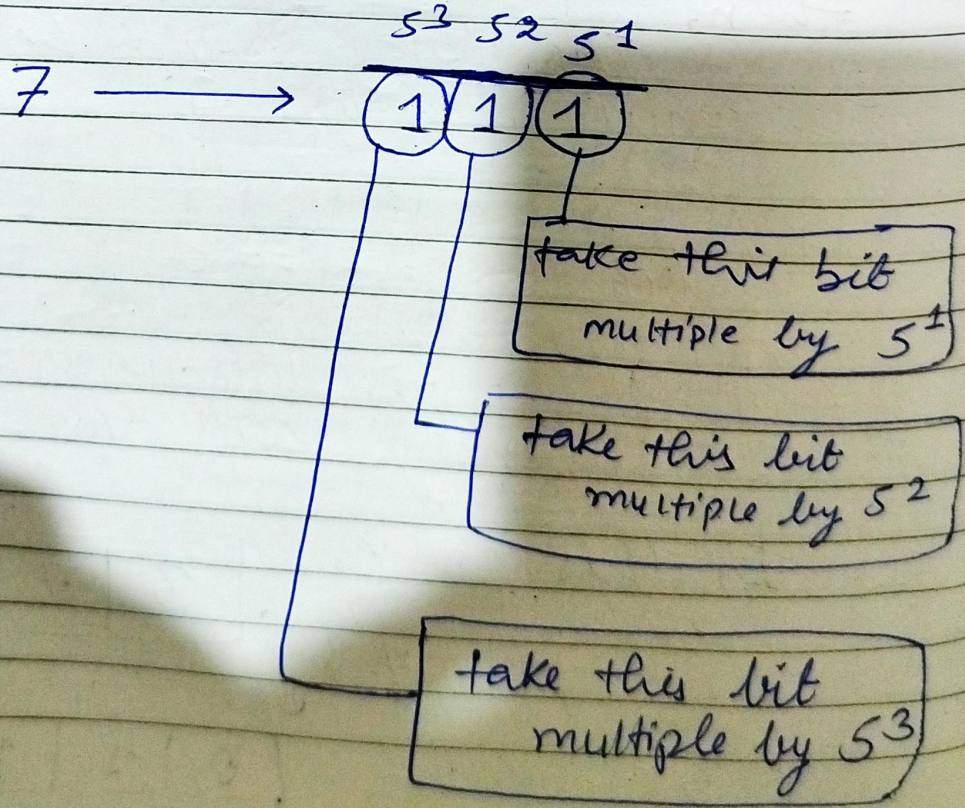
3

Ques) Find the n^{th} magic number :-

	5^3	5^2	5^1	
1 =	0	0	1	= 5
2 =	0	1	0	= 25
3 =	0	1	1	= 30
4 =	1	0	0	= 125
5 =	1	0	1	= 130
:				:

- Amazon Interview Questions.

Soln Let's say we need to find for 7



What you need to do -

(1) Keep track of the power of 5

(2) Shift the bit
↳ Right Shift.

(3) until the number becomes zero

∴ Since we know the The condition not the numbers of iterations

↓

Use while loop

Application 5 ⇒ :- Find No of bits in a number.

(I) method

(1) use a counter and right shift until the (method.) number becomes zero. increment the counter

(2) use the formula

Formula

$$\log_b a = x$$

$$a = b^x$$

then, $\log_2 6 = x$

$6 = 2^{(x)}$

no of digits in representation of this Base

So, $\log_2 10 = 3.32$

$10 = 2^{3.32}$

number of times 2 has to be multiplied to give 10

So,

$10 = 2^{3.32}$

(int) + 1

= No of digits

Formula

$$\text{No of digits} = \text{int}(\log_b n) + 1$$

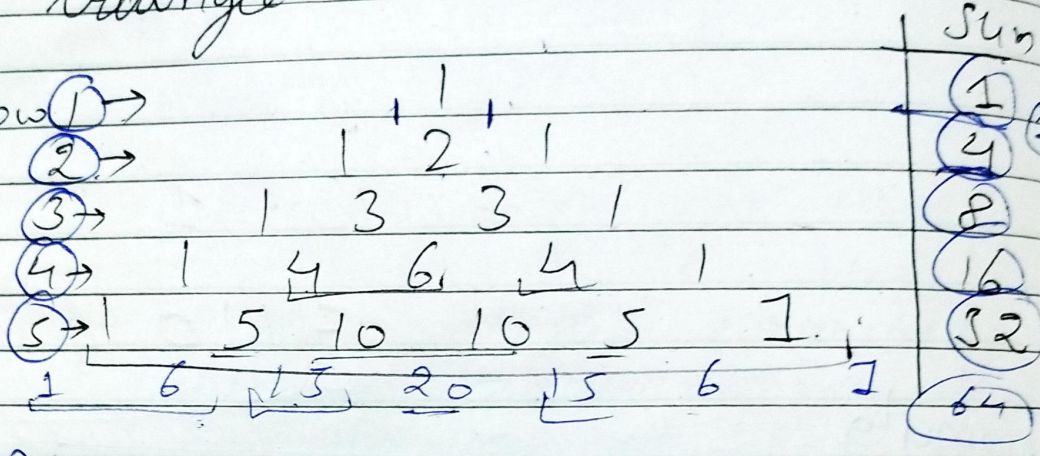
Now,

$$\log_b a = \frac{\log_x a}{\log_x b}$$

Complexity → log(n)

Que Find the sum of the n^{th} row of the pascal's triangle

Sol:



Pattern Observed :-

Sum of row(r) = 2^{r-1}

Sum of n^{th} row = 2^{n-1}

Eg :- Sum of 3rd row :-

Sum of 3rd row = $2^{3-1} = 2^2 = 4$

Ans = 4

Ans = $1 << (n-1) = 1 * 2^n$

a ^ 1 = a
a ^ 0 =

Q. You are given a number, & find if it a power of 2 or not ??

Soln Simple Solution :- AND (&)

eg → 10000 (16)
01111
10000 & 1
0

10000 + 1

if it is a power of two.

else, eg → 10100 (20)
01111
00100

↓
not zero
↓
Not the power of two.

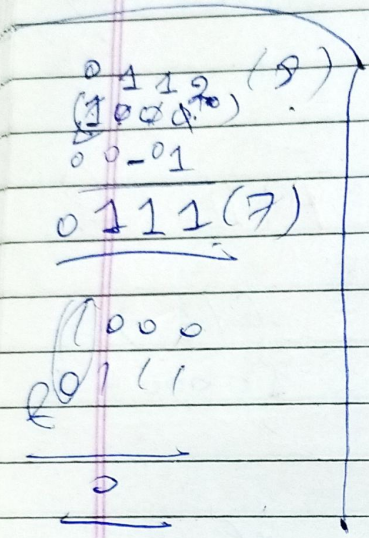
* This power of 2 → eg 10000
↓
can be written as
↓
11111 + 1

same

So, if

$$n \& (n-1) == 0$$

then it is a power of (2)



Que Calculate a^b.

blogger.

Sol^n (1) method ⇒ eg ⇒ 3^6 ⇒ 3 x 3 x 3 x 3 x 3 x 3

→ Complexity ⇒ O(6)

(2) method ⇒ $3^{(6)}$ → (110) → 3

3^{4+2}

$2^2 2^1 2^0$

110

⇒ 3^{110}

ans = 1 9 (Start)	n = 1 1 0 2 1
base = 3	n >> 1 ⇒ 0 (ignore)
base = 9	n = 11 >> 1 = 1
base = 81	n = 1 >> 1 = 1

② ans = ans * base

① Base = Base * Base

for ⓪ → (ignore) ans = 1, Base = 3

for ① → ans = 1 (ans) × 3 (Base)

ans = 3

So,

$3^{110} \longrightarrow 3^4 * 3^2 * 3^0$

Important to understand

Complexity → (log (b))

(#7) Integer to Binary String (n)
will convert and give you binary representation.

Ques. Given a number n, find the no of set bits in it

Soln. n = 9 — (1001)

Ans = 2.

(1) method :- use a counter and put every bit inverse increment for a set bit until the number becomes zero.

(1) method :-

n = 1001 = 6

n & (-n) = 0001

formula for right most set bit

and if we,

n - [n & (-n)]

= 1000

Eg. 9 → 1001
8 → & 1000
8 → 1000 → Count (1)

8 → 1000
7 → & 0111
0 → 0000 → Count (2)

Ans (2)

4000

Ques: Find the position of rightmost set bit.

→ 10110 1 1 00 (4, 2, 1)
a b

Ans = (4) position

Sol (2)

$N = a \mid b$

$-N = \bar{a} \mid b$

$N \& (-N)$

101101 & 000000 = 101101

Ans :- $\boxed{N \& (-N)}$

So, $N = 6 \Rightarrow 110$

$-6 \Rightarrow$.

Negative numbers in Binary.

1 byte = 8 bits

10 = 0 0 0 0 1 0 1 0

MSB ↓ can tell weather the number is +ve (0) or -ve (1)

LSB ↓ Tells if number is odd or even

-10

→ MSB is reserved bit.

Steps ①:- Take complement of number

Step ② Add One to it

10 → | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

① $\overline{10}$ → | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

② + 1

| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |

2's compliment method.

Easy way

10 → 00001010

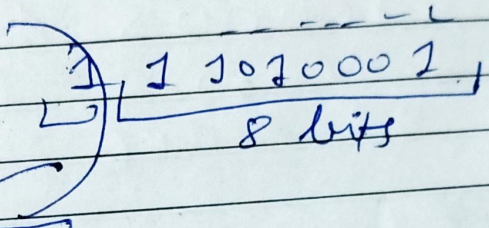
-10 → 11110110

① Copy as is it till you find a 1 bit

② After, every number should

be written as complement.

Now, Observe.

if 

will discarded.

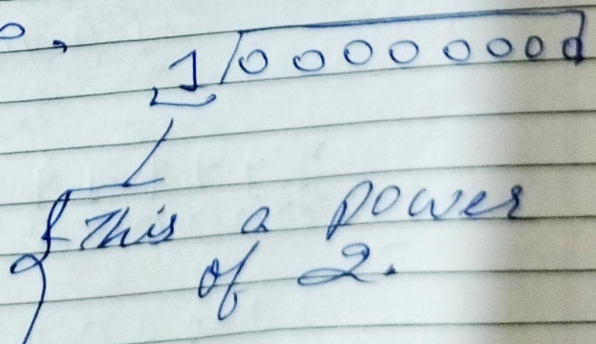
will be discarded anyway.

So, 1 00000000
- 11010001

will be -(11010001)

Right-

Now See,



→ 1000 = 111 + 1
(8) 7 + 1

100 → 11 + 1
(4) (3) + 1

10000 → 1111 + 1
(16) (15) + 1

So,

100000000 = 11111111 + 1

Now,

11111111 + 1 −
0000 1010

= 11111111 - 00001010 + 1

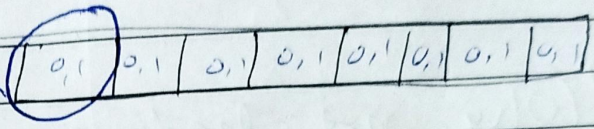
Step 1: Compliment of a number

Step 2

11111111
00001010
=
11110101 → $\bar{a}$

✱ Range of numbers.

1 byte → 8 bits



Total = 2^8 = 2 × 2 × ... 8 times

= 2^8 = 256

Sign of number

✱ actual number is stored in bits

= n - 1

In 1 byte = 7 bits

total = 2^7 = 128.

-128 to 128

└ 0 is there as well

So,

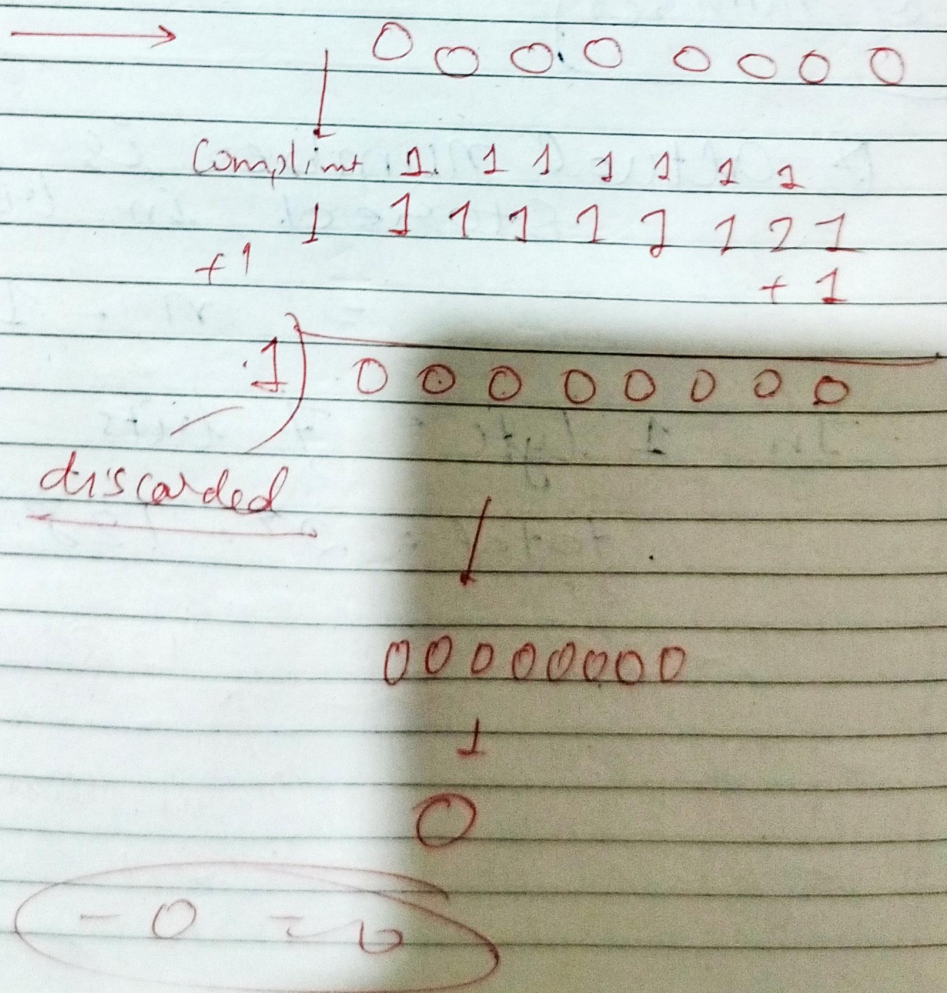
actual range becomes

-128 to 127

? Why not reduce from -ve side.

Because → 0 complement is 0

-ve of 0 is 0



So,

Range formula for n bits

$$-2^{n-1} \text{ to } 2^{n-1}-1$$

Ques Find XOR of numbers from 0 to a

Soln

a	XOR from 0 to a
0	0
1	0^1 = 1
2	0^1^2 = 3
3	0^1^2^3 = 0
4	0^4 = 4
5	4^5 = 1
6	1^6 = 7
7	7^7 = 0
8	8^0 = 8
9	8^9 = 1

1 0
0 1

1 0 0
1 0 1
0 0 1

1 1 0
0 0 1
1 1 1

0 - 0
1 - 1
4 - 4
8 - 8

Observe

0 → 0	1 → 1
4 → 4	5 → 1
8 → 8	9 → 1

2 → 3	3 → 0
6 → 7	7 → 0
10 → 11	11 → 0

* There are sets becoming of same number, ones, as n + 1, and last zero.

same number	0
1	1
number + 1	2
0	3

same number	4
1	5
number + 1	6
0	7

.
.
.

Repeats

After

① Every four pattern is changing.

So, if (a % 4 == 0)

→ same number.

if (a % 4 == 1)

→ 1

if (a % 4 == 2)

→ a + 1

if (a % 4 == 3)

→ 0

Ques Find XOR in Range 3 to 9

Ans till - 9

0 ^ 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6 ^ 7 ^ 8 ^ 9

Extra

(a-1) ⑨ ← Data Pass → ⑥

0 ^ 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6 ^ 7 ^ 8 ^ 9

$\Rightarrow \underbrace{(0 \wedge 1 \wedge 2)}_{a-1} \wedge \underbrace{(0 \wedge 1 \wedge 2 \wedge \ldots \wedge 9)}_{b}$

ans = 3 ^ 4 ^ 5 ^ 6 ^ ... ^ 9

3 ^ 1

011
001
010

Ans = 2

Leetcode - 832
fliping an image

① input

1	1	0
1	0	1
0	0	0

→

Output

1	0	0
0	1	0
1	1	1

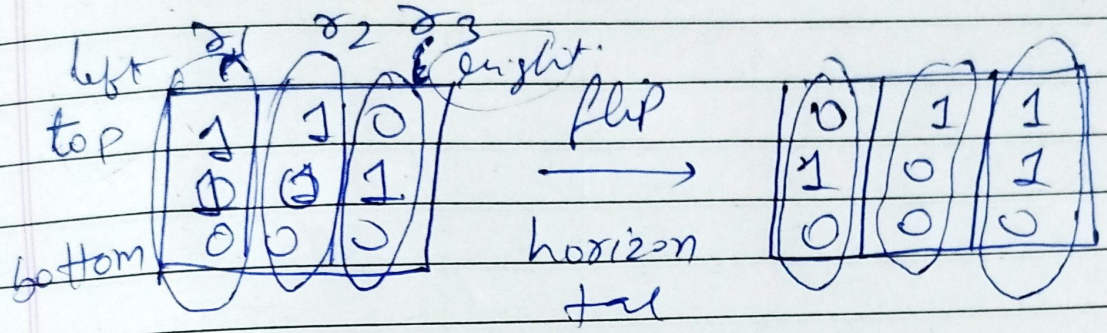
② input

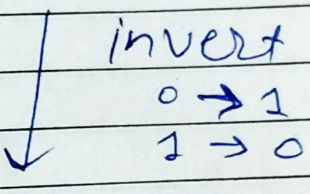
1	1	0	0
1	0	0	1
0	1	1	1
1	0	1	0

→

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

① first flip the imge then invert it.





return

1	0	0
0	1	0
1	1	1

a^b ⇒ Math.log(a) / Math.log(b)