Bit Manipulation

😤🔥 ठीक है bhai, ab full Google-level BIT BASICS padhate hain — ek-ek cheez tod ke, logic + use + code (Java) + comments ke saath.

Ye foundation hai, agar ye clear ho gaya to **XOR, Bitmasking, Trie, DP sab aasan** lagega.

🧠 PART 1: Binary Representation

🔷 Binary kya hota hai?

Computer **sirf 0 aur 1** samajhta hai.

Decimal	Binary
5	`0101`
7	`0111`
10	`1010`

👉 Har position ka weight hota hai:

```
Binary:  1   0   1   0
Index:   3   2   1   0
Value:   8   4   2   1
```

🔷 Unsigned Binary

- Sirf **positive numbers**
- Example: `8-bit`

```
11111111 = 255
```

🔷 Signed Binary (IMPORTANT)

Java me `int` **32-bit signed** hota hai.

```
[ sign bit ][ rest bits ]
```

- **MSB (Most Significant Bit)** = sign
- `0` → positive
- `1` → negative

🧠 PART 2: 2's Complement (NEGATIVE NUMBERS)

👉 **Negative number ka binary = 2's Complement**

Steps:

1️⃣ Positive number ka binary lo

2️⃣ Sab bits invert karo (1 → 0, 0 → 1)

3️⃣ +1 add karo

Example: `5`

```
+5  = 00000101
Invert → 11111010
+1    → 11111011  ✅ (-5)
```

👉 Isi wajah se:

```
~x + 1 == -x
```

⚠️ **Google favorite question**: why `Integer.MIN_VALUE` ka positive nahi hota?

(kyunki range asymmetric hoti hai)

🧠 PART 3: Left Shift `<<`

Meaning:

```
x << n  ==  x * (2^n)
```

Example:

```
5 << 1 = 10
Binary:
0101 → 1010
```

👉 Use cases:

- Fast multiply
- Bit masking
- Power of two generation

🧠 PART 4: Right Shift `>>` & `>>>`

`>>` (Arithmetic Right Shift)

- Sign bit copy hota hai

```
-8 >> 1 = -4
```

`>>>` (Logical Right Shift)

- Zero fill hota hai (even for negative)

```
-8 >>> 1 = big positive number
```

👉 **Google asks difference 😈**

🧠 PART 5: MSB & LSB

🔷 LSB (Least Significant Bit)

- Rightmost bit
- Even / Odd check

```
x & 1
```

🔷 MSB (Most Significant Bit)

- Leftmost bit
- Sign decide karta hai

✅ SOLVED EXAMPLES (Java + Line by Line)

🟢 EASY — Check Even / Odd

Logic:

- LSB = 0 → Even
- LSB = 1 → Odd

```
class Solution {
    public static boolean isEven(int n) {

        // n & 1 checks last bit
        // If last bit is 0 → even
        return (n & 1) == 0;
    }

    public static void main(String[] args) {
        System.out.println(isEven(10)); // true
        System.out.println(isEven(7));  // false
    }
}
```

🟡 MEDIUM — Find MSB position of a number

Example:

```
n = 18 → 10010
MSB index = 4
```

```
class Solution {
    public static int findMSB(int n) {

        int pos = -1;

        // Jab tak number 0 nahi ho jata
        while (n > 0) {

            // Right shift number
            n = n >> 1;

            // Position increase
            pos++;
        }

        return pos;
    }

    public static void main(String[] args) {
        System.out.println(findMSB(18)); // 4
    }
}
```

🧠 **Why works?**

Right shift se number chhota hota jata hai → jitni baar shift hua wahi MSB index.

🔴 HARD — Check if number is power of 2 (WITHOUT loop)

Property:

```
Power of 2 → only ONE bit set
```

```
n & (n-1) == 0
```

Example:

```
8  = 1000
7  = 0111
&  = 0000
```

```
class Solution {
    public static boolean isPowerOfTwo(int n) {

        // Negative & zero case
        if (n <= 0) return false;

        // Single set bit check
        return (n & (n - 1)) == 0;
    }

    public static void main(String[] args) {
        System.out.println(isPowerOfTwo(16)); // true
        System.out.println(isPowerOfTwo(18)); // false
    }
}
```

🔥 **Google + LeetCode #231**

🧪 UNSOLVED PRACTICE (Try yourself 😈)

📝 Practice 1 (Easy)

👉 Find **LSB position** of a number

Input: `10 (1010)`

Output: `1`

📝 Practice 2 (Medium)

👉 Without using `+ - * /`, **add two integers**

📝 Practice 3 (Hard)

👉 Given an integer, **toggle its MSB**

Example:

```
10 → 1010
Toggle MSB → 0010 → 2
```

🚀 Next Step

Bhai ye clear ho gaya to next level unlock:

➡️ **XOR PATTERNS (Google ka favourite weapon)**

➡️ **Bit Masking (subset / permissions)**

Bas bol:

> "XOR patterns padha de Google level"

Aur fir dekh magic 😈🔥

😤👍 **sahi pakda bhai** — jab tak `>>` aur `>>>` **dimaag ke andar picture ban ke clear** nahi hote, tab tak bit manipulation half-cooked hi rehta hai.

Ab main **slow + visual + dry run + real use** ke saath samjha raha hoon. Ye wahi level hai jo **Google interview me bol ke explain** kar sakta hai.

🧠 STEP 0: 32-bit INT ka REAL STRUCTURE (Java)

Java ka `int` = **32 bits**

```
[ b31 ][ b30 ][ b29 ] ... [ b1 ][ b0 ]
```

- **b31 = MSB (sign bit)**
- `0` → positive
- `1` → negative

🧠 STEP 1: RIGHT SHIFT `>>` (Arithmetic Right Shift)

👉 Kya karta hai?

- Bits ko **right side shift**
- **Left side me sign bit copy** karta hai

Isliye:

- Positive → left me `0` bhar deta hai
- Negative → left me `1` bhar deta hai

🔷 Example 1: Positive Number

`10 >> 1`

```
Decimal: 10
Binary : 00000000 00000000 00000000 00001010

>> 1
---------------------------------------
00000000 00000000 00000000 00000101
```

👉 Result = **5**

💡 Formula:

```
x >> n  ≈  x / (2^n)
```

🔷 Example 2: Negative Number (IMPORTANT)

`10 >> 1`

Step 1: +10 ka binary

```
10 = 00000000 00000000 00000000 00001010
```

Step 2: 2's complement (for -10)

```
Invert → 11111111 11111111 11111111 11110101
+1     → 11111111 11111111 11111111 11110110  (-10)
```

Step 3: Right shift `>> 1`

```
11111111 11111111 11111111 11110110
>>
11111111 11111111 11111111 11111011
```

👉 Result = **-5**

🔥 **Notice**: left me `1` fill hua (sign preserved)

🧠 USE CASE of `>>`

✔️ Fast divide

✔️ Math operations

✔️ MSB nikalna

✔️ Sign-preserving operations

🧠 STEP 2: LOGICAL RIGHT SHIFT `>>>`

👉 Kya karta hai?

- Bits right shift
- **Left me hamesha 0 bharta hai**
- **Sign ignore karta hai**

⚠️ Java me sirf `>>>` hota hai (C++ me nahi)

🔷 Example 3: Same number, different result

`10 >>> 1`

Original -10:

```
11111111 11111111 11111111 11110110
```

`>>> 1`

```
01111111 11111111 11111111 11111011
```

👉 Result = **2147483643** 😱 (huge positive)

🧠 DIFFERENCE TABLE

Operator	Name	Sign Preserve?	Left Fill
`>>`	Arithmetic Right Shift	YES	MSB
`>>>`	Logical Right Shift	NO	0

🧠 STEP 3: REAL INTERVIEW USE CASES

🔥 Use Case 1: Check sign (positive / negative)

```
int sign = n >> 31;
```

Result	Meaning
0	Positive
-1	Negative

🔥 Use Case 2: Convert negative to positive bits

```
int unsigned = n >>> 0;
```

Used in:

✔️ Hashing

✔️ Bit masking

✔️ Low-level operations

🔥 Use Case 3: MSB position using `>>>`

Example: `n = 18`

```
10010
```

Dry Run:

```
18 >>> 1 = 9
9  >>> 1 = 4
4  >>> 1 = 2
2  >>> 1 = 1
1  >>> 1 = 0
```

Count = **4** → MSB index

🧠 FULL CODE with DRY RUN COMMENTS

```
class Solution {

    public static void main(String[] args) {

        int n = -10;

        // Arithmetic Right Shift
        int a = n >> 1;

        // Logical Right Shift
        int b = n >>> 1;

        System.out.println(a); // -5
        System.out.println(b); // 2147483643
    }
}
```

🧪 DRY RUN SUMMARY

```
n = -10
Binary = 11111111 11111111 11111111 11110110

n >> 1
→ sign bit copied
→ stays negative

n >>> 1
→ zero fill
→ becomes positive
```

🔴 PRACTICE (UNSOLVED – GOOGLE STYLE)

1️⃣ Without using if/else, find if number is negative

2️⃣ Find MSB position using only `>>>`

3️⃣ Implement divide by 2^k using bit operators (handle negative)

🚀 Next Level?

Ab tu bol:

> "XOR patterns + dry run padha de"

Fir asli Google wale sawal shuru honge 😈🔥

😂😂 **arey bhai shaant, ab full package milega**

Starting se **binary → 2's complement → >> >>> → MSB/LSB**

👉 **HAR solved example ke saath:**

- concept
- dry run (bit-by-bit)

- Java full code (line-by-line comments)
- **LeetCode / Codeforces reference**
- end me **unsolved practice**

Socha tha skip kar dunga, par ab **Google-ready bana ke chhodunga** 😤🔥

🔰 PART 1: Binary Representation (SIGNED / UNSIGNED)

🔷 Binary kya hota hai?

Computer base-2 me kaam karta hai.

Example:

```
Decimal: 13
Binary : 1101
```

👉 Position wise value:

```
Bit:    1  1  0  1
Index:  3  2  1  0
Value:  8  4  2  1
Sum = 8 + 4 + 1 = 13
```

🔷 Signed vs Unsigned (Java POV)

Java me `int`:

- **32-bit**
- **signed**
- MSB = sign bit

```
0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx → positive
1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx → negative
```

🔰 PART 2: 2’s Complement (NEGATIVE NUMBERS)

❓ Question: Computer `5` kaise store karta hai?

Step 1: +5 ka binary

```
00000000 00000000 00000000 00000101
```

Step 2: Invert bits

```
11111111 11111111 11111111 11111010
```

Step 3: +1

```
11111111 11111111 11111111 11111011  → -5
```

👉 **Ye hi 2’s complement hai**

🔥 Property (INTERVIEW GOLD)

```
-x = ~x + 1
```

🔰 PART 3: LSB & MSB (USE + DRY RUN)

🔷 LSB (Least Significant Bit)

- Rightmost bit
- Even / Odd batata hai

```
n & 1
```

✅ SOLVED EXAMPLE 1 (EASY)

🔷 Check Even / Odd

📌 **LeetCode 258 – Add Digits (LSB logic use hota hai)**

📌 **CF: 4A – Watermelon**

Dry Run:

```
n = 6 → 110
LSB = 0 → even

n = 7 → 111
LSB = 1 → odd
```

Java Code (FULL COMMENTED)

```
class Solution {

    // Function to check even or odd
    static boolean isEven(int n) {

        // n & 1 checks the last bit (LSB)
        // If last bit is 0 → even
        // If last bit is 1 → odd
        return (n & 1) == 0;
    }

    public static void main(String[] args) {
        System.out.println(isEven(6)); // true
        System.out.println(isEven(7)); // false
    }
}
```

🔷 MSB (Most Significant Bit)

- Leftmost `1`
- Power, range, sign sab batata hai

✅ SOLVED EXAMPLE 2 (MEDIUM)

🔷 Find MSB position

📌 LeetCode 231 – Power of Two

📌 CF: 1368A – C+=

Example:

```
n = 18
Binary = 10010
MSB index = 4
```

Dry Run:

```
18 >> 1 = 9   (1001)
9  >> 1 = 4   (100)
4  >> 1 = 2   (10)
2  >> 1 = 1   (1)
1  >> 1 = 0   stop
count = 4
```

Java Code

```
class Solution {

    static int findMSB(int n) {

        int pos = -1;

        // Keep shifting right until number becomes 0
        while (n > 0) {
            n = n >> 1;   // divide by 2
            pos++;        // count shifts
        }

        return pos;
    }
```

```
    public static void main(String[] args) {
        System.out.println(findMSB(18)); // 4
    }
}
```

🔰 PART 4: RIGHT SHIFT `>>` vs `>>>`

🔷 `>>` (Arithmetic Right Shift)

- Sign bit copy hota hai
- Negative negative hi rehta hai

🔷 `>>>` (Logical Right Shift)

- Left me **0 fill**
- Sign ignore karta hai

✅ SOLVED EXAMPLE 3 (HARD / CONFUSING)

🔷 Compare `>>` and `>>>`

📌 **LeetCode 190 – Reverse Bits**

Example:

```
n = -8
Binary = 11111111 11111111 11111111 11111000
```

`n >> 1`

```
11111111 11111111 11111111 11111100 → -4
```

`n >>> 1`

```
01111111 11111111 11111111 11111100 → 2147483644
```

Java Code (Clear Difference)

```
class Solution {

    public static void main(String[] args) {

        int n = -8;

        // Arithmetic shift (sign preserved)
        int a = n >> 1;

        // Logical shift (zero fill)
        int b = n >>> 1;

        System.out.println(a); // -4
        System.out.println(b); // 2147483644
    }
}
```

🔥 GOOGLE FAVORITE SOLVED EXAMPLE

✅ Power of Two

📌 **LeetCode 231**

📌 **CF: 702A**

Logic:

```
n = 8 → 1000
n-1 = 7 → 0111
AND = 0
```

Java Code

```
class Solution {

    public boolean isPowerOfTwo(int n) {

        // Power of two must be positive
        if (n <= 0) return false;

        // Only one bit should be set
        return (n & (n - 1)) == 0;
    }
}
```

🧪 UNSOLVED PRACTICE (TRY YOURSELF 😈)

📝 Practice 1 (Easy)

📌 **LC 191 – Number of 1 Bits**

👉 Count set bits using right shift

📝 Practice 2 (Medium)

📌 **LC 476 – Number Complement**

👉 Flip all bits of a number

📝 Practice 3 (Hard)

📌 **LC 201 – Bitwise AND of Numbers Range**

👉 MSB concept use hota hai

🚀 NEXT STEP

Agar ye clear ho gaya na bhai,

to next bolega:

> “XOR patterns full detail + dry run + LC questions”

Fir tu bolega:

“abe bit manipulation to easy nikla” 😈🔥

😈🔥 ab asli BIT MANIPULATION ka engine start hota hai

Ye **CORE OPERATORS** samajh aa gaye na bhai, to **Google / LeetCode / Codeforces ke 70% bit questions yahin se ho jaate hain**.

Main **starting se**, **truth table**, **real-life analogy**, **dry run**, **SOLVED examples (Easy–Medium–Hard)** + **Java full code with comments** + **LC / CF numbers** + **unsolved practice** sab dunga.

🔰 CORE BIT OPERATORS (MUST-MASTER)

1️⃣ AND (`&`)

2️⃣ OR (`|`)

3️⃣ XOR (`^`)

4️⃣ NOT (`~`)

1️⃣ AND OPERATOR (`&`)

🔷 Truth Table

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

👉 Rule:

Dono bits 1 honi chahiye

🔷 Binary Example

```
a = 10 → 1010
b =  6 → 0110
--------------
a&b =    0010 → 2
```

🔥 REAL USE CASES (IMPORTANT)

✅ Even / Odd check

✅ Check ith bit set or not

✅ Clear bits

✅ Power of two check

✅ SOLVED EXAMPLE 1 (EASY)

🔷 Check ith bit set or not

📌 **LeetCode 231 – Power of Two (AND logic)**

Problem:

👉 Check whether **3rd bit** of number `n` is set

Logic:

```
mask = 1 << i
if (n & mask) != 0 → bit set
```

Dry Run:

```
n = 10 → 1010
i = 1
mask = 0010

1010
0010
```

```
----
0010 != 0 → bit is set
```

Java Code (Fully Commented)

```
class Solution {

    static boolean isBitSet(int n, int i) {

        // Create mask by shifting 1 to ith position
        int mask = 1 << i;

        // AND operation checks if ith bit is 1
        return (n & mask) != 0;
    }

    public static void main(String[] args) {
        System.out.println(isBitSet(10, 1)); // true
        System.out.println(isBitSet(10, 2)); // false
    }
}
```

2️⃣ OR OPERATOR (|)

🔷 Truth Table

A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

👉 **Rule:**

Koi ek bhi bit 1 hui → result 1

🔷 Binary Example

```
a = 10 → 1010
b =  6 → 0110
--------------
a|b =    1110 → 14
```

🔥 REAL USE CASES

✅ Set ith bit

✅ Permissions / flags

✅ Feature enable logic

✅ SOLVED EXAMPLE 2 (MEDIUM)

🔷 Set ith bit of a number

📌 **Codeforces: 1760A – Medium style bit usage**

Problem:

👉 Set **2nd bit** of number `n`

Logic:

```
n | (1 << i)
```

Dry Run:

```
n = 8 → 1000
i = 2
mask = 0100

1000
0100
----
1100 → 12
```

Java Code

```
class Solution {

    static int setBit(int n, int i) {

        // Shift 1 to ith position
        int mask = 1 << i;

        // OR operation sets the bit
        return n | mask;
    }

    public static void main(String[] args) {
        System.out.println(setBit(8, 2)); // 12
    }
}
```

3️⃣ XOR OPERATOR (^) 🔥🔥🔥 (GOOGLE FAVORITE)

🔷 Truth Table

A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

👉 **Rule:**

Same bits → 0

Different bits → 1

🔥 GOLDEN PROPERTIES

```
x ^ x = 0
x ^ 0 = x
x ^ y ^ x = y
```

🔥 REAL USE CASES

✅ Find single number

✅ Toggle bits

✅ Swap numbers

✅ Missing number problems

✅ SOLVED EXAMPLE 3 (MEDIUM – CLASSIC)

🔷 Find single number

📌 **LeetCode 136 – Single Number**

📌 **CF: 1915E**

Problem:

👉 Array me har number **2 baar**, ek **1 baar**

Logic:

```
ans = a ^ a ^ b ^ b ^ c = c
```

Dry Run:

```
nums = [4,1,2,1,2]

ans = 0
0^4 = 4
4^1 = 5
5^2 = 7
7^1 = 6
6^2 = 4
```

Java Code

```
class Solution {

    public int singleNumber(int[] nums) {

        int ans = 0;

        // XOR all numbers
        for (int num : nums) {
            ans = ans ^ num;
        }

        // Remaining number is unique
        return ans;
    }
}
```

4️⃣ NOT OPERATOR (~)

🔷 Truth Table

A	~A
0	1
1	0

👉 **All bits flip ho jaati hain**

🔷 Example

```
n = 5 → 00000101
~n    → 11111010 → -6
```

👉 **Why -6?**

```
~n = -(n + 1)
```

✅ SOLVED EXAMPLE 4 (HARD-ish)

🔷 Number Complement

📌 **LeetCode 476**

Problem:

👉 Flip only **significant bits**

Example:

```
n = 5 → 101
Complement → 010 → 2
```

Java Code

```
class Solution {

    public int findComplement(int num) {

        int mask = 0;
        int temp = num;

        // Create mask with all bits set till MSB
        while (temp > 0) {
            mask = (mask << 1) | 1;
            temp = temp >> 1;
        }

        // XOR with mask flips only required bits
        return num ^ mask;
    }
}
```

🧪 UNSOLVED PRACTICE (MUST TRY)

📝 Easy

📌 **LC 191 – Number of 1 Bits**

📝 Medium

📌 **LC 268 – Missing Number**

📝 Hard

📌 **LC 201 – Bitwise AND of Numbers Range**

🚀 NEXT STEP (VERY IMPORTANT)

Ab bol bhai:

> “XOR patterns full deep dive + 10 Google questions”

Fir bit manipulation tera **weapon** ban jaayega 😈🔥

😎🔥 **bhai ab Google ka favourite zone aa gaya**

Ye **Power & Bit Checks** aise topics hain jinke bina **FAANG interviews incomplete** hote hain.

Main **starting se**, **formula + dry run + fully commented Java code**, **LC + CF references**, aur end me **unsolved practice** de raha hoon.

🚀 POWER & CHECKS (GOOGLE LOVES THESE)

Topics Covered

1️⃣ Check even / odd using bit

2️⃣ Check power of 2

3️⃣ Check power of 4

4️⃣ Check ith bit set or not

5️⃣ Set ith bit

6️⃣ Unset (clear) ith bit

7️⃣ Toggle ith bit

1️⃣ Check EVEN / ODD using BIT

🔷 Logic

👉 **LSB (last bit)** decide karta hai

- `0` → even
- `1` → odd

```
n & 1
```

✅ SOLVED (EASY)

📌 **Codeforces: 4A – Watermelon**

📌 **LC usage: everywhere (fundamental)**

Dry Run

```
n = 10 → 1010
LSB = 0 → even

n = 7 → 111
LSB = 1 → odd
```

Java Code

```
class Solution {

    static boolean isEven(int n) {

        // AND with 1 checks last bit
        return (n & 1) == 0;
    }
```

```
    public static void main(String[] args) {
        System.out.println(isEven(10)); // true
        System.out.println(isEven(7));  // false
    }
}
```

2️⃣ Check POWER OF 2 🔥🔥

🔷 Concept (VERY IMPORTANT)

Power of 2 → **only ONE bit set**

```
n & (n - 1) == 0
```

Dry Run

```
n = 8  → 1000
n-1=7  → 0111
AND    → 0000 ✔
```

✅ SOLVED (MEDIUM)

📌 **LeetCode 231 – Power of Two**

📌 **CF: 702A – Maximum Increase (logic use hota hai)**

Java Code

```
class Solution {

    public boolean isPowerOfTwo(int n) {

        // Power of 2 must be positive
        if (n <= 0) return false;
```

```
        // Only one bit set
        return (n & (n - 1)) == 0;
    }
}
```

3️⃣ Check POWER OF 4 (Google Twist 😈)

🔷 Concept

- Power of 2 ✔️
- AND **odd position bit** (0,2,4...)

```
n & (n - 1) == 0
AND
(n & 0x55555555) != 0
```

Dry Run

```
n = 16 → 10000
Mask  → 10101
AND   → non-zero ✔
```

✅ SOLVED (MEDIUM)

📌 **LeetCode 342 – Power of Four**

Java Code

```
class Solution {

    public boolean isPowerOfFour(int n) {

        if (n <= 0) return false;

        // Check power of 2 AND odd position bit
```

```
        return (n & (n - 1)) == 0 && (n & 0x55555555) != 0;
    }
}
```

4️⃣ Check ITH BIT SET OR NOT

🔷 Logic

```
mask = 1 << i
n & mask
```

Dry Run

```
n = 10 → 1010
i = 1
mask = 0010

AND → 0010 ✔ (set)
```

✅ SOLVED (EASY)

📌 **CF: 1742E – Scuza (bit checking idea)**

Java Code

```
class Solution {

    static boolean isIthBitSet(int n, int i) {

        int mask = 1 << i;

        return (n & mask) != 0;
    }

    public static void main(String[] args) {
```

```
        System.out.println(isIthBitSet(10, 1)); // true
        System.out.println(isIthBitSet(10, 2)); // false
    }
}
```

5️⃣ SET ITH BIT

🔷 Logic

```
n | (1 << i)
```

✅ SOLVED (EASY)

📌 **CF: 1760A – Medium Number**

Dry Run

```
n = 8 → 1000
i = 2
mask = 0100

OR → 1100 → 12
```

Java Code

```
class Solution {

    static int setBit(int n, int i) {

        return n | (1 << i);
    }

    public static void main(String[] args) {
        System.out.println(setBit(8, 2)); // 12
```

```
    }
}
```

6️⃣ UNSET (CLEAR) ITH BIT

🔷 Logic

```
n & ~(1 << i)
```

✅ SOLVED (MEDIUM)

📌 LC 190 – Reverse Bits (clearing bits idea)

Dry Run

```
n = 10 → 1010
i = 1
mask = ~(0010) → 1101

AND → 1000 → 8
```

Java Code

```
class Solution {

    static int clearBit(int n, int i) {

        return n & ~(1 << i);
    }

    public static void main(String[] args) {
        System.out.println(clearBit(10, 1)); // 8
    }
}
```

7️⃣ TOGGLE ITH BIT 🔥

🔷 Logic

```
n ^ (1 << i)
```

✅ SOLVED (MEDIUM)

📌 CF: 1805A – We Need the Zero

Dry Run

```
n = 10 → 1010
i = 1
mask = 0010

XOR → 1000 → 8
```

Java Code

```
class Solution {

    static int toggleBit(int n, int i) {

        return n ^ (1 << i);
    }

    public static void main(String[] args) {
        System.out.println(toggleBit(10, 1)); // 8
        System.out.println(toggleBit(8, 3));  // 0
    }
}
```

🧪 UNSOLVED PRACTICE (GOOGLE LEVEL)

📝 Easy

📌 LC 191 – Number of 1 Bits

📝 Medium

📌 LC 268 – Missing Number

📝 Hard

📌 LC 201 – Bitwise AND of Numbers Range

🔥 NEXT LEVEL

Ab bhai bolega to next unlock hoga:

> "XOR PATTERNS (single, missing, two unique, range xor)"

Uske baad bit manipulation **game** ban jaayega 😈🔥

😈🔥 BHAI AB ASLI SONA

Agar **XOR patterns** solid ho gaye na, to **Google / Amazon / Microsoft** ke aadhe bit questions *automatic* ho jaate hain.

Main bilkul **zero se**, **properties → pattern → dry run → solved examples (LC + CF) → full Java code with line-by-line comments → unsolved practice** sab de raha hoon.

⚠️ Ye section dhyaan se padhna, yahin se bit manipulation click karta hai.

🔥 XOR PATTERNS (SUPER IMPORTANT)

Topics:

1️⃣ XOR properties

2️⃣ Find single number

3️⃣ Find missing number

4️⃣ Two non-repeating numbers

5️⃣ XOR from 1 to N trick

1️⃣ XOR PROPERTIES (FOUNDATION)

🔷 Truth Table

A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

👉 **Same → 0, Different → 1**

🔷 GOLDEN PROPERTIES (YAAD KAR LE 🔥)

```
x ^ x = 0
x ^ 0 = x
x ^ y ^ x = y
XOR is commutative & associative
```

👉 **Order matter nahi karta**

🧠 Real-life analogy

👉 XOR = **pair cancel karne ki machine**

2️⃣ FIND SINGLE NUMBER (CLASSIC)

🔷 Problem

Array me:

- Har number **2 baar**
- Ek number **1 baar**

🧠 Idea

```
a ^ a = 0
0 ^ b = b
```

✅ SOLVED EXAMPLE (EASY)

📌 **LeetCode 136 – Single Number**

📌 **CF: 1915E – XOR and Array**

Example

```
nums = [4,1,2,1,2]
```

Dry Run

```
ans = 0
0 ^ 4 = 4
4 ^ 1 = 5
5 ^ 2 = 7
7 ^ 1 = 6
6 ^ 2 = 4
```

👉 Answer = **4**

Java Code (FULL COMMENTED)

```
class Solution {

    public int singleNumber(int[] nums) {

        int ans = 0; // start with 0

        // XOR all numbers
        for (int num : nums) {
            ans = ans ^ num; // duplicate numbers cancel out
```

```
        }

        // remaining number is unique
        return ans;
    }
}
```

3️⃣ FIND MISSING NUMBER (0 to N)

🔷 Problem

Array size = `n`

Numbers = `0 to n`

Ek number missing

🧠 Idea

```
(0 ^ 1 ^ 2 ^ ... ^ n) ^ (array XOR)
```

✅ SOLVED EXAMPLE (MEDIUM)

📌 LeetCode 268 – Missing Number

📌 CF: 1552A – Subsequence Permutation (idea)

Example

```
nums = [3,0,1]
n = 3
```

Dry Run

```
xorAll = 0^1^2^3 = 0
xorArr = 3^0^1 = 2
missing = 0 ^ 2 = 2
```

Java Code

```
class Solution {

    public int missingNumber(int[] nums) {

        int xor = 0;
        int n = nums.length;

        // XOR all indices and values
        for (int i = 0; i < n; i++) {
            xor = xor ^ i ^ nums[i];
        }

        // finally XOR with n
        xor = xor ^ n;

        return xor;
    }
}
```

4️⃣ TWO NON-REPEATING NUMBERS 🔥🔥

🔷 Problem

Array me:

- Sab numbers **2 baar**
- **2 numbers sirf 1 baar**

🧠 Idea (INTERVIEW FAVORITE)

1️⃣ XOR all → `xor = a ^ b`

2️⃣ Find rightmost set bit

3️⃣ Divide numbers into 2 groups

4️⃣ XOR separately

✅ SOLVED EXAMPLE (HARD)

📌 **LeetCode 260 – Single Number III**

📌 **CF: 1918C – XOR and Or**

Example

```
nums = [1,2,1,3,2,5]
```

Dry Run

```
xor = 1^2^1^3^2^5 = 6 (110)
rightmost set bit = 2 (010)
```

Groups:

```
Group 1: [2,2,3]
Group 2: [1,1,5]
```

Results:

```
3 and 5
```

Java Code (FULL)

```
class Solution {

    public int[] singleNumber(int[] nums) {

        int xor = 0;

        // Step 1: XOR all numbers
        for (int num : nums) {
            xor ^= num;
        }
```

```
        // Step 2: rightmost set bit
        int diffBit = xor & (-xor);

        int a = 0, b = 0;

        // Step 3: divide into two groups
        for (int num : nums) {
            if ((num & diffBit) == 0)
                a ^= num;
            else
                b ^= num;
        }

        return new int[]{a, b};
    }
}
```

5️⃣ XOR FROM 1 TO N TRICK ⚡

🔷 Pattern (VERY IMPORTANT)

N % 4	XOR (1 → N)
0	N
1	1
2	N + 1
3	0

🧠 Why?

XOR pattern repeats every **4 numbers**

✅ SOLVED EXAMPLE (MEDIUM)

📌 CF: 1741C – Minimize the Thickness

Example

```
N = 5
5 % 4 = 1 → answer = 1
```

Java Code

```
class Solution {

    static int xorFrom1ToN(int n) {

        if (n % 4 == 0) return n;
        if (n % 4 == 1) return 1;
        if (n % 4 == 2) return n + 1;

        return 0;
    }

    public static void main(String[] args) {
        System.out.println(xorFrom1ToN(5)); // 1
    }
}
```

🧪 UNSOLVED PRACTICE (MUST TRY)

📝 Easy

📌 LC 389 – Find the Difference

📝 Medium

📌 LC 1310 – XOR Queries of a Subarray

📝 Hard

📌 LC 1734 – Decode XORed Permutation

🔥 NEXT LEVEL (GOOGLE MODE)

Next bolega to yeh unlock hoga:

> “Bit Masking + Subsets + State Compression”

Uske baad tu bolega:

“abe Google wale bit se darte kyo hain 😈”

😈🏆 BHAI AB INTERVIEW GOLD ZONE

Bit Masking wahi cheez hai jahan se **Google / Uber / Atlassian / Airbnb** ke *real thinking questions* aate hain.

Tu agar ye samajh gaya na, to **subset, DP, permissions, configuration, scheduling** sab clear ho jaayega.

Main **zero se**, **intuition → pattern → dry run → solved examples (LC + CF)** → **Java full code (line-by-line comments)** → **unsolved practice** sab de raha hoon.

🏆 BIT MASKING (INTERVIEW GOLD)

Topics

1️⃣ Bitmask creation

2️⃣ Subsets using bitmask

3️⃣ Permissions / Flags problems

4️⃣ State Compression idea (intro)

1️⃣ BITMASK CREATION (FOUNDATION)

🔷 Bitmask kya hota hai?

👉 **Ek integer jisme har bit kisi state / feature ko represent karti hai**

Example:

```
Bit position:  3 2 1 0
Mask:          1 0 1 1
```

Meaning: ON OFF ON ON

🔷 Common Operations (YAAD KAR LE 🔥)

Operation	Formula
Check ith bit	`(mask & (1<<i)) != 0`
Set ith bit	`mask
Clear ith bit	`mask & ~(1<<i)`
Toggle ith bit	`mask ^ (1<<i)`

✅ SOLVED EXAMPLE 1 (EASY)

🔷 Create mask for characters present

📌 LeetCode 318 – Maximum Product of Word Lengths

Idea

- Each letter → ek bit
- `'a'` → bit 0, `'b'` → bit 1 ...

Dry Run

```
word = "abc"
mask = 000...000
a → set bit 0 → 0001
b → set bit 1 → 0011
c → set bit 2 → 0111
```

Java Code

```
class Solution {

    static int createMask(String word) {

        int mask = 0;
```

```
        for (char ch : word.toCharArray()) {
            int bit = ch - 'a';      // position of character
            mask = mask | (1 << bit); // set that bit
        }

        return mask;
    }

    public static void main(String[] args) {
        System.out.println(createMask("abc")); // 7 (111)
    }
}
```

2️⃣ SUBSETS USING BITMASK 🔥🔥

🔷 Core Idea

- Agar array size = `n`
- Total subsets = `2^n`
- Numbers `0 → (2^n - 1)` act as masks

🔷 Mapping

```
nums = [1,2,3]

mask = 101
means:
bit0 → include 1
bit2 → include 3
subset = [1,3]
```

✅ SOLVED EXAMPLE 2 (MEDIUM)

🔷 Generate all subsets

📌 **LeetCode 78 – Subsets**

📌 **CF: 550A – Two Substrings (idea)**

Dry Run

```
nums = [1,2,3]
n = 3
masks = 0 → 7

mask = 3 → 011
→ include 1,2
```

Java Code (FULL)

```
import java.util.*;

class Solution {

    public List<List<Integer>> subsets(int[] nums) {

        List<List<Integer>> result = new ArrayList<>();
        int n = nums.length;

        // Total subsets = 2^n
        for (int mask = 0; mask < (1 << n); mask++) {

            List<Integer> subset = new ArrayList<>();

            // Check each bit
            for (int i = 0; i < n; i++) {

                // If ith bit is set, include nums[i]
                if ((mask & (1 << i)) != 0) {
                    subset.add(nums[i]);
                }
            }
```

```
            result.add(subset);
        }

        return result;
    }
}
```

3️⃣ PERMISSIONS / FLAGS PROBLEMS 🔐 (REAL LIFE USE)

🔷 Real World Example

Linux permissions:

```
READ  = 4 (100)
WRITE = 2 (010)
EXEC  = 1 (001)
```

🔷 Combine permissions

```
READ + WRITE → 110 (6)
```

✅ SOLVED EXAMPLE 3 (MEDIUM – INTERVIEW TYPE)

🔷 Check if user has WRITE permission

📌 CF: 1665B – Array Cloning Technique (flag idea)

Dry Run

```
permissions = 6 → 110
WRITE = 2 → 010
```

```
AND → 010 → yes
```

Java Code

```
class Solution {

    static final int READ = 1 << 2;   // 100
    static final int WRITE = 1 << 1;  // 010
    static final int EXEC = 1 << 0;   // 001

    static boolean hasWritePermission(int mask) {

        // Check WRITE bit
        return (mask & WRITE) != 0;
    }

    public static void main(String[] args) {

        int userPermission = READ | WRITE; // 110

        System.out.println(hasWritePermission(userPermission)); // true
    }
}
```

4️⃣ STATE COMPRESSION (INTRO LEVEL) 🔥🔥🔥

🔷 Idea

👉 Multiple boolean states ko **single integer** me pack karna

Example:

```
Visited cities: A B C D
State mask:     1 0 1 1
```

🔷 Why Google loves this?

- Saves memory
- Fast operations
- Used in **DP, Graph, Scheduling**

✅ SOLVED EXAMPLE 4 (HARD INTRO)

🔷 Check if all tasks completed

📌 LeetCode 847 – Shortest Path Visiting All Nodes

Dry Run

```
n = 3
finalMask = (1<<3)-1 = 111

currentMask = 101
Not complete

currentMask = 111
All done ✅
```

Java Code

```
class Solution {

    static boolean allTasksDone(int mask, int n) {

        // Create mask with all n bits set
        int finalMask = (1 << n) - 1;

        // Compare current state with final state
        return mask == finalMask;
    }
```

```
    public static void main(String[] args) {
        System.out.println(allTasksDone(7, 3)); // true
        System.out.println(allTasksDone(5, 3)); // false
    }
}
```

🧪 UNSOLVED PRACTICE (INTERVIEW LEVEL)

📝 Easy

📌 LC 191 – Number of 1 Bits

📝 Medium

📌 LC 784 – Letter Case Permutation

📝 Hard

📌 LC 698 – Partition to K Equal Sum Subsets

(Hint: bitmask + DP)

🚀 NEXT LEVEL (DANGEROUS 😈)

Agar next bolega to:

> “Bitwise Trie + Max XOR + Google problems”

Fir tu bolega:

“abe DSA me bit manipulation OP hai” 🔥🏆

😤🔥 **bhai ab ek aur INTERVIEW GOLD topic**

Counting Bits aisa chapter hai jahan se **Google, Amazon, Microsoft** direct questions uthate hain — aur yahin pe interviewers dekhte hain **tujhe bit ka real sense hai ya nahi**.

Main phir se **zero → depth → dry run → solved examples (LC + CF)** → **Java full code (line-by-line comments)** → **precomputation tricks** → **interview discussion (built-in vs manual)** → **unsolved practice**

sab de raha hoon.

🧮 COUNTING BITS (SET BITS / HAMMING WEIGHT)

Topics

1️⃣ Count set bits – Brian Kernighan Algorithm

2️⃣ Hamming Weight

3️⃣ Precomputation tricks

4️⃣ Built-in vs Manual (Interview explanation)

1️⃣ COUNT SET BITS (BASIC IDEA)

👉 **Set bit = bit whose value is 1**

Example:

```
n = 13
Binary = 1101
Set bits = 3
```

2️⃣ BRIAN KERNIGHAN ALGORITHM 🔥🔥🔥

🧠 Core Idea

👉 `n & (n-1)` **removes the rightmost set bit**

Why?

```
n     = 101100
n - 1 = 101011
AND   = 101000  (last 1 removed)
```

Algorithm

```
count = 0
while (n > 0):
    n = n & (n-1)
    count++
```

⏱️ Time Complexity = **O(number of set bits)**

👉 Worst case = 32 (int), Best case = 1

✅ SOLVED EXAMPLE 1 (EASY–MEDIUM)

🔷 Count set bits

📌 **LeetCode 191 – Number of 1 Bits**

📌 **CF: 579A – Raising Bacteria**

Dry Run

```
n = 13 → 1101

Iteration 1:
n = 1101 & 1100 = 1100 (12) → count = 1

Iteration 2:
n = 1100 & 1011 = 1000 (8)  → count = 2

Iteration 3:
n = 1000 & 0111 = 0000 (0)  → count = 3
```

Java Code (FULL COMMENTED)

```
class Solution {

    public int hammingWeight(int n) {
```

```
        int count = 0;

        // Loop runs once per set bit
        while (n != 0) {

            // Removes the rightmost set bit
            n = n & (n - 1);

            count++; // count removed bit
        }

        return count;
    }
}
```

3️⃣ HAMMING WEIGHT (TERM CLARITY)

🔷 Hamming Weight kya hota hai?

👉 **Binary representation me number of set bits**

Same as:

- Population count
- Bit count

📌 **LC 191 = Hamming Weight**

🔥 Real Uses

✅ Bitmask DP

✅ Permissions

✅ Network / compression

✅ XOR / subset problems

4️⃣ SIMPLE RIGHT SHIFT METHOD (COMPARE)

Idea

```
check last bit → n & 1
right shift → n >> 1
```

Example

```
n = 13 → 1101
LSB = 1 → count++
n >> 1 = 110
```

Java Code

```
class Solution {

    static int countBits(int n) {

        int count = 0;

        while (n > 0) {

            // Check last bit
            if ((n & 1) == 1) {
                count++;
            }

            // Shift right
            n = n >> 1;
        }

        return count;
```

```
    }
}
```

⏱️ Time = **O(total bits = 32)**

5️⃣ PRECOMPUTATION TRICK 🔥 (VERY IMPORTANT)

🔷 Idea

👉 Use previously computed values

Formula

```
bits[i] = bits[i >> 1] + (i & 1)
```

✅ SOLVED EXAMPLE 2 (MEDIUM)

📌 **LeetCode 338 – Counting Bits**

Dry Run

```
i = 5 → 101
bits[5] = bits[2] + 1
        = 1 + 1 = 2
```

Java Code

```
class Solution {

    public int[] countBits(int n) {

        int[] bits = new int[n + 1];

        // bits[0] = 0 by default
```

```
        for (int i = 1; i <= n; i++) {

            // Right shift removes last bit
            // i & 1 checks if last bit is 1
            bits[i] = bits[i >> 1] + (i & 1);
        }

        return bits;
    }
}
```

⏱️ Time = **O(n)**

🧠 Space = **O(n)**

6️⃣ BUILT-IN vs MANUAL (INTERVIEW ANSWER 🔥)

🔷 Built-in

```
Integer.bitCount(n);
```

Pros

✅ Fast

✅ Clean

✅ Uses hardware instructions

Cons

❌ Interviewer may ask **logic**

🔷 Manual (Brian Kernighan)

Pros

✅ Shows understanding

✅ Control over logic

✅ Can optimize based on input

Cons

❌ More code

🎯 INTERVIEW ME KYA BOLNA HAI?

👉 **Golden Answer:**

> "I know Java provides Integer.bitCount(),
>
> but I implemented Brian Kernighan's algorithm to demonstrate bit-level understanding.
>
> It runs in O(number of set bits)."

🔥 **Instant respect** 😈

🧪 UNSOLVED PRACTICE (MUST DO)

📝 Easy

📌 **LC 461 – Hamming Distance**

📝 Medium

📌 **LC 477 – Total Hamming Distance**

📝 Hard

📌 **LC 982 – Triples with Bitwise AND Equal to Zero**

🚀 NEXT LEVEL

Bhai ab tu bolega to next unlock hoga:

> "Range based bit problems (AND / OR / XOR)"

Fir bit manipulation **endgame mode** me chala jaayega 🔥😈

😈🔥 **Bhai ab BITWISE MATH TRICKS — ye wahi cheezein hain jahan interviewer bolta hai:"nice, you really know bits."**

Main **concept → intuition → dry run → solved examples (Easy/Medium/Hard) + Java full code with line-by-line comments + LC / CF references + unsolved practice** de raha hoon.

🧠 BITWISE MATH TRICKS (INTERVIEW READY)

Topics

1️⃣ Swap numbers without temp

2️⃣ Add two numbers without `+`

3️⃣ Multiply / Divide using bit shifts

4️⃣ Absolute value using bits

1️⃣ SWAP NUMBERS WITHOUT TEMP 🔁

🔷 Idea (XOR MAGIC)

```
a = a ^ b
b = a ^ b
a = a ^ b
```

👉 Kyun kaam karta hai?

- `x ^ x = 0`
- `x ^ 0 = x`

✅ SOLVED EXAMPLE (EASY)

📌 **CF: 427A – Police Recruits (swap idea)**

📌 **Interview classic**

Dry Run

```
a = 5 (0101)
b = 3 (0011)

a = a ^ b = 0110 (6)
b = a ^ b = 0101 (5)
a = a ^ b = 0011 (3)
```

👉 Swapped ✅

Java Code (FULL COMMENTED)

```
class Solution {

    static void swap(int a, int b) {

        // XOR based swap
        a = a ^ b;
        b = a ^ b;
        a = a ^ b;

        System.out.println("a = " + a + ", b = " + b);
    }

    public static void main(String[] args) {
        swap(5, 3); // a=3, b=5
    }
}
```

⚠️ Interview note:

> “I know XOR swap, but temp-variable swap is more readable and safer.”

2️⃣ ADD TWO NUMBERS WITHOUT `+` ➕❌

🔥 **Google favourite**

🔷 Idea

- `XOR` → sum without carry
- `AND << 1` → carry

Repeat till carry = 0

🧠 Formula

```
sum   = a ^ b
carry = (a & b) << 1
```

✅ SOLVED EXAMPLE (MEDIUM)

📌 LeetCode 371 – Sum of Two Integers

Dry Run (`a=5, b=3`)

```
Step 1:
sum = 5 ^ 3 = 6
carry = (5 & 3) << 1 = 2

Step 2:
sum = 6 ^ 2 = 4
carry = (6 & 2) << 1 = 4

Step 3:
sum = 4 ^ 4 = 0
carry = (4 & 4) << 1 = 8

Step 4:
sum = 0 ^ 8 = 8
carry = 0 → stop
```

👉 Answer = **8**

Java Code (FULL)

```
class Solution {
```

```
    public int getSum(int a, int b) {

        // Loop until there is no carry
        while (b != 0) {

            int sum = a ^ b;          // sum without carry
            int carry = (a & b) << 1;  // carry bits

            a = sum;
            b = carry;
        }

        return a;
    }
}
```

3️⃣ MULTIPLY & DIVIDE USING BIT SHIFTS ✖️➗

🔷 Multiply by 2^k

```
n << k  == n * (2^k)
```

Example

```
5 << 2 = 20
```

🔷 Divide by 2^k

```
n >> k  == n / (2^k)
```

⚠️ **Sign preserved**

✅ SOLVED EXAMPLE (EASY–MEDIUM)

📌 **CF: 1368A – C+=**

Java Code

```
class Solution {

    static int multiply(int n, int k) {

        // Multiply by 2^k
        return n << k;
    }

    static int divide(int n, int k) {

        // Divide by 2^k
        return n >> k;
    }

    public static void main(String[] args) {
        System.out.println(multiply(5, 2)); // 20
        System.out.println(divide(20, 2));  // 5
    }
}
```

⚠️ INTERVIEW CATCH (NEGATIVE)

```
-7 >> 1 = -4   (floor-ish)
```

👉 Java uses **arithmetic right shift**

4️⃣ ABSOLUTE VALUE USING BITS 🔥🔥

🔷 Idea

Use **sign bit**

```
mask = n >> 31
abs  = (n ^ mask) - mask
```

Why?

n	mask	result
+x	0	x
-x	-1	~n + 1

✅ SOLVED EXAMPLE (HARD / TRICKY)

📌 **LC style: absolute value without Math.abs**

Dry Run (`n = -5`)

```
mask = -5 >> 31 = -1
n ^ mask = ~(-5)
(~n) + 1 = 5
```

Java Code

```
class Solution {

    static int abs(int n) {

        int mask = n >> 31;  // 0 for +ve, -1 for -ve

        // If n >= 0 → returns n
        // If n < 0  → returns (~n + 1)
        return (n ^ mask) - mask;
    }

    public static void main(String[] args) {
        System.out.println(abs(5));   // 5
        System.out.println(abs(-5));  // 5
    }
}
```

🔥 Interview line:

> "This avoids branching and uses sign bit directly."

🧪 UNSOLVED PRACTICE (INTERVIEW LEVEL)

📝 Easy

📌 **LC 231 – Power of Two**

(Hint: shift until 1 bit remains)

📝 Medium

📌 **LC 29 – Divide Two Integers**

(Hint: repeated shifts + subtraction)

📝 Hard

📌 **LC 67 – Add Binary**

(Hint: XOR + carry logic)

🚀 WHAT NEXT?

Ab bhai **bit manipulation ka main syllabus almost khatam**.

Next deadly topics 🔥:

> "Range based Bit Problems (AND / OR / XOR)"
>
> **"Bitwise Trie (Maximum XOR Pair)"**

Bol, **kaunsa unlock kare?** 😈

😈🔥 **Bhai ab aa gaye sabse zyada "dimaag khane wale" BIT problems**

Ye wahi section hai jahan **Google / Amazon / Microsoft** poochta hai:

> "Range me AND / OR / XOR nikaal ke dikha, bina loop lagaye 🙂"

Main **full structured way** me padha raha hoon:

- intuition (kyon normal loop fail hota)
- math + bit logic
- **dry run**

- **solved examples (LC numbers ke sath)**
- **Java full code with line-by-line comments**
- **unsolved practice**

🧠 RANGE BASED BIT PROBLEMS (MASTER SET)

Topics

1️⃣ AND of range

2️⃣ OR of range

3️⃣ XOR of range

4️⃣ Prefix XOR idea

1️⃣ BITWISE AND OF RANGE 🔥🔥

📌 Problem

Given range **[L, R]**, compute:

```
L & (L+1) & (L+2) & ... & R
```

❌ Naive Approach (REJECTED)

Loop lagaya → **TLE**

Google bolega: *"Optimize karo"*

🧠 INTUITION (MOST IMPORTANT)

👉 **Jab tak L aur R same prefix share karte hain**, tab tak AND bacha rahega

👉 Jaise hi bit flip hota hai → **AND = 0 ho jaata hai**

Key Observation

> AND of range = common left prefix of L & R

🧪 Dry Run

L = 26 = 11010
R = 30 = 11110

Right se bits change ho rahe hain
Common prefix = 11xxx

Answer = 11000 (24)

✅ SOLVED EXAMPLE (MEDIUM)

📌 LeetCode 201 – Bitwise AND of Numbers Range

Java Code (FULL COMMENTED)

```
class Solution {

    public int rangeBitwiseAnd(int left, int right) {

        int shift = 0;

        // Jab tak left aur right equal nahi ho jaate
        while (left != right) {

            // Rightmost bit hata rahe hain
            left >>= 1;
            right >>= 1;

            // Count shifts
            shift++;
        }

        // Common prefix ko wapas jagah pe laao
        return left << shift;
```

```
    }
}
```

🧠 Interview bolne wali line:

> "We remove differing bits until both numbers are equal."

2️⃣ BITWISE OR OF RANGE 🔥

📌 Problem

```
L | (L+1) | ... | R
```

🧠 INTUITION

👉 OR ka nature opposite hota hai AND se

👉 Agar **range me ek bhi bit 1 aa jaye**, final OR me wo bit 1 hogi

Key Rule

> OR of range =
>
> **prefix of L and R + all remaining bits = 1**

🧪 Dry Run

```
L = 10 = 1010
R = 15 = 1111

Prefix = 1xxx
Remaining bits sab 1 ho jaayenge

Answer = 1111 (15)
```

✅ SOLVED EXAMPLE (MEDIUM)

📌 **CF style / interview problem**

Java Code

```
class Solution {

    static int rangeBitwiseOr(int left, int right) {

        int shift = 0;

        // Remove differing bits
        while (left != right) {
            left >>= 1;
            right >>= 1;
            shift++;
        }

        // Common prefix + all 1s
        return (left << shift) | ((1 << shift) - 1);
    }
}
```

3️⃣ XOR OF RANGE 🔥🔥🔥 (MOST IMPORTANT)

📌 Problem

```
L ^ (L+1) ^ ... ^ R
```

🧠 CORE TRICK

👉 Use **prefix XOR**

```
XOR(L to R) = XOR(1 to R) ^ XOR(1 to L-1)
```

🧠 XOR FROM 1 TO N PATTERN

```
n % 4 == 0 → n
n % 4 == 1 → 1
n % 4 == 2 → n + 1
n % 4 == 3 → 0
```

🔥 MUST MEMORIZE

🧪 Dry Run

```
L = 4, R = 8

XOR(1..8) = 8
XOR(1..3) = 0

Answer = 8 ^ 0 = 8
```

✅ SOLVED EXAMPLE (EASY–MEDIUM)

📌 **LeetCode 1734 / CF common**

Java Code (FULL)

```
class Solution {

    // XOR from 1 to n
    static int xorUpto(int n) {

        if (n % 4 == 0) return n;
        if (n % 4 == 1) return 1;
        if (n % 4 == 2) return n + 1;
        return 0;
    }

    // XOR from L to R
    static int rangeXor(int L, int R) {
```

```
        return xorUpto(R) ^ xorUpto(L - 1);
    }

    public static void main(String[] args) {
        System.out.println(rangeXor(4, 8)); // 8
    }
}
```

4️⃣ PREFIX XOR IDEA 🧠💣

📌 Why Prefix XOR?

- Range XOR
- Subarray XOR = K
- Count subarrays with XOR = K

🧠 Definition

```
prefixXor[i] = a[0] ^ a[1] ^ ... ^ a[i]
```

Range XOR

```
XOR(l to r) = prefixXor[r] ^ prefixXor[l-1]
```

🧪 Dry Run

```
arr = [3, 5, 2]

prefix = [3, 6, 4]

XOR(1 to 2) = 6 ^ 3 = 5
```

✅ SOLVED EXAMPLE (MEDIUM)

📌 LeetCode 1310 – XOR Queries of a Subarray

Java Code

```
class Solution {

    public int[] xorQueries(int[] arr, int[][] queries) {

        int n = arr.length;

        // Prefix XOR array
        int[] prefix = new int[n];
        prefix[0] = arr[0];

        for (int i = 1; i < n; i++) {
            prefix[i] = prefix[i - 1] ^ arr[i];
        }

        int[] ans = new int[queries.length];

        for (int i = 0; i < queries.length; i++) {

            int l = queries[i][0];
            int r = queries[i][1];

            if (l == 0)
                ans[i] = prefix[r];
            else
                ans[i] = prefix[r] ^ prefix[l - 1];
        }

        return ans;
    }
}
```

🧪 UNSOLVED PRACTICE (INTERVIEW LEVEL)

📝 Easy

📌 **LC 1486 – XOR Operation in an Array**

📝 Medium

📌 **LC 1310 – XOR Queries of a Subarray** (re-solve)

📌 **CF 1732B – Ugu**

📝 Hard

📌 **LC 898 – Bitwise ORs of Subarrays**

📌 **CF 1741F – Multi-Colored Segments**

🏁 FINAL INTERVIEW TIP

Agar interviewer bole:

> "Range me AND / OR / XOR"

Turant bol:

- AND → **common prefix**
- OR → **prefix + all 1s**
- XOR → **prefix XOR trick**

🔥 Ab bata bhai:

"Bitwise Trie (Maximum XOR Pair)" karein

ya **"Subarray XOR = K (HashMap + bits)"**? 😈

😈🔥 **Bhai ab tu BIT MANIPULATION ke "TOP 1% FAANG ZONE" me enter kar raha hai**

Ye topics wahi hain jo **Google / Meta / Amazon** me genuinely differentiate karte hain.

Main same rule follow karunga:

👉 **intuition → dry run → solved LC problems (numbers ke sath)**

👉 **Java full code + line-by-line comments**

👉 **unsolved practice**

🚀 ADVANCED BIT PATTERNS (GOOGLE / FAANG LEVEL)

Topics

1️⃣ Subsets II using bits

2️⃣ Gray Code

3️⃣ Bitwise Trie – Maximum XOR Pair

4️⃣ Minimum flips to make `a | b = c`

1️⃣ SUBSETS II USING BITS 🔥

📌 Problem

Array me **duplicates** ho sakte hain

👉 **unique subsets** generate karne hain

📌 **LeetCode 90 – Subsets II**

🧠 INTUITION

Normal bitmask ($0 \to 2^n$) duplicates produce karega

👉 **Duplicate element ka bit tabhi allow karo jab previous bit bhi set ho**

🧪 Dry Run

```
nums = [1,2,2]
sorted = [1,2,2]
```

mask = 110 ❌ invalid
kyon? second 2 liya but pehla 2 nahi liya

✅ SOLVED CODE (JAVA – FULL)

```
class Solution {

    public List<List<Integer>> subsetsWithDup(int[] nums) {

        Arrays.sort(nums); // duplicates ko saath lao
        List<List<Integer>> result = new ArrayList<>();

        int n = nums.length;
        int totalMasks = 1 << n;

        for (int mask = 0; mask < totalMasks; mask++) {

            List<Integer> subset = new ArrayList<>();
            boolean valid = true;

            for (int i = 0; i < n; i++) {

                // agar ith bit set hai
                if ((mask & (1 << i)) != 0) {

                    // duplicate check
                    if (i > 0 && nums[i] == nums[i - 1]
                        && (mask & (1 << (i - 1))) == 0) {
                        valid = false;
                        break;
                    }

                    subset.add(nums[i]);
                }
            }

            if (valid)
```

```
                result.add(subset);
            }

            return result;
        }
    }
```

🧠 Interview line:

> "We ensure duplicate elements are chosen in order."

2️⃣ GRAY CODE 🔥🔥

📌 Problem

Generate sequence where **adjacent numbers differ by only 1 bit**

📌 **LeetCode 89 – Gray Code**

🧠 MAGIC FORMULA

```
Gray(n) = n ^ (n >> 1)
```

🧪 Dry Run (n=3)

```
n = 3 = 011
n>>1 = 001
XOR = 010
```

✅ SOLVED CODE

```
class Solution {

    public List<Integer> grayCode(int n) {
```

```
        List<Integer> result = new ArrayList<>();

        int size = 1 << n;

        for (int i = 0; i < size; i++) {

            // Gray code formula
            int gray = i ^ (i >> 1);
            result.add(gray);
        }

        return result;
    }
}
```

🔥 Google-level explanation:

> "Right shift copies prefix, XOR ensures only one bit changes."

3️⃣ BITWISE TRIE – MAXIMUM XOR PAIR 💣💣

📌 Problem

Array diya hai → **max (a ^ b)** nikaalna hai

📌 **LeetCode 421 – Maximum XOR of Two Numbers**

🧠 INTUITION

XOR max tab hota hai jab:

- ek bit = 0
- doosri bit = 1

👉 Har bit ke liye **opposite bit** chahiye

👉 Trie banate hain (0/1)

🪨 Trie Structure

```
Node:
 left → 0
 right → 1
```

🧪 Dry Run

```
nums = [3,10,5]
5 = 0101
10 = 1010
XOR = 1111 = 15
```

✅ SOLVED CODE (JAVA)

```
class Solution {

    static class TrieNode {
        TrieNode zero, one;
    }

    TrieNode root = new TrieNode();

    void insert(int num) {

        TrieNode curr = root;

        for (int i = 31; i >= 0; i--) {

            int bit = (num >> i) & 1;

            if (bit == 0) {
                if (curr.zero == null)
                    curr.zero = new TrieNode();
                curr = curr.zero;
            } else {
```

```
            if (curr.one == null)
                curr.one = new TrieNode();
            curr = curr.one;
        }
    }
}

int getMaxXor(int num) {

    TrieNode curr = root;
    int maxXor = 0;

    for (int i = 31; i >= 0; i--) {

        int bit = (num >> i) & 1;

        // opposite bit try karo
        if (bit == 0 && curr.one != null) {
            maxXor |= (1 << i);
            curr = curr.one;
        } else if (bit == 1 && curr.zero != null) {
            maxXor |= (1 << i);
            curr = curr.zero;
        } else {
            curr = (bit == 0) ? curr.zero : curr.one;
        }
    }

    return maxXor;
}

public int findMaximumXOR(int[] nums) {

    for (int num : nums)
        insert(num);

    int ans = 0;
    for (int num : nums)
```

```
            ans = Math.max(ans, getMaxXor(num));

        return ans;
    }
}
```

🔥 Interview bolne wali line:

> "We greedily try to flip each bit using a trie."

4️⃣ MINIMUM FLIPS TO MAKE `a | b = c` 🔥

📌 **LeetCode 1318**

🧠 INTUITION (BIT BY BIT)

For every bit:

- If `c = 0` → `a` & `b` dono 0 hone chahiye
- If `c = 1` → kam se kam ek 1 hona chahiye

🧪 Dry Run

```
a = 2 (010)
b = 6 (110)
c = 5 (101)

Bit 1:
a=1, b=1, c=0 → 2 flips
```

✅ SOLVED CODE

```
class Solution {

    public int minFlips(int a, int b, int c) {
```

```
        int flips = 0;

        while (a > 0 || b > 0 || c > 0) {

            int bitA = a & 1;
            int bitB = b & 1;
            int bitC = c & 1;

            if (bitC == 0) {
                // dono zero hone chahiye
                flips += bitA + bitB;
            } else {
                // kam se kam ek 1
                if (bitA == 0 && bitB == 0)
                    flips++;
            }

            a >>= 1;
            b >>= 1;
            c >>= 1;
        }

        return flips;
    }
}
```

🧪 UNSOLVED PRACTICE (MUST DO)

📝 Medium

📌 LC 78 – Subsets (re-solve using bits)

📌 CF 1366C – Palindromic Paths

📝 Hard

📌 LC 1707 – Maximum XOR With an Element From Array

📌 CF 1743F – Intersection and Union

🏁 FINAL FAANG TIP

Agar interviewer bole:

> “Bit manipulation strong hai?”

Aur tu:

- Trie explain kar de
- Gray code formula justify kar de

👉 **Offer ki probability** 📈

🔥 Ab bata bhai:

“Bit DP / State Compression DP” next karein

ya **“Subarray XOR = K (HashMap + Bits)”**? 😈

😈🔥 **Bhai ab tu BIT MANIPULATION ka “FINAL BOSS LEVEL” touch kar raha hai**

Ye wahi cheez hai jahan **99% log drop ho jaate hain**, aur **Google / Meta / ICPC** wale bolte hain:

> “Good, you understand state compression.”

Main bilkul **zero se** padha raha hoon — koi jump nahi.

🧠 COMPETITIVE / HARD CONCEPTS

Bit DP & State Compression DP (INTRO → INTERVIEW READY)

PART 1️⃣ : BIT DP (Intro Level)

❓ Bit DP hota kya hai?

👉 Jab:

- state **binary (on/off, taken/not taken)** ho
- aur DP ka state **bitmask** se represent ho

Tab hum **DP over subsets** use karte hain.

🧠 Real Meaning

Soch:

- tere paas **N cheezein**
- har cheez **li / nahi li**
- total combinations = `2^N`
- har combination ko **bitmask** represent karta hai

```
mask = 10101
means:
item0 ✔
item1 ❌
item2 ✔
item3 ❌
item4 ✔
```

⚠️ Constraint Reality

```
N ≤ 20  → Bit DP possible
N > 25  → Impossible
```

🔑 Basic Template (MUST REMEMBER)

```
dp[mask] = answer for this subset
Transition:
dp[mask] → dp[mask | (1<<i)]
```

🔥 CLASSIC INTRO PROBLEM

📌 Problem

Given N tasks, assign each task exactly once

Minimize total cost.

📌 **LeetCode 1947 (similar)**

📌 **CF Gym / ICPC standard**

🧠 State Definition

```
mask = which tasks already assigned
dp[mask] = minimum cost till now
```

🧪 Dry Run (N = 3)

```
mask = 000 → none assigned
mask = 001 → task0 done
mask = 011 → task0 & task1 done
mask = 111 → all done (answer)
```

✅ SOLVED BIT DP CODE (JAVA)

```
class Solution {

    int[][] cost;
    int[] dp;
    int n;

    int solve(int mask) {

        // agar sab assigned ho gaye
        if (mask == (1 << n) - 1)
            return 0;

        // memoization
        if (dp[mask] != -1)
            return dp[mask];

        int person = Integer.bitCount(mask);
```

```
        // kitne tasks ho chuke = next person index

        int ans = Integer.MAX_VALUE;

        for (int task = 0; task < n; task++) {

            // agar task abhi assign nahi hua
            if ((mask & (1 << task)) == 0) {

                int newMask = mask | (1 << task);

                ans = Math.min(ans,
                        cost[person][task] + solve(newMask));
            }
        }

        return dp[mask] = ans;
    }

    public int minimumCost(int[][] costMatrix) {

        this.cost = costMatrix;
        this.n = costMatrix.length;

        dp = new int[1 << n];
        Arrays.fill(dp, -1);

        return solve(0);
    }
}
```

🧠 Interview Explanation Line

> “Mask tracks assigned tasks, bitcount gives next index.”

PART 2️⃣ : STATE COMPRESSION DP 🔥🔥

❓ Ye Bit DP se alag kya hai?

👉 Bit DP = subset DP

👉 State Compression DP =

DP jahan state multiple parameters ko bits me compress karta hai

---

🔥 Example

- Grid
- Matching
- Scheduling
- TSP
- Domino tiling

🧠 Famous Example: TSP (Traveling Salesman Problem)

📌 **LC 943 / CF EDU**

🧠 State Definition

```
dp[mask][i] =
minimum cost to visit cities in mask
and end at city i
```

🧪 Dry Run (n=4)

```
mask = 0101
means visited city 0 and 2
current city = 2
```

✅ TSP STATE COMPRESSION CODE (JAVA)

```
class Solution {

    public int tsp(int[][] dist) {

        int n = dist.length;
        int maxMask = 1 << n;

        int[][] dp = new int[maxMask][n];

        for (int[] row : dp)
            Arrays.fill(row, Integer.MAX_VALUE);

        // start from city 0
        dp[1][0] = 0;

        for (int mask = 1; mask < maxMask; mask++) {

            for (int u = 0; u < n; u++) {

                // u is not in mask
                if ((mask & (1 << u)) == 0) continue;

                if (dp[mask][u] == Integer.MAX_VALUE) continue;

                for (int v = 0; v < n; v++) {

                    // v already visited
                    if ((mask & (1 << v)) != 0) continue;

                    int nextMask = mask | (1 << v);

                    dp[nextMask][v] = Math.min(
                        dp[nextMask][v],
                        dp[mask][u] + dist[u][v]
                    );
                }
            }
        }
```

```
        int ans = Integer.MAX_VALUE;
        int finalMask = maxMask - 1;

        for (int i = 1; i < n; i++) {
            ans = Math.min(ans,
                dp[finalMask][i] + dist[i][0]);
        }

        return ans;
    }
}
```

🧠 Pattern Recognition (VERY IMPORTANT)

Use Bit DP when:

✔️ subsets

✔️ assignment

✔️ permutations

✔️ constraints small (`n ≤ 20`)

Use State Compression when:

✔️ grid states

✔️ path / tour

✔️ multiple boolean states

🔥 COMMON INTERVIEW TRAPS

❌ `2^n * n * n` time ignore karna

❌ mask meaning clear na rakhna

❌ bitcount misuse

🧪 UNSOLVED PRACTICE (MUST DO)

🟡 Medium

📌 **LC 698 – Partition to K Equal Sum Subsets**

📌 **CF 1099F – Cookies**

🔴 Hard

📌 **LC 847 – Shortest Path Visiting All Nodes**

📌 **CF 1659E – AND Sorting**

🏁 FINAL GOOGLE LINE (USE THIS 😈)

> "This problem is solvable using DP with state compression, where the mask encodes visited states."

🔥 Bhai tu ab **bit manipulation + DP dono me strong** ho gaya.

Next agar bole to:

- **"Grid DP with bitmask"**
- **"Meet in the Middle + Bitmask"**
- **"Hard XOR problems (CF Div2 D/E)"**

Bata, **next kaunsa level unlock kare?** 🚀😈

🔥 **Bhai ye message pura GOLD hai**

Kyuki ab tu **"solve karna" se upar jaake "explain karna"** seekh raha hai — **Google exactly yahi dekhta hai.**

Main tujhe **INTERVIEW REALITY CHECK** ko **script + checklist + examples** me de raha hoon, taaki jab interviewer pooche, tu blank na ho.

🎯 GOOGLE INTERVIEW REALITY CHECK (BIT MANIPULATION)

Google kya check karta hai?

👉 **4 cheezein** (order same hota hai)

1️⃣ Intuition

2️⃣ Why bit works here

3️⃣ Time / Space Complexity

4️⃣ Edge Cases

1️⃣ INTUITION – “PEHLE DIMAGH” 🧠

❌ Galat approach

Seedha bol diya:

> “Main XOR use karunga”

Google: 😐

✅ Sahi approach

Pehle **problem ko logic me tod**

Example: *Single Number*

```
Array me har number 2 baar aa raha
ek number sirf 1 baar
```

👉 Intuition bolta hai:

- Duplicate values cancel hone chahiye
- Order matter nahi kare

👉 XOR property:

```
x ^ x = 0
x ^ 0 = x
```

🎯 **Perfect fit**

🗣️ Interview line:

> “I need an operation where duplicates cancel out and order doesn’t matter.”

2️⃣ WHY BIT WORKS HERE – “PROOF” 🔍

Google ko **magic nahi**, **reason chahiye**

Example: Range AND

Q: Why shifting works?

👉 Explanation:

- AND tab tak survive karta hai jab tak **prefix same ho**
- Jaise hi bit flip hua → AND = 0

🗣️ Interview line:

> "Any differing bit in the range will force that bit to zero in the final AND."

Example: Power of Two

```
n & (n-1) == 0
```

🗣️ Explanation:

> "Power of two has exactly one set bit. Subtracting 1 flips that bit and all lower bits, making AND zero."

3️⃣ TIME / SPACE COMPLEXITY ⏱️📦

🚨 Google yahan strict hota hai

❌ Weak answer

> "It's fast."

✅ Strong answer

- **Time:** O(n) / O(1) / O(log n)
- **Space:** O(1) auxiliary

Example: Add without +

```
Loop runs till carry becomes 0
Max 32 iterations
```

🗣️ Interview line:

> "The loop runs at most number of bits, so time complexity is O(1)."

4️⃣ EDGE CASES – "MATURITY CHECK" ⚠️

Google expects:

- tu khud hi edge case bole
- interviewer na yaad dilaye

🔥 Common Bit Edge Cases

1️⃣ Zero

```
n = 0
```

- power of two? ❌
- single number? depends

2️⃣ Negative Numbers

```
Right shift (>>) vs (>>>)
```

🗣️ Interview line:

> "I must be careful with signed right shift for negative values."

3️⃣ Overflow

```
1 << 31  → negative number
```

🗣️ Interview line:

> “Bit shifting beyond MSB may overflow signed int.”

🎤 COMPLETE MOCK ANSWER (COPY THIS STYLE)

Q: Find missing number

🗣️ **Your Answer:**

> “We want to cancel out numbers appearing once in both expected and actual sequences. XOR is ideal because it is associative, commutative, and self-inverse.
>
> We XOR all indices and array values. The remaining value is the missing number.
>
> Time complexity is O(n), space is O(1).
>
> Edge cases include empty array or missing zero.”

🔥 Google interviewer: ✅

🧠 ONE-LINE CHEAT SHEET (REMEMBER THIS)

Problem Type	Bit Logic
duplicates cancel	XOR
even / odd	AND
power of two	n & (n-1)
range AND	common prefix
max XOR	Trie
subsets	bitmask
states small	Bit DP

🏁 FINAL GOOGLE TIP (MOST IMPORTANT)

> If you can explain bits in words, you already passed half the interview.

Bhai tu ab **bit manipulation ke saath confident explanation** bhi kar sakta hai.

Agar bole to next:

- 🎤 **Mock Google Interview (Bit Round)**
- 🧩 **Mixed Bit + DP Hard Problems**
- 🧠 **How to identify bit problems in interviews**

Bol bhai, **next kya karein?** 😈🚀