Todays Content:

- Number system basics
- Binary to decimal & viceversa
- Adding 2 Binary Numbers
- Bitwise operators
  - Basic Properties
  - Basic Problems

Number System Basics → Decimal Number System
- Each Digit : [0-9]
- Base power: 10

Base: 10

$10^3\ 10^2\ 10^1\ 10^0$

7 3 4 → 700 + 30 + 4

6 5 9 4 → 6000 + 500 + 90 + 4

2 4 5 → 200 + 40 + 5

Other Number Systems:

Octal
- digit : [0 - 7]
- Each power : 8

$8^1\ 8^0$

$(24)_8$ → $2*8^1 + 4*8^0$ → $16 + 4$ → Decimal: 20

$8^2\ 8^1\ 8^0$

$(125)_8$ → $1*8^2 + 2*8 + 5*8^0$ → $64 + 16 + 5$ → 85

8	85	- 5
8	10	- 2
8	1	- 1
	0	

= 125

Binary
- Every digit [0 1]
- Every power [2]

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$(10110)_2$ → $2^4*1 + 0 + 2^2*1 + 2^1*1$ → $16 + 4 + 2 =$ Decimal: 22

$2^3\ 2^2\ 2^1\ 2^0$

$(1010)_2$ → $2^3*1 + 0 + 2^1*1$ → $8 + 2$ → 10

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$(10100)_2$ → $2^4*1 + 2^2*1$ → $16 + 4$ → 20

$2^2\ 2^1\ 2^0$

$(120)_2$ → $2^2*1 + 2^1*2$ → $4 + 4$ → 8

We cannot have digit 2 in Binary Number System

Decimal to Binary → Long division ?

2 | 37	-1	
2 | 18	-0	
2 | 9	-1	
2 | 4	-0	
2 | 2	-0	
2 | 1	-1	
0		

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$(1\ 0\ 0\ 1\ 0\ 1) = 37$

$32 + 4 + 1 = 37$

2 | 25	-1	
2 | 12	-0	
2 | 6	-0	
2 | 3	-1	
2 | 1	-1	
0		

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$(1\ 1\ 0\ 0\ 1) = 25$

$16 + 8 + 1 = 25$

2 | 19	-1	
2 | 9	-1	
2 | 4	-0	
2 | 2	-0	
2 | 1	-1	
0		

$2^4\ \ 2^1\ 2^0$

$(1\ 0\ 0\ 1\ 1) = 19$

$16 + 2 + 1 = 19$

Note: Keep dividing untill Quotient becomes 0, & keep all remainders in reverse order

Add 2 Decimal Numbers → d = s%10, c = s/10, 10 → Decimal Number System

	13/10	11/10	
9/10	7	8	9
↓			
0	1	4	2
	9%10	13%10	11%10

0 9 3 1

	13/10	8/10	13/10	
11/10	7	8	3	9
↓				
1	3	9	4	8
	1%10	11%10	13%10	8%10	13%10

1 1 7 8 7

Add 2 Binary Numbers: → d = s%2, c = s/2

	1/2	3/2	2/2	1/2		
1/2	0	1	1	0	0	→ 22
↓						
0	0	0	1	1	1	→ 7
	1%2	1%2	3%2	2%2	1%2	

0 1 1 1 0 1 → 29

	2/2	1/2	2/2	1/2		
3/2	1	0	1	0	1	→ 27
↓						
1	1	1	0	1	0	→ 26
	3%2	2%2	1%2	2%2	1%2	

1 1 0 1 0 1 → 53

Why Binary? → Electronics → Voltage/Current

vol > = n : 1
vol < n : 0

int n = 25

In System data stored in binary is ——

decimal → binary

done by system

n → | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
    | 1 | 1 | 0 | 0 | 1 | → Stored in System

Binary → decimal

print(n) = 25

Bit Wise Operators: &, |, ^, ~, <<, >>

Truth Table ?

a&b : If both are 1, it is 1

a|b : If one of them 1, it is 1

a^b : Same Same puppy Shame

~a : 1 ⇄ 0

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

// Basic Problems on Bitwise operators

a = 29, b = 19

	2^4	2^3	2^2	2^1	2^0	
a :	1	1	1	0	1	→ 16+8+4+1 = 29
b :	1	0	0	1	1	→ 16+2+1 = 19

Binary operator

print(a&b): 1 0 0 0 1 → 17

print(a|b): 1 1 1 1 1 → 31

print(a^b): 0 1 1 1 0 → 14

a = 13, b = 10 : TODO

a = 1 1 0 1

b = 1 0 1 0

→ TODO: Cross verify values

print(a&b) = 1 0 0 0 → 8

print(a|b) = 1 1 1 1 → 15

print(a^b) = 0 1 1 1 → 7

// Properties:

a = 10 = 1 0 1 0 (0th bit)
1 = 0 0 0 1
print(a&1) = 0 0 0 0 = 0

a = 11 = 1 0 1 1 (0th bit)
1 = 0 0 0 1
print(a&1) = 0 0 0 1 = 1

a = 14 = 1 1 1 0 (0th bit)
1 = 0 0 0 1
print(a&1) = 0 0 0 0 = 0

a = 13 = 1 1 0 1
1 = 0 0 0 1
print(a&1) = 0 0 0 1 = 1

observations:

```
if(a&1==0){
|  a is even
|  0th bit in a = 0
}
else{
|  a is odd
|  0th bit in a = 1
}
```

a&0 ⇒ 0 a&a ⇒ a

a&1 → 0 : a is even
  ↳ 1 : a is odd

a|0 = a a|a ⇒ a

a|1 → a : a is odd
  ↳ a+1 : a is even

a^0 = a a^a = 0

a^1 → {TODO}

TODO:

a = 1 0 1 0 = a
1 = 0 0 0 1
a|1 = 1 0 1 1 = a+1

a = 1 0 1 1 = a
1 = 0 0 0 1
a|1 = 1 0 1 1 = a

Bitwise Properties :

$a \& b = b \& a$

$a | b = b | a$

$a \wedge b = b \wedge a$

Commutative property

$a \& b \& c = c \& b \& a$

$a \& (b \& c)$

$(b \& c) \& a \rightarrow (c \& b) \& a$

$a \& b \& c = b \& c \& a = c \& a \& b = c \& b \& a = a \& c \& b$

$a | b | c = b | c | a = c | a | b = c | b | a = a | c | b$

$a \wedge b \wedge c = b \wedge c \wedge a = c \wedge a \wedge b = c \wedge b \wedge a = a \wedge c \wedge b$

Associative Property

a	=	1	0	1	1	0
b	=	0	1	1	1	1
c	=	1	0	1	1	1
$a \& b \& c$	=	0	0	1	1	0
$a \& (c \& b)$	=	0	0	1	1	0

Q1 → $x = a \wedge d \wedge a \wedge d \wedge b$; print(n) = b

$a \wedge d \wedge a \wedge d \wedge b = \underset{0}{a \wedge a} \wedge \underset{0}{d \wedge d} \wedge \underset{b}{b} = b$

Q2 → $x = e \wedge f \wedge a \wedge f \wedge e \wedge g \wedge a$; print(n) = g

$e \wedge f \wedge a \wedge f \wedge e \wedge g \wedge a = \underset{0}{e \wedge e} \wedge \underset{0}{a \wedge a} \wedge \underset{0}{f \wedge f} \wedge g$

Q Given N array elements, every element repeats twice except 1 find unique element ? { Array contains only 1 unique element }

ar[5] : { 6 9 6 10 9 } = 10 (unique)

ar[7] : { 12 9 12 8 7 9 8 } = 7 (unique)

ar[5] : { 2 9 7 2 7 } = 9 (unique)

Idea : For every ar[i], Iterate in array, get its occurence = 1

```
int unique(int ar[]){ TC O(N^2) SC O(1)
    int n = ar.length;
    for(int i=0; i<n; i++){
        // ar[i] is element
        c=0
        for( _ ; _ ; _ ){
            ========== } TODO
        }
        If(c==1) { return ar[i]}
    }
}
```

Idea 2: Give xor of Entire array

```
int uniqueElement(int ar[]){
    int ans = 0 ? {0^a==a}
    int n = ar.length
    for(i=0; i<n; i++){
        ans = ans ^ ar[i]
    }
    return ans;
}
```

TC: O(N) SC: O(1)

Tracing:

	0	1	2	3	4
ar[5] : {	2	9	7	2	7 }

ans = 0

ans = ans ^ ar[0] = 2

ans = ans ^ ar[1] = 11

ans = ans ^ ar[2] = 12

ans = ans ^ ar[3] = 14

ans = ans ^ ar[4] = 9

ans = 0 : 0 0 0 0

ar[0] = 2 : 0 0 1 0

---

ans = 2 : 0 0 1 0

ar[1] = 9 : 1 0 0 1

---

ans = 11 : 1 0 1 1

ar[2] = 7 : 0 1 1 1

---

ans = 12 : 1 1 0 0

ar[3] = 2 : 0 0 1 0

---

ans = 14 : 1 1 1 0

ar[4] = 7 : 0 1 1 1

---

ans = 9 : 1 0 0 1

---

left shift: <<

8 bit Number

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

a = 10 : 0 0 0 0 1 0 1 0 = $2^3 + 2^1 = 10 = 10 * 2^0$

discard

a<<1 : 0 0 0 1 0 1 0 0 = $2^4 + 2^2 = 20 = 10 * 2^1$ (add)

discard

a<<2 : 0 0 1 0 1 0 0 0 = $2^5 + 2^3 = 40 = 10 * 2^2$ (add)

discard

a<<3 : 0 1 0 1 0 0 0 0 = $2^6 + 2^4 = 80 = 10 * 2^3$ (add)

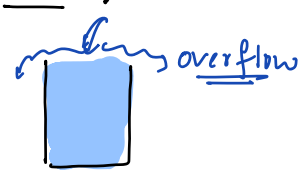
discard

a<<4 : 1 0 1 0 0 0 0 0 = $2^7 + 2^5 = 160 = 10 * 2^4$ (add)

discard

a<<5 : 0 1 0 0 0 0 0 0 = $2^6 = 64$?

Bucket:



overflow: When we are exceding the data we can actually store

// If say no overflow:

$a << n = a * 2^N$

If we replace a = 1

$1 << n = 2^N$

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

a=1 : 0 0 0 0 0 0 0 1

a<<1 : 0 0 0 0 0 0 1 0 = 2^1

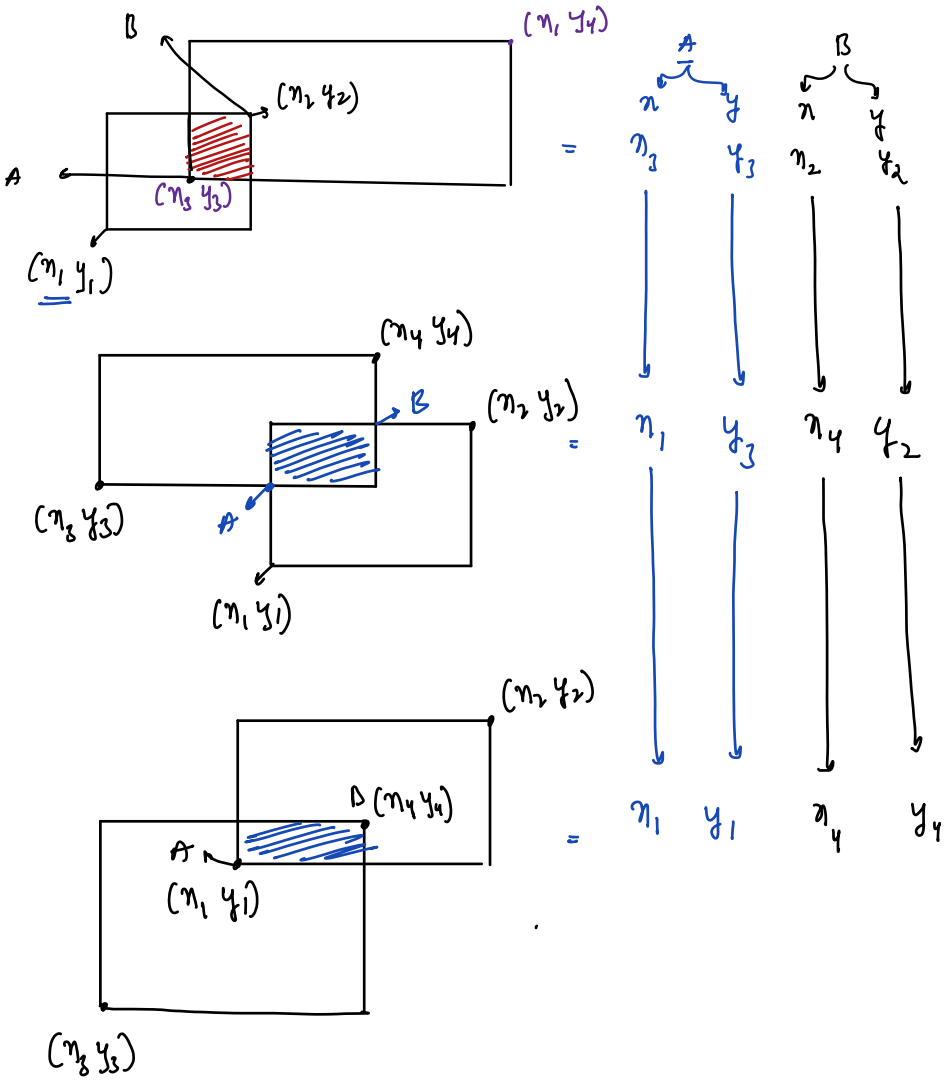
a<<2 : 0 0 0 0 0 1 0 0 = 2^2

a<<3 : 0 0 0 0 1 0 0 0 = 2^3

→ Common Area : → Check overlapping : TODO

→ If overlapped then find points A & B

bottom left top right



obs1: A depends only Both left of Both Rectangles

Rect1 Rect2

A (x, y) → (x_1, y_1) (x_3, y_3)

A.x ⇒ max{ x_1, x_3 }

A.y ⇒ max{ y_1, y_3 }

obs2: B depends only Top Right of Both Rectangles

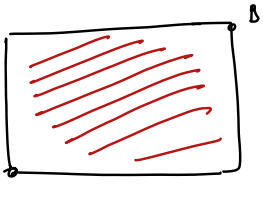
Rect1 Rect2

B (x, y) = (x_2, y_2) (x_4, y_4)

B.x = min{ x_2, x_4 }

B.y = min{ y_2, y_4 }

$(x = \min(x_2, x_4), y = \min(y_2, y_4))$



$(x = \max(x_1, x_3), y = \max(y_1, y_3))$

len = B.x − A.x

bre = B.y − A.y

area = (len * bre)