11/02/22

DSML Intermediate - DSA

Problem Solving - 1

☆ Today's Content :

1 - 1.30 hr + Doubt.

☆. As posted on Slack Thread :

- 13 Assignment + HW Questions
- Bit Manipulation Recap

Adv. Trees } Mid of Adv.

{ 2 DP
  2 Graph

Digital Logic

NAND Universal

Negative

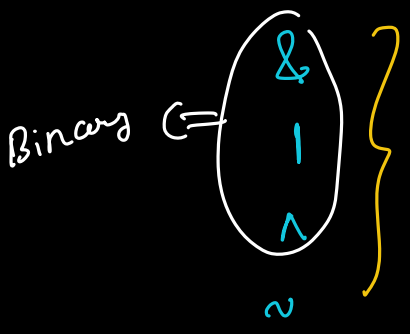
Bird View Recap on Bit Manipulation

0, 1.

1) Number System
   → Conversions.
   Decimal → Binary.

2) Add 2 Binary Numbers.

3) Bitwise operators

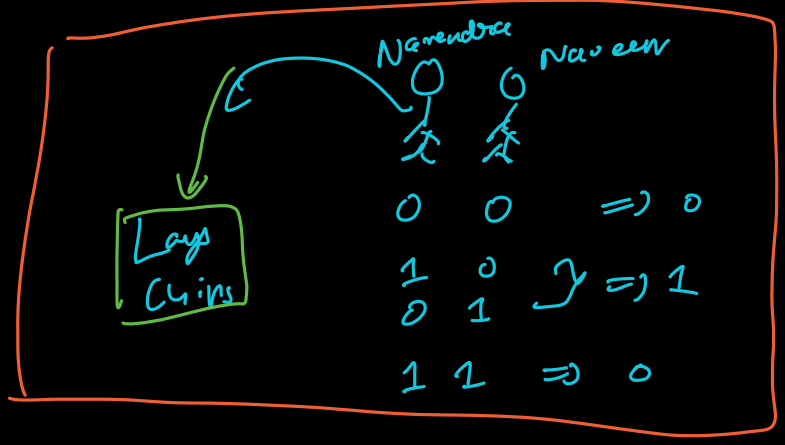


- RSA, AES-DES } Security.
- Image Compression.
- Hashing.
- Hamming distances.

Truth Table

bits.

a	b	a & b	a | b	a ^ b	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0



Q) $\boxed{\sim a}$ Decimal value of a.

3 bits

If ~ 3

$\downarrow$

011

100 => 4

4 bits

0011

1100 => 4+8=12

000101 => 5

000001 => 1

Negative

3 bits

X

1's complement

2's complement. ✓

sign bit

0 1

+ -

MSB

000	= +0
001	= +1
010	= +2
011	= +3
100	= -0
101	= -1
110	= -2
111	= -3

MSB

$2^2=4$ $2^1=2$ $2^0=1$

001 = 1

~

110 = -2

3 bits

2's c. = 1's + 1

= (1's c 1) + 1

001

1's complement 110

1's complement = $\sim a$

$\sim a = -(a+1)$

2's complement

$\sim a + 1 = -a$

$\Rightarrow \sim a = -a - 1$

MSB. a	=	$\sim a$
000	= +0	111
001	= +1	110
010	= +2	101
011	= +3	100
100		
101		
110		
111		

111
001
1000 (−a)

111	= −1
110	= −2
101	= −3

$\Rightarrow -a = (\sim a) + 1$

4 bits.

1 = 0001

−1 = 1's c + 1

1110
0001
1111

$\sim a = -a - 1$

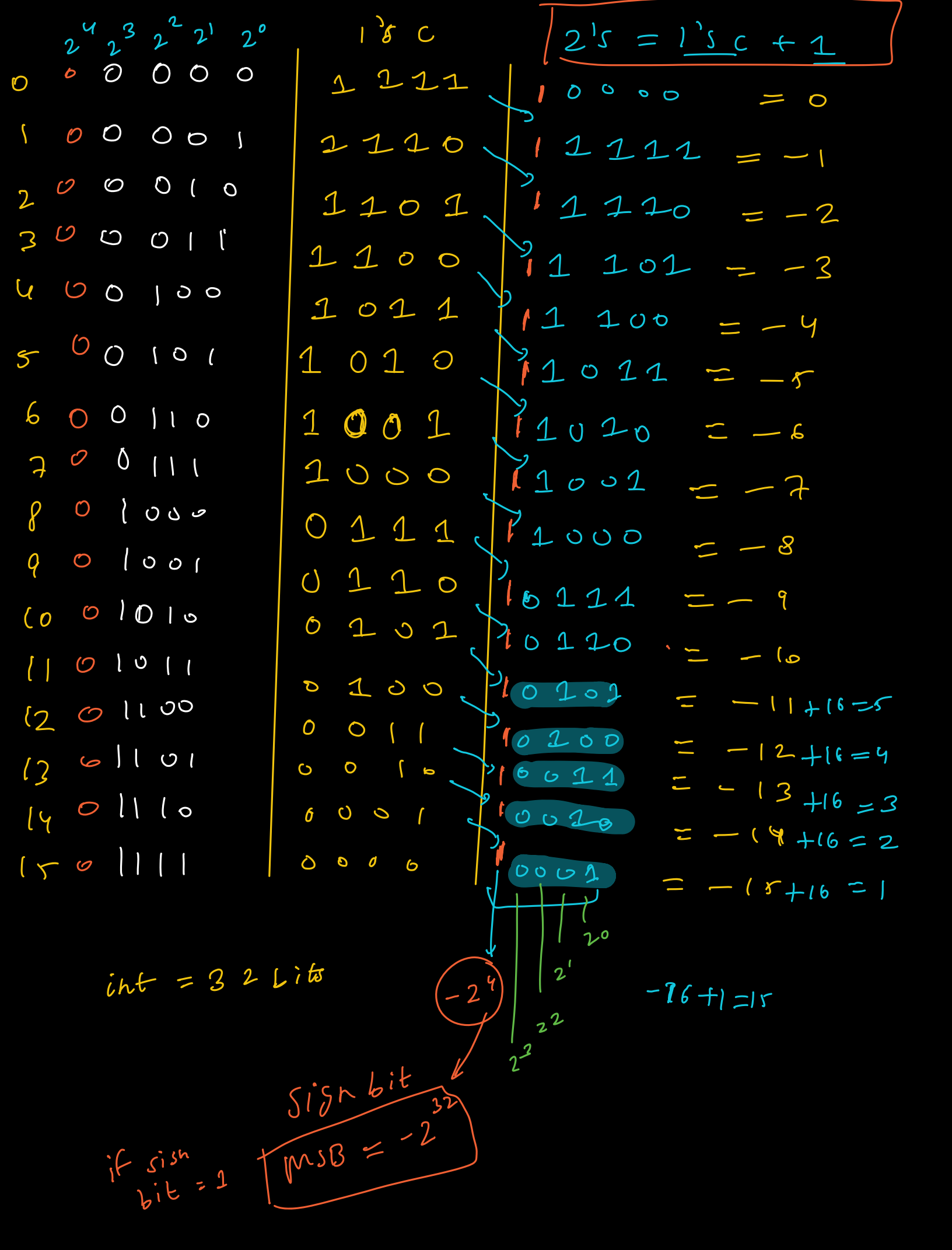
$\sim a = -(a+1)$

Negative of num.

$-num$ = 1's c + 1

= $(\sim num) + 1$

$\Rightarrow (\sim num) = -num - 1$


2^4 2^3 2^2 2^1 2^0
0 0 0 0 0
1 0 0 0 1
2 0 0 0 1 0
3 0 0 0 1 1
4 0 0 1 0 0
5 0 0 1 0 1
6 0 0 1 1 0
7 0 0 1 1 1
8 0 1 0 0 0
9 0 1 0 0 1
10 0 1 0 1 0
11 0 1 0 1 1
12 0 1 1 0 0
13 0 1 1 0 1
14 0 1 1 1 0
15 0 1 1 1 1
1's C
1 1 1 1
1 1 1 0
1 1 0 1
1 1 0 0
1 0 1 1
1 0 1 0
1 0 0 1
1 0 0 0
0 1 1 1
0 1 1 0
0 1 0 1
0 1 0 0
0 0 1 1
0 0 1 0
0 0 0 1
0 0 0 0
2's = 1's C + 1
0 0 0 0 = 0
1 1 1 1 = -1
1 1 1 0 = -2
1 1 0 1 = -3
1 1 0 0 = -4
1 0 1 1 = -5
1 0 1 0 = -6
1 0 0 1 = -7
1 0 0 0 = -8
0 1 1 1 = -9
0 1 1 0 = -10
0 1 0 1 = -11+16=5
0 1 0 0 = -12+16=4
0 0 1 1 = -13+16=3
0 0 1 0 = -14+16=2
0 0 0 1 = -15+16=1
2^0
2^1
2^2
2^3
-2^4
-16+1=15
int = 32 bits
sign bit
if sign bit = 1
MSB = -2^{32}

Properties of Operators

identity.

→ $a \wedge 0 = a$

→ $a \wedge a = 0$

same bits = 0

diff bits = 1

$0 \wedge 0 = 0$

$0 \wedge 1 = 1$

$1 \wedge 0 = 1$

$1 \wedge 1 = 0$

$1 \ 1 \ 0 \ 1 \ 1$

$1 \ 1 \ 0 \ 1 \ 1$

$0 \ 0 \ 0 \ 0 \ 0$

⇒ $n = [1, 0, 1, 0, 2, 2, 3]$

All numbers appear twice except one.

Xor commutative

$a \wedge b = b \wedge a$

associative.

$a \wedge (b \wedge c) = (a \wedge b) \wedge c$

$(1 \wedge 1) \wedge (0 \wedge 0) \wedge (2 \wedge 2) \wedge (3)$

$0 \wedge 0 \wedge 0 \wedge 3$

$= 3$

All numbers appearing even num times except 1 (odd).

10 = 8 + 2

14 = 8 + 4 + 2

16	8	4	2	1
0	1	0	1	0
0	1	1	1	0

|
&
^

a = 1 0 1 0

b = 1 1 1 0

a | b = 1 1 1 0

8 + 4 + 2 = 14

a = 1 0 1 0

b = 1 1 1 0

a & b = 1 0 1 0

8 + 2 = 10

a = 1 0 1 0

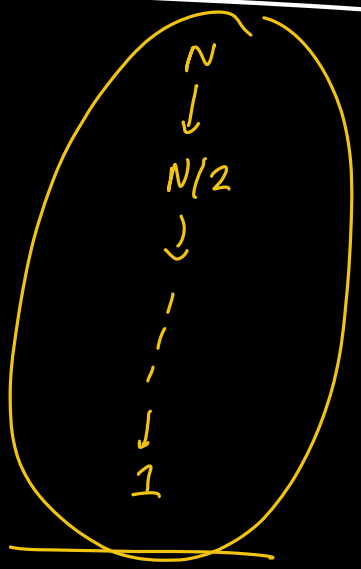
b = 1 1 1 0

a ^ b = 0 1 0 0

4

14 + 10 + 4 = 28

- Left Shift — multiply by 2
- Right Shift. — divide by 2



O(log N)

(1) << 5 = 1 * 2^5 = 32

(32) >> 5 = $\frac{32}{2^5}$ = 1

4 bits

0001 = 1

<< 1

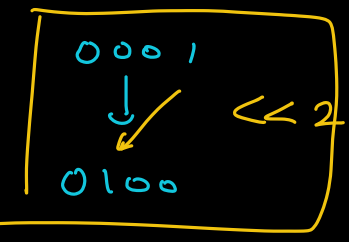
= 0010 = 2

1000 8

>> 1

0100 = 4

Bit Masking.



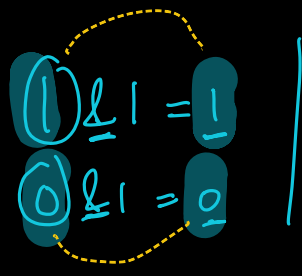
from LSB side.

① Check i^{th} bit is set or not in given number

$N = 14 = 8+4+2$

Check if 2nd bit.

```
    i
    ↓
  3 2 1 0
  1 1 1 0
& 0 1 0 0   ← bitmask
-----------
  0 1 0 0   > 0.
```




```
  1 1 0 1 1 0 1
& 0 0 0 0 0 0 1
---------------
            1
```

0th bit → set → odd

→ result even.

```
if a & 1:
    # odd
else:
    # even
```

```
def test_bit(number, idx):
    bitmask = (1 << idx)
    res = number & bitmask.
    return res > 0
```

TC. O(1)

- Set
- Unset
- Clear bit

Q) Check if a number has 1 set bit!

(32 bits)

10000100 => False

Brute → Check for all 32 bits & maintain a counter.

```
cnt = 0
for i in range(0, 32):
    if test_bit(num, i):
        cnt += 1
if cnt >= 2:
    return False
else:
    return True.
```

O(32)

O(total bits)

Observation

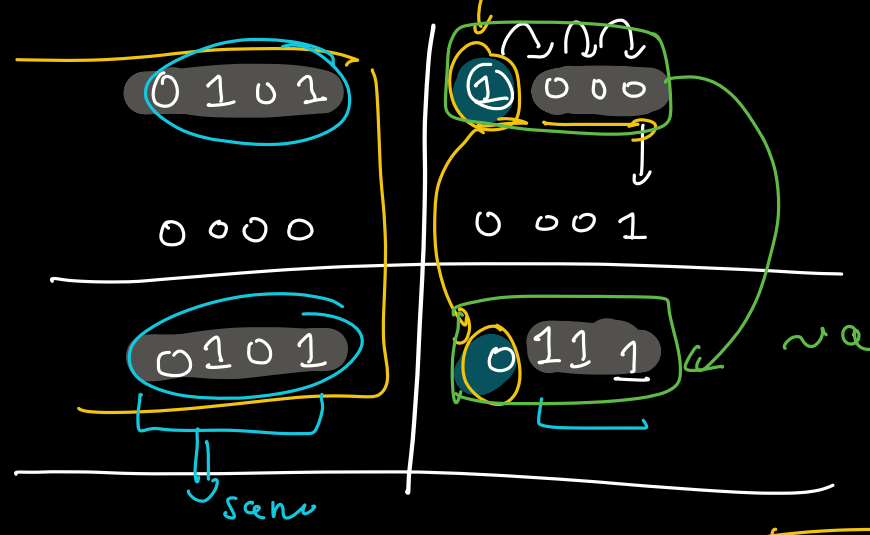
1 set bit
⇒ N is a power of 2

0 0 0 0 | 0 0 0 1

0 0 0 0 | 0 0 1 0

0 0 0 1 | 0 0 0 0

a & (a-1) == 2*(a-1)



a & a = a

a & (~a) = 0

N & (N-1):

0 1 0 1 | 0 0 0 0

LS set bit is cleared

N: 0 0 0 0 | 1 0 0 0

N-1: 0 0 0 0 | 0 1 1 1

N & (N-1): 0 0 0 0 | 0 0 0 0 ⇒ Only zero

N & (N-1)

0 → 1 set bit

>0 → >1 set bit

N: 00 00 0000

N-1: 11111 1111

(-1) &

0000 0 000

0 & anything = 0.

-1 = 1's complement of 1 + 1

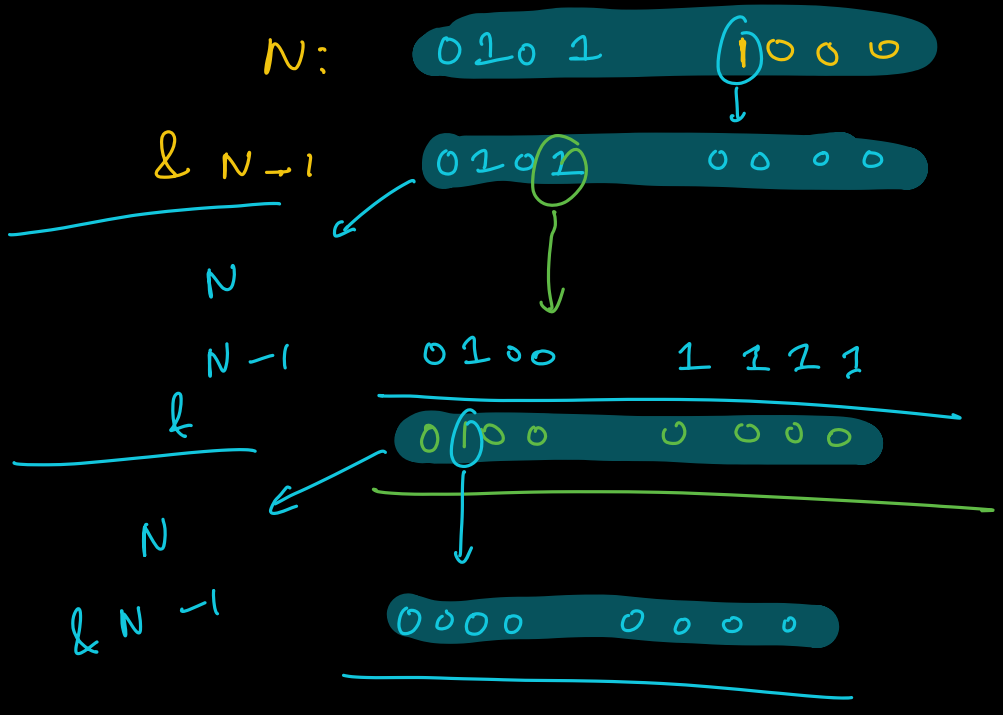
= 11111110

+ 00000001

11111111

```
def single_set_bit(num):
    if num == 0:
        return False
    return (num) & (num-1) == 0
```

Count Set Bits



TC:
O(number of set bits)

```
def cnt_set_bits (N):
    cnt = 0
    while (N > 0):
        cnt += 1
        N = N & (N-1)   # unset the rightmost set bit
```

Special Index

Count ways to make sum of odd and even indexed elements equal by removing an array element

Problem Description

Given an array, arr[] of size N, the task is to find the count of array indices such that removing an element from these indices makes the sum of even-indexed and odd-indexed array elements equal.

Problem Constraints

```
1<=n<=1e5
-1e5<=A[i]<=1e5
```

Input Format

First argument contains an array A of integers of size N

Output Format

Return the count of array indices such that removing an element from these indices makes the sum of even-indexed and odd-indexed array elements equal.

Example Input

Input 1:

```
A=[2, 1, 6, 4]
```

Input 2:

```
A=[1, 1, 1]
```

Spiral Order Matrix

Spiral Order Matrix II

Problem Description

Given an integer **A**, generate a square matrix filled with elements from 1 to A^2 in spiral order.

Problem Constraints

1 <= **A** <= 1000

Input Format

First and only argument is integer A

Output Format

Return a 2-D matrix which consists of the elements in spiral order.

Example Input

Input 1:

```
1
```

Input 2:

```
2
```

Check Subarray with sum = 0 using CF Techniques.

Longest Consecutive Sequence

Problem Description

Given an unsorted integer array **A** of size N.

Find the length of the longest set of consecutive elements from the array A.

Problem Constraints

$1 <= N <= 10^6$

$-10^6 <= A[i] <= 10^6$

Input Format

First argument is an integer array A of size N.

Output Format

Return an integer denoting the length of the longest set of consecutive elements from the array A.

Example Input

Input 1:

```
A = [100, 4, 200, 1, 3, 2]
```

Input 2:

```
A = [2, 1]
```

Q2.

Maximum Satisfaction

Problem Description

Given an array of integers **A** of size **N** denoting the fruits quality. **A[i]** denotes the fruit quality of the i^{th} fruit.

Shivam needs to pick **4** fruits but he needs to pick them in such a way that his satisfaction value will be maximum.

If **a, b, c** and **d** are fruit quality of the 4 fruits picked then the **satisfaction value(a, b, c, d) = (a & b & c & d)** where **&** is bitwise **AND** operator.

Find the maximum satisfaction value Shivam can obtain.

Problem Constraints

$4 <= N <= 10^5$

$1 <= A[i] <= 2 * 10^9$

Constraints

int => 32 bits

Input Format

The only argument given is the integer array A.

Output Format

Return the maximum satisfaction value Shivam can obtain.

Example Input

Input 1:

```
A = [10, 20, 15, 4, 14]
```

A = [10, 20, 15, 4, 14]

4 a & b & c & d is maximum.

Brute Force → 4 nested loops TC: $O(N^4)$.

4 bits

a: 1010

b: 1101

c: 1010

d: 1101

& 1000

$X_3 X_2 X_1 X_0$

$X_0 = 1$?

$X_1 = 1$?

$X_2 = 1$?

$X_3 = 1$?

maximum value.

MSB.

4 3 2 1 0

1) 10000 $= 2^4 = 16$

2) 11000 $= 2^4 + 2^3 = 16 + 8 = 24$

3) 00010 $= 2^1 = 2 = 2$

4) 01111 $= 2^3 + 2^2 + 2^1 + 2^0 = 8 + 4 + 2 + 1 = 15$

5) 10001 $= 2^4 + 2^0 = 16 + 1 = 17$

Ignoring sign bit

32 bit

```
ans = 0
mask = 0
for i = 31 to 0:        # MSB
```

31 to 0.

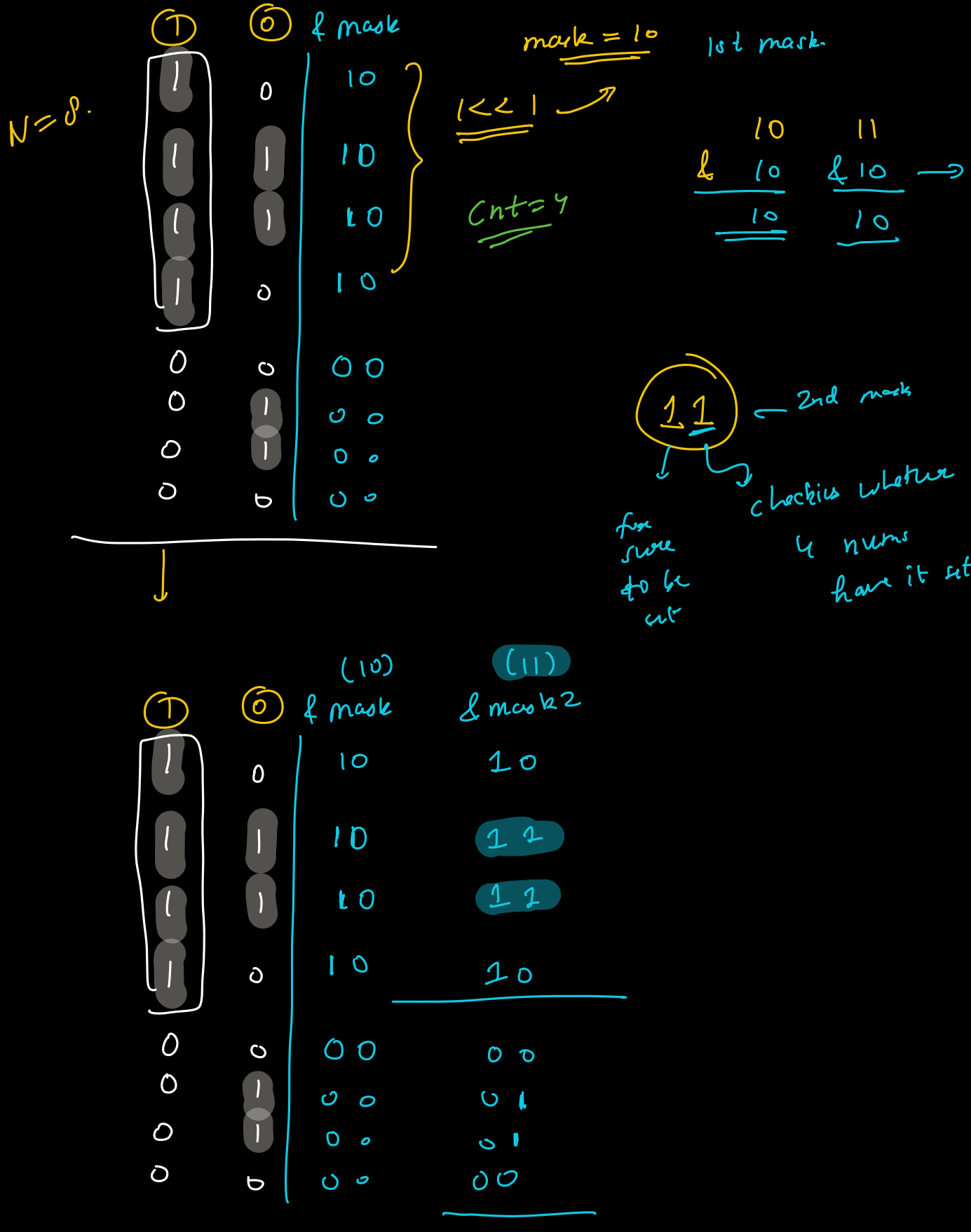
Q) Can we set ith bit as set bit here

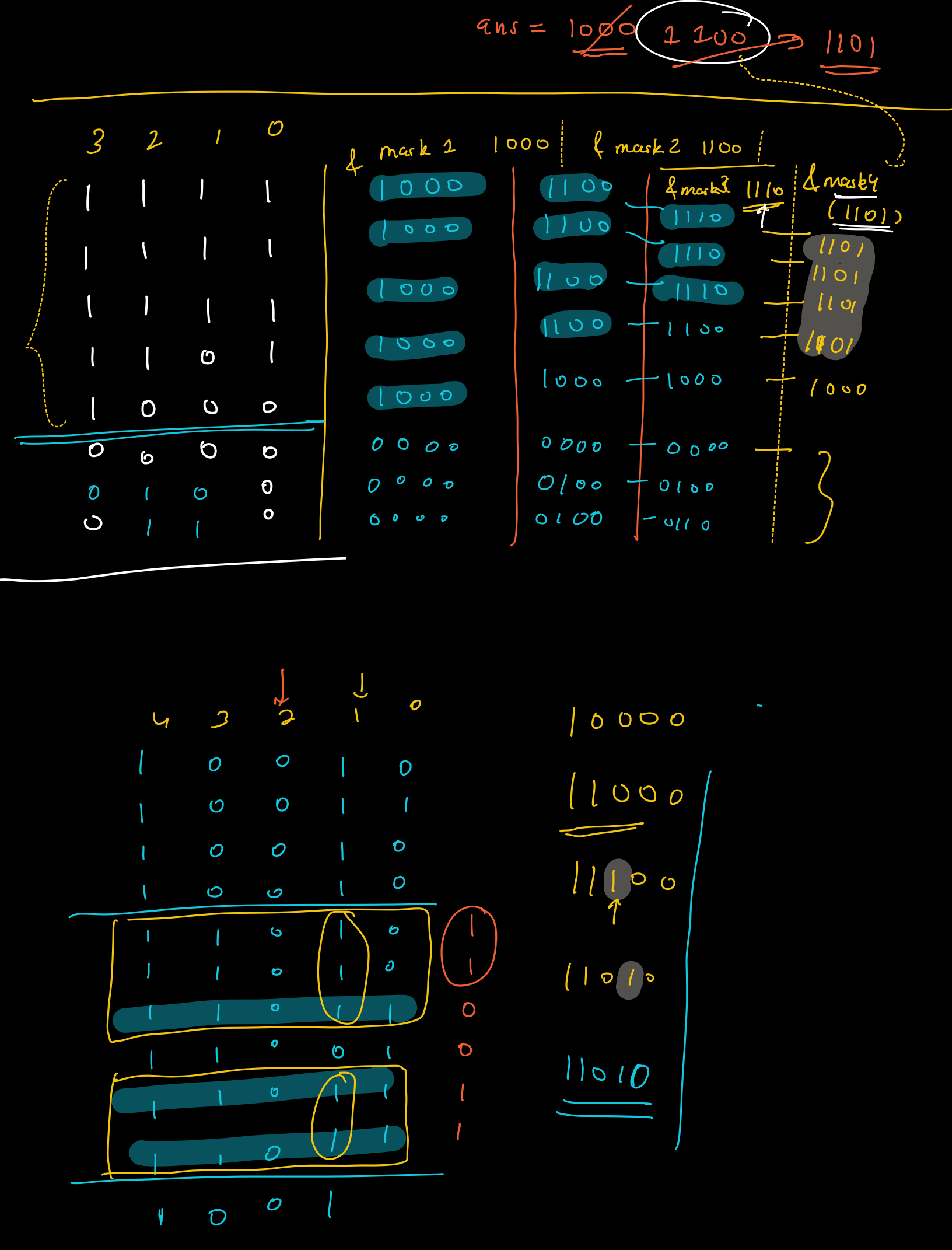
a & b & c & d

↳ atleast 4 numbers having a set bit here.

Take & with mask for all nums.

```
mask = ans | (1<<i)
if check(A, mask) >= 4:
    # update ans.
    ans = ans | (1 << i)      # set Bit
```

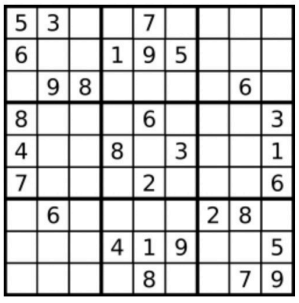

N=8.
1
0
& mask
10
1 0
1 0
1 0
0 0
0 1
0 0
0 0
mask = 10
1<<1
Cnt=4
1st mask.
10 & 10 → 10
11 & 10 → 10
11
2nd mask
checking whether
4 nums
have it set/not
for sure
to be
set
(10)
(11)
& mask
& mask2
1 0
1 0
1 0
1 0
1 1
1 1
0 0
0 1
0 0
0 0
0 1
0 1
0 0


ans = 1000
1100
1101
3 2 1 0
& mask 1 1000
& mask 2 1100
& mask3 1110
& mask4 (1101)
4 3 2 1 0
10000
11000
11100
11010

Valid Sudoku

Determine if a Sudoku is valid, according to: http://sudoku.com.au/TheRules.aspx

The Sudoku board could be partially filled, where empty cells are filled with the character '.'.



The input corresponding to the above configuration :

```
["53..7....", "6..195...", ".98....6.", "8...6...3", "4..8.3..1", "7...2...6", ".6....28.", "...419..5", "....8..79"]
```

A partially filled sudoku which is valid.

> ***Note:***
> - A valid Sudoku board (partially filled) is not necessarily solvable. Only the filled cells need to be validated.

Return `0 / 1` (0 for false, 1 for true) for this problem

Largest Continuous Sequence Zero Sum

Problem Description

Given an array **A** of **N** integers.

Find the largest continuous sequence in a array which sums to zero.

Problem Constraints

$1 <= N <= 10^6$

$-10^7 <= A[i] <= 10^7$

Input Format

Single argument which is an integer array **A**.

Output Format

Return an array denoting the longest continuous sequence with total sum of zero.

NOTE : If there are multiple correct answers, return the sequence which occurs first in the array.

Example Input

```
A = [1,2,-2,4,-4]
```

Single Number III

Problem Description

Given an array of numbers **A** , in which exactly two elements appear only once and all the other elements appear exactly twice. Find the two elements that appear only once.

Note: Output array must be sorted.

Problem Constraints

2 <= |A| <= 100000
1 <= A[i] <= 10^9

freq_map.

Input Format

First argument is an array of interger of size N.

Hashmap

Hashset.

Output Format

Return an array of two integers that appear only once.

TC : O(N)
SC : O(N).

1 : X 2
2 : X 2
3 : 1
4 : 1

3, 4

Example Input

Input 1:

```
A = [1, 2, 3, 1, 2, 4]
```

Input 2:

```
A = [1, 2]
```

XOR

1, 2, 3, 1, 2, 4 (a, b)

Xor = (1^1) ^ (2^2) ^ (3^4)

= 0 ^ 0 ^ (3^4)

= 3^4 = 7

a: 0 1 1
b: 1 0 0
^
1 1 1

0th

a^b

a b.

Hint: What info is provided by xor of 2 numbers?

a^b will have ith bit as set if

it is set in a and unset in b

or set in b and unset in a

= different in both a, b.

1, 2, 3, 1, 2, 4

0th bit ⟹ different in a and b.

1	001
2	010
3	011
1	001
2	010
4	100

Segregate numbers based on 0th bit:

0th bit is unset

2 : 010

2 : 010

4 : 100

a ^ 4 : 100

0th bit is set

1 : 001

3 : 011

1 : 001 → a

a ^ 3 : 011

---

1 : 001

2 : 010

3 : 011

1 : 001

2 : 010

5 : 101

a ^ 1 1 0

Rightmost set bit

3 : 011

5 : 101

1 1 0 = 6

N : 110

N-1 : 101

& 100

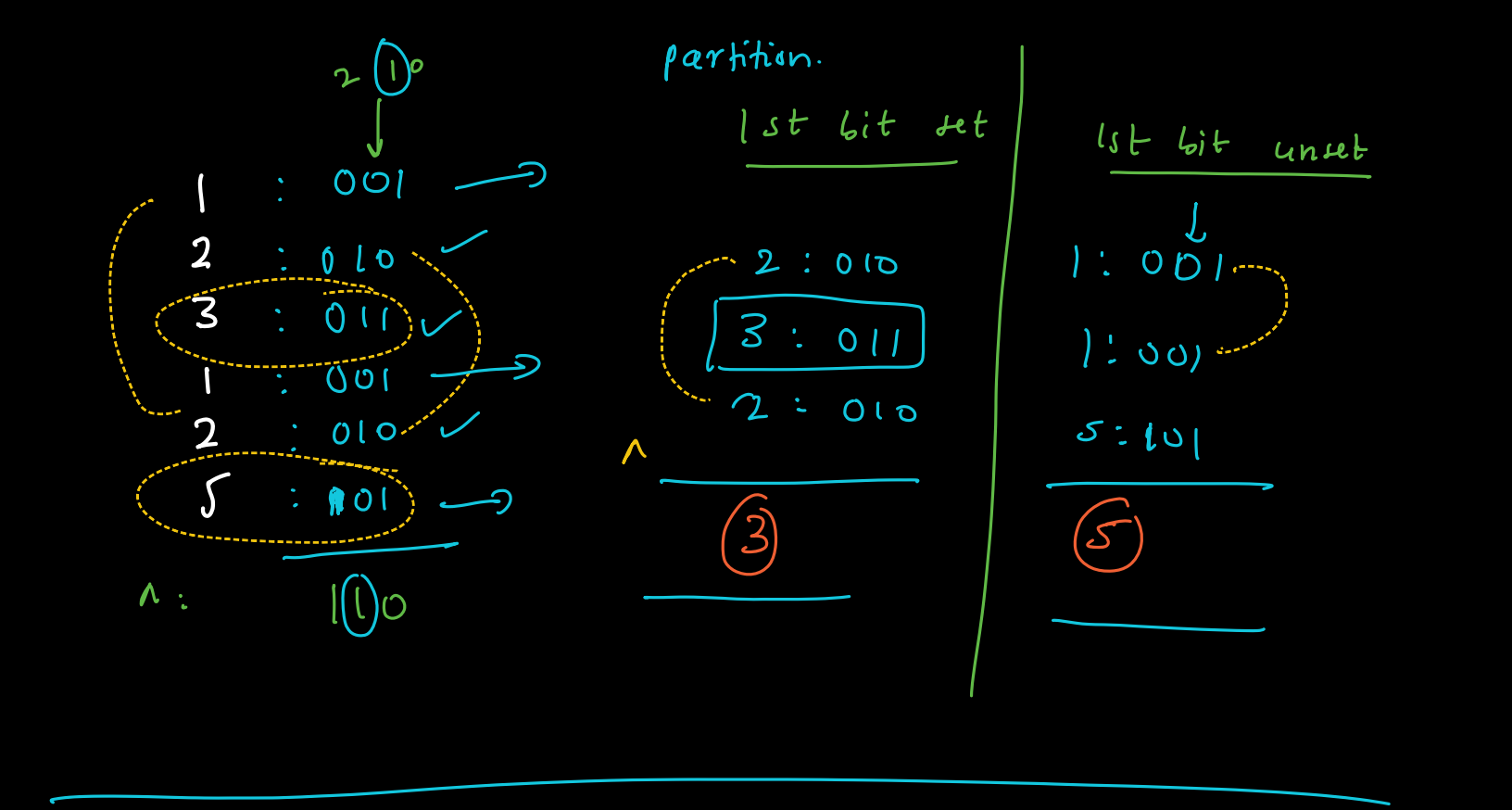
N : 110

N & (N-1) : 100

N ^ 010

N ^ (N & N-1)

a & (~a + 1)


2 (1) 0
1 : 001
2 : 010
3 : 011
1 : 001
2 : 010
5 : 101
^ : 1(1)0
partition.
1st bit set
2 : 010
3 : 011
2 : 010
^
(3)
1st bit unset
1 : 001
1 : 001
5 : 101
(5)

Bit Compression

Problem Description

Richard Hendricks, a mastermind on compression algorithms, in an employee of Hooli in the Silicon Valley.
One day, he finally decided to quit and work on his new idea of middle - out algorithm.

He needed to work at the bit - level to compress data. He, eventually, encountered this problem.
There was an array **A** of **N** integers. He had to perform certain operations on the elements.
In any one operation, two indices **i** and **j** (i < j) are chosen, and **A[i]** is replaced with **A[i] & A[j]**,
and **A[j]** is replaced with **A[i] | A[j]**, where **&** represents the Bitwise AND operation and **|** represents the Bitwise OR operation.
This operation is performed over all the pairs of integers in the array.

Help Richard find the Bitwise XOR of all the elements after performing the operations.

Problem Constraints

$1 <= N <= 10^5$
$1 <= A_i <= 10^9$

Input Format

The first argument is the integer array A.

Output Format

Return a single integer denoting the XOR of the elements after performing the operations.

Max Sum Contiguous Subarray

Problem Description

Find the **contiguous non empty** subarray within an array, **A** of length **N** which has the **largest sum**.

Problem Constraints

1 <= N <= 1e6
-1000 <= A[i] <= 1000

Input Format

The first and the only argument contains an integer array, A.

Output Format

Return an integer representing the maximum possible sum of the contiguous subarray.

Example Input

Input 1:

```
A = [1, 2, 3, 4, -10]
```

Input 2:

Length of longest consecutive ones

Given a binary string **A**. It is allowed to do at most one swap between any 0 and 1. Find and return the length of the longest consecutive 1's that can be achieved.

Input Format

```
The only argument given is string A.
```

Output Format

```
Return the length of the longest consecutive 1's that can be achieved.
```

Constraints

```
1 <= length of string <= 1000000
A contains only characters 0 and 1.
```

For Example

```
Input 1:
    A = "111000"
Output 1:
    3

Input 2:
    A = "111011101"
Output 2:
    7
```

```
4
```

Input 2:

```
6
```

Example Output

Output 1:

```
********
***  ***
**    **
*      *
*      *
**    **
***  ***
********
```

Output 2:

```
************
*****  *****
****    ****
***      ***
**        **
*          *
*          *
**        **
***      ***
****    ****
*****  *****
************
```

Example Input

Input 1:

```
7
```

Input 2:

```
4
```

Example Output

Output 1:

```
*******
*     *
*    *
*   *
* *
**
*
```

Output 2:

```
****
* *
**
*
```