

Number System

Bit Manipulation

① AND :-

a	b	$a \& b$
0	0	0
0	1	0
1	0	0
1	1	1

★ When you 'and' any number with 1.
The digits remain the same.

Eg:-

$$\begin{array}{r} 11001 \\ \otimes 11111 \\ \hline 11001 \end{array}$$

Same.

② OR:-

a	b	$a \text{ OR } b$
0	0	0
0	1	1
1	0	1
1	1	1

③ XOR (^) (if and only if)

X

Exclusive OR :-

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	1
1	1	0

OBSERVATIONS :- X

①. $a \wedge 1 = \bar{a}$

②. $a \wedge 0 = a$

③. $a \wedge a = 0$

④. Complement (\sim)

$$a = 10110$$

$$\bar{a} = 01001$$

Number systems

①. Decimal \rightarrow 0, 1, 2, ..., 9 Base 10

$$(357)_{10}, \quad (10)_{10}$$

②. Binary \rightarrow 0 & 1 Base 2

$$(10)_{10} = (1010)_2$$

$$(7)_{10} = (111)_2$$

③. Octal \rightarrow 0, 1, 2, 3, ..., 7 Base 8

④. Hexadecimal \rightarrow 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

COMPARISON :-

DECIMAL :- 0 1 2 3 4 5 6 7 (8) 9 10 11

OCTAL :- 0 1 2 3 4 5 6 7 (10) 11 12 13

So, here 8 in decimal means
10 in octal.

$$(9)_{10} = (11)_8$$

HEXADECIMAL :- $(10)_{10} = (A)_{16}$

$$(10)_8 = (8)_{16}$$

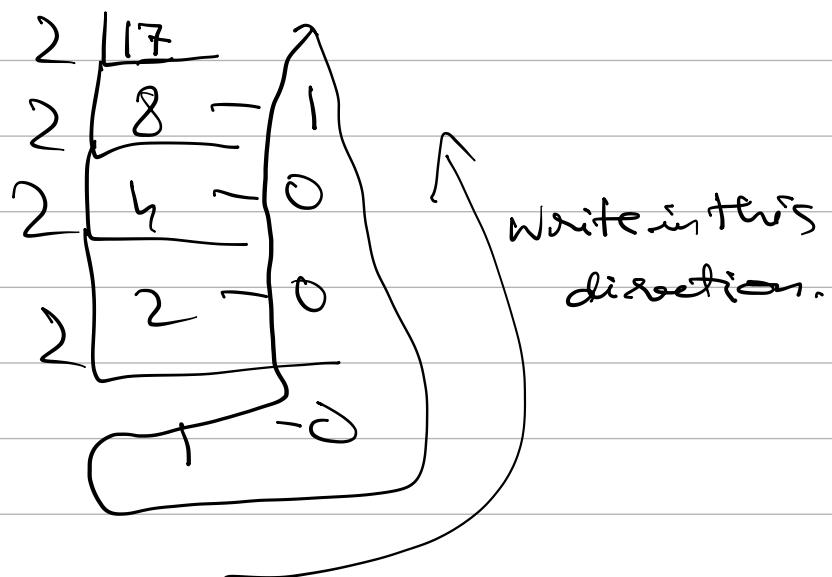
Conversions

- ① Decimal to Base 6.
- ② base 6 to Decimal.

- ① Decimal to base 6 :-

Q). Convert $(17)_{10}$ to base 2.

Keep dividing by base, take remainders, write in opposite.



$$(100001)_2 = (17)_{10}$$

$$Q). \quad (17)_{10} = (?)_8.$$

$$\begin{array}{r} 8 \\ \times 17 \\ \hline \end{array}$$

Write in opposite.

$$(17)_{10} = (21)_8.$$

② Convert any base to decimal

$$Q). (10001)_2 = (?)_{10}.$$

Steps :- Multiple and add the power of base with digits.

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

$$1) \quad 16 + 0 + 0 + 0 + 1 = \binom{17}{0}$$

$$Q). \quad (21)_8 = (?)_{10}.$$

$$= 8 \times 2 + 8^{\circ} \times 1 = (17)_{10}$$

NOTE :-

If the question is to convert

base a number to base b number.

*). First convert base a to decimal.

*). Next convert decimal to base b.

(5) - Left shift operator ($<<$) (^{used to double a number})

$$\text{Step-1:- } (10)_{10} = (1010)_2$$

convert any number to binary first.

$$\text{Step-2 :- } 1010 \ll 1 = 10100$$

$$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 \\ = \underline{\underline{20}}$$

$$\therefore a \ll 1 = 2a$$

General Point :-

$\ddot{\times}$

$$a \ll b = a \times 2^b$$

6]. Right Shift :- >>

Q). 0011001 >> 1

Solu:- \Rightarrow 001100

GENERAL POINT:-

$$a \gg b = a \times \frac{1}{2^b}$$

Questions

1]. Given a number n. Find whether it is odd or even.

Solu:-

Logic :- If 2^0 place is - 1 then, number is odd else it is even.

\therefore If $n \& 1 = 1 \Rightarrow$ odd else even.

```

3 ► public class OddEven {
4
5 ►     public static void main(String[] args) {
6         int n = 68;
7         System.out.println(isOdd(n));
8     }
9
10    private static boolean isOdd(int n) {
11        return (n & 1) == 1;
12    }
13}
14

```

Output is :- false as 68 is even and $68 \& 1$ will return 0.

≡

- ②. An array is given, every number appears twice, but only one number appears once.
Find that number.

Soln:-

Logic:- XOR all elements in the array. We know, $a \wedge a = 0$. So, if the duplicate element is present, it will become 0.

```

public class FindUnique {
    public static void main(String[] args) {
        int[] arr = {2, 3, 3, 4, 2, 6, 4};
        System.out.println(ans(arr));
    }

    private static int ans(int[] arr) {
        int unique = 0;

        for(int n : arr) {
            unique ^= n;
        }

        return unique;
    }
}

```

Output:-

6.

≡

Q]. Find i^{th} bit of a number.

110110

654321
110110

Soln:

Suppose we have to find 3rd bit.

110110
 $\&$ 000100 → mask.

000100

Logic:- We use left shift to get this number.

To get 2 zeroes on the right side of 1 we use,

$1 \ll 2$ (here 3rd bit).

i.e $1 \ll (i-1)$

($i \rightarrow$ bit to find).

Later we 'AND' this $1 \ll (i-1)$ to number.

$n \& (1 \ll (i-1))$

Idea

```

1 import java.util.Scanner;
2
3 public class bit {
4     public static void main(String[] args) {
5         Scanner in=new Scanner(System.in);
6         System.out.println("Enter the value of n");
7         int n=in.nextInt();
8
9         System.out.println("Enter the value of i");
10        int i=in.nextInt();
11        System.out.println(fun(n,i));
12    }
13
14    static int fun(int n,int i){
15        return (n & (1<<(i-1)));
16    }
17
18 }

```

Run bit

```

C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.1\lib\idea_rt.jar=50988:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.1\bin
Enter the value of n
2
Enter the value of i
2
0

Process finished with exit code 0

```

Q). Set i^{th} bit to 1.

Solu. -

10 10 11 0
(OR) 00 00 0 → Mask.

Here $8 \rightarrow 1000$
 2nd bit is set
 So, $1010 = 10$

10 10 11 0
 Set to 1.

Project

```

1 Main.java 2 rotatedbinarysearch.java 3 pattern.java 4 bit.java

```

```

1 import java.util.Scanner;
2
3 public class bit {
4     public static void main(String[] args) {
5         Scanner in=new Scanner(System.in);
6         System.out.println("Enter the value of n");
7         int n=in.nextInt();
8
9         System.out.println("Enter the value of i");
10        int i=in.nextInt();
11        System.out.println(fun(n,i));
12    }
13
14    static int fun(int n,int i){
15        return (n | (1<<(i-1)));
16    }
17
18 }

```

Run bit

```

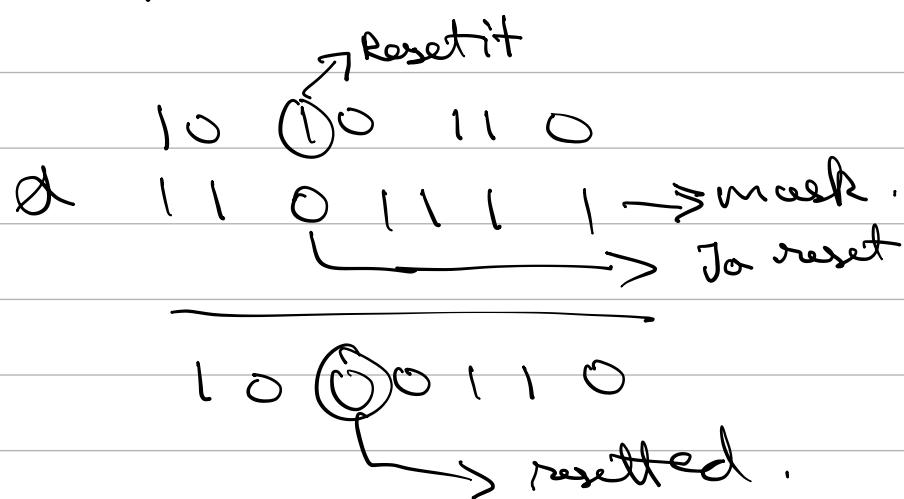
C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.1\lib\idea_rt.jar=50998:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.1\bin
Enter the value of n
8
Enter the value of i
2
10

Process finished with exit code 0

```

Q). Reset ith bit :-

Solu:-



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

A screenshot of an IDE (IntelliJ IDEA) showing a Java project named "DSA". The "bit" package is selected. The "bit.java" file is open, containing the following code:

```
import java.util.Scanner;
public class bit {
    public static void main(String[] args) {
        Scanner in=new Scanner(System.in);
        System.out.println("Enter the value of n");
        int n=in.nextInt();
        System.out.println("Enter the value of i");
        int i=in.nextInt();
        System.out.println(fun(n,i));
    }
    static int fun(int n,int i){
        return (n & ~(1<<(i-1)));
    }
}
```

The terminal window shows the execution of the program:

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.1\lib\idea_rt.jar=51022:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.1\bin" -Dfile.encoding=UTF-8
Enter the value of n
7
Enter the value of i
2
5
Process finished with exit code 0
```

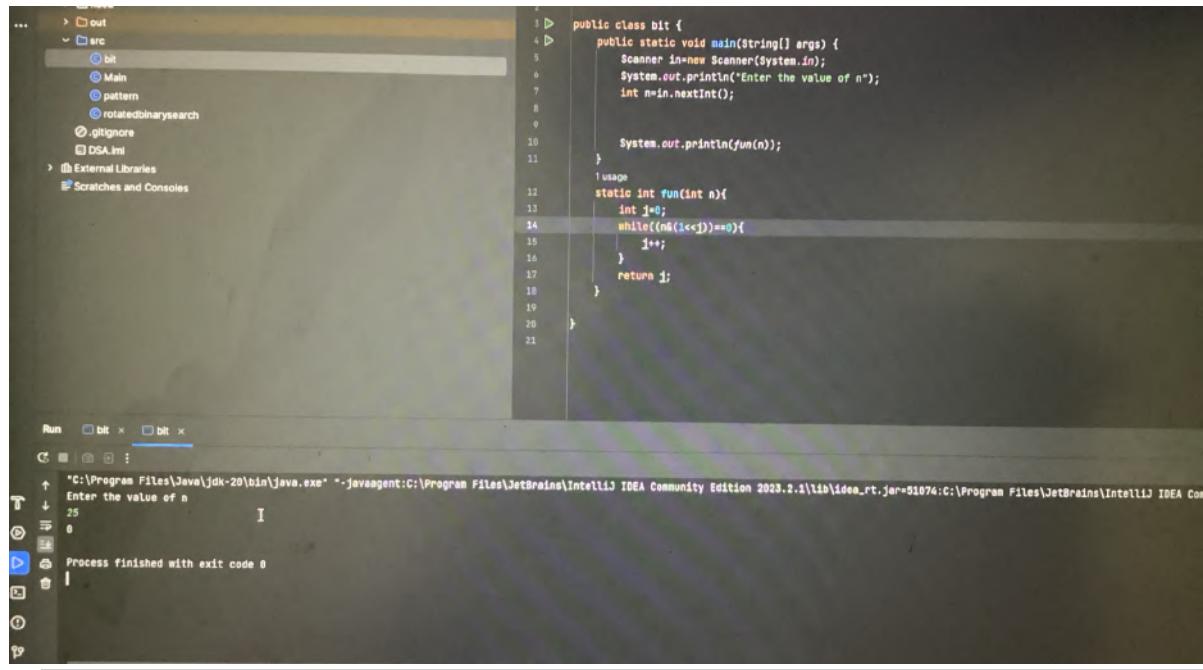
Q). Find the position of the rightmost set bit.

Solu:-

$$i = 0;$$

11 0

$\text{while}((n \& (1 \ll i)) != 0)$
 d
 }.
 i++;
 }.



```

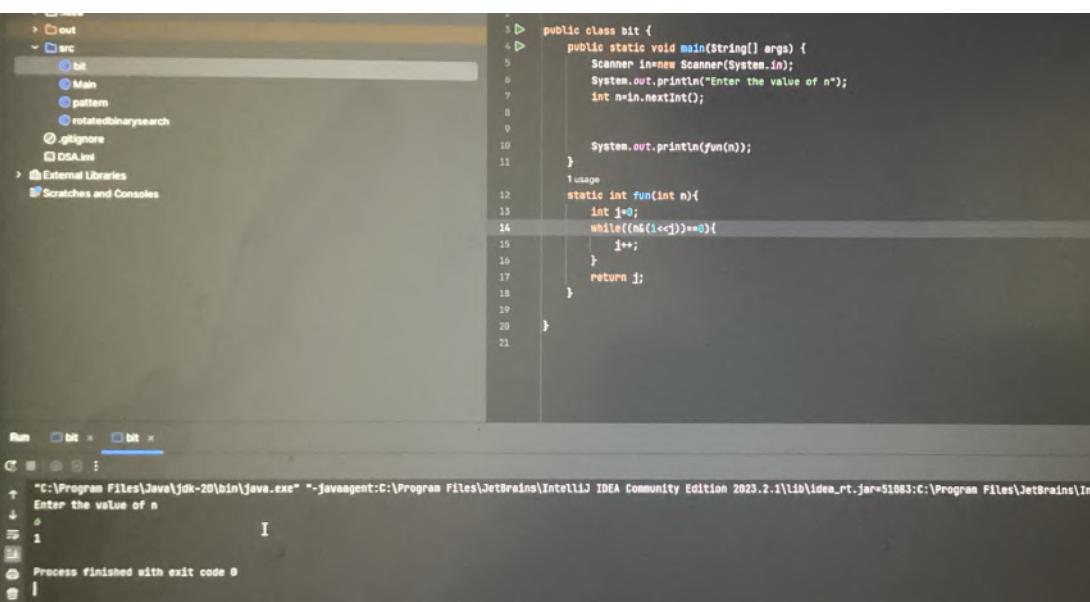
public class bit {
  public static void main(String[] args) {
    Scanner in=new Scanner(System.in);
    System.out.println("Enter the value of n");
    int n=in.nextInt();

    System.out.println(fun(n));
  }
  static int fun(int n){
    int j=0;
    while((n&(1<<j))==0){
      j++;
    }
    return j;
  }
}
  
```

The screenshot shows the IntelliJ IDEA interface with the code editor open. The code implements a function `fun` that returns the position of the rightmost set bit in a given integer `n`. The IDE's run tool window is visible at the bottom, showing the output of the program when run with the input `25`.

Here 2S., is

0 1 1 0 0 0 1 → 0th position



```

public class bit {
  public static void main(String[] args) {
    Scanner in=new Scanner(System.in);
    System.out.println("Enter the value of n");
    int n=in.nextInt();

    System.out.println(fun(n));
  }
  static int fun(int n){
    int j=0;
    while((n&(1<<j))==0){
      j++;
    }
    return j;
  }
}
  
```

The screenshot shows the IntelliJ IDEA interface with the code editor open. The code is identical to the one in the previous screenshot, but the run tool window shows the output for the input `6`.

Here 6, is

1 0 0 → 1st position.
So 1

SIMPLE FORMULA WITHOUT USING
WHILE LOOP IS.

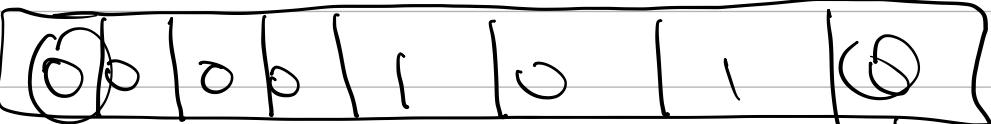
$$N \& (-N).$$

FOR THIS WE WILL LEARN

NEGATIVE BINARY NUMBERS :-

Negative of a number in Binary form

1 byte = 8 bits.

10 \Rightarrow 
 \downarrow MSB LSB

MSB \rightarrow Tells us whether the number is
+ve or -ve

1 \rightarrow -ve

0 \rightarrow +ve.

LSB \rightarrow Tells whether the number is even
or odd.

Step :-

① Take complement of no., ..

② Add 1 to it.

$$(10)_{10} = (00001010)_2$$

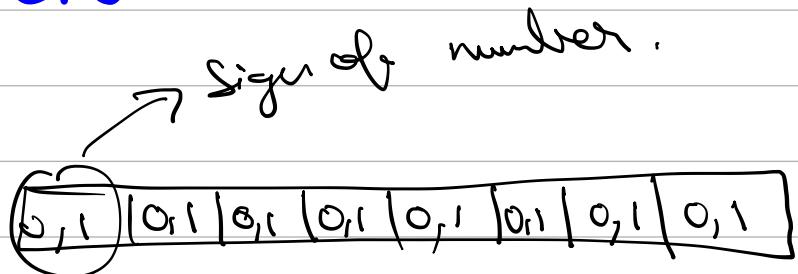
① $(1111\ 0101)$

② $1111\ 0101$

$$\begin{array}{r} + 1 \\ \hline (11110110)_2 \Rightarrow (-10). \end{array}$$

Range of numbers

i) 1 byte :-



$$\begin{aligned} \text{Total} &= 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \\ &= 2^8 = 256 \text{ numbers} \end{aligned}$$

Actual number stored in bit = $n - 1$

$n \rightarrow$ total bits.

i.e 1 byte = 7 bits. (1 bit for sign).

means $2^7 = 128$ numbers

When MSB = 1, -128 numbers
MSB = 0, 128 numbers.

from -128 to 127 there
are 257 numbers ($128+128$
- 1 (0 is in middle))

Range of Number

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

$n \rightarrow$ bits.

FORMULA

Questions Continued

Q8]. All numbers are repeating 3 times, only one number is appearing once, Find that time.

$$\text{arr} = [2, 2, 3, 2, 7, 7, 8, 7, 8, 8]$$

Soln:- Logic :- The numbers which are appearing 3 times, have their bit positions appearing 3 times as well. If we add all the numbers and take the modulo, we get the extra number.

$$\begin{array}{r} 10 \\ 10 \\ 11 \\ \hline 10 \\ 111 \\ 1000 \\ \hline 111 \\ 1000 \\ \hline 1000 \\ \hline 3374 \end{array}$$

→ There is extra
11 → means answer
is 3.

Q9]. Find the n^{th} magic number.

$$1 = \begin{matrix} 5^3 & 5^2 & 5^1 \\ 0 & 0 & 1 \end{matrix} \rightarrow \frac{\text{Magic no}}{5}$$

Amazon Question

$$2 = \begin{matrix} 0 & 1 & 0 \end{matrix} \rightarrow 10$$

$$3 = \begin{matrix} 0 & 1 & 1 \end{matrix} \rightarrow 130$$

$$4 = \begin{matrix} 1 & 0 & 0 \end{matrix} \rightarrow 125$$

,

{

:

1,

:

n

Solve if n is given. n^{th} magic number is

Loop

$n \neq 1 \Rightarrow$ This will give me last digit in binary.

$n >> 1 \Rightarrow$ To proceed to next number.

Till
 $n=0$
number
becomes
0.

```

2
3 public class MagicNumber {
4     public static void main(String[] args) {
5         int n = 6;
6
7         int ans = 0;
8         int base = 5;
9         while (n > 0) {
10             int last = n & 1;
11             n = n >> 1;
12             ans += last * base;
13             base = base * 5;
14         }
15
16         System.out.println(ans);
17     }

```

Q). Find number of digits in base b.

Soln

$(6)_{10} = 1 \rightarrow$ if $base = 10$, divide by 10 and increment counter till $n=0$.

$$(6)_{10} = (110)_2 = 3.$$

use right shift till $n=0$ increment the counter.

Other than these two methods,

General formula :-

$$= \text{int}(\log_b n) + 1$$

$n \rightarrow$ any number
 $b \rightarrow$ at which base the number is given.

Eg:-

$$(6)_2 \rightarrow_b$$

$$= \lceil \log_2 6 \rceil + 1$$

$$= 2 + 1 = 3 \text{ (correct)}$$

110

```
3 public class NoOfDigits {  
4     public static void main(String[] args) {  
5         int n = 34567;  
6         int b = 10;  
7  
8         int ans = (int)(Math.log(n) / Math.log(b)) + 1;  
9  
10        System.out.println(ans);  
11    }  
12 }  
13
```

Defn :-

$$\log_a b = \frac{\log_{10} a}{\log_{10} b}$$

PASCAL'S TRIANGLE :-

The screenshot shows an IDE interface with a project structure on the left and a code editor on the right. The code editor contains Java code for generating Pascal's triangle. The code uses a nested loop to print rows of the triangle, calculating factorials for each element using a helper function. The output window at the bottom shows the first few rows of the triangle.

```
1 1  
1 2 1  
1 3 3 1  
1 4 6 4 1  
1 5 10 10 5  
1 6 15 20 15 1
```

```
1  
1 1  
1 2 1  
1 3 3 1  
1 4 6 4 1  
1 5 10 10 5  
1 6 15 20 15 1
```

```
"C:\Program Files\Java\jdk-20\bin\java.exe" -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.1\lib\idea_rt.jar=56145:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.1\bin  
1  
11  
121  
1331  
Process finished with exit code 0
```

logic :- Each element position is given by.

n Ca.

$n \rightarrow$ row, 0, 1, 2, ...

$a \rightarrow$ column, 0, 1, 2, ...

Q). Find the sum of n^{th} row:-

Soln

Sum of each row =

$n=0, 1, 2, 3,$

$${}^n C_0 + {}^n C_1 + {}^n C_2 + \dots + {}^n C_n = \boxed{2^n}$$

For n^{th} row, sum = 2^{n-1}

Answer =

$$1 << (n-1)$$

$$\Rightarrow [1 \times 2^{n-1}], n = 1, 2, 3, \dots$$

Q). You are given a number, find out if

its power of 2 or not.

16

20

Eg:-

1 0 0 0 0



Power of two

1

1 0 1 0 0



more than one 1's
are there so, not
power of 2.

Generic method, keep right shifting,
increment counter when 1 is encountered

$$N \& 1 = 1$$

if counter > 1, not a power of 2.

Formula :- We know if a number n is given and we AND it with number -1 , we should get 0 , if it is power of 2 .

Eg:-

$$\begin{aligned} & 1000 = 8 \text{ (power of 2)} \\ \otimes & 111 = 7 \text{ (n-1 i.e } 8-1) \\ \hline & \underline{0000} = 0 \text{ (So 8 is power of 2).} \end{aligned}$$

$$(n \otimes (n-1)) == 0 ;$$

↓
returns true if it is power of 2

```
▶ public class PowOfTwo {  
▶   public static void main(String[] args) {  
    int n = 32;  
    boolean ans = (n & (n-1)) == 0;  
    System.out.println(ans);  
}  
}
```

→ Returns
true as
32 is
power of
2.

Q). Calculate a^b .

$$3^6 = 3 \times 3 \times 3 + 3 \times 3 \times 3.$$

Time complexity = $O(b)$.

Instead use,

$3^{(10)}$,
concept.

Logic :- Using right shifting of

power and multiplying with base (initially 3) when $\text{power} \& 1 = 1$
and ignore when

$\text{power} \& 1 = 0$, but

base = base \times base always.

```
public class Power {  
    public static void main(String[] args)  
    {  
        int base = 3;  
        int power = 6;  
  
        int ans = 1;  
  
        while (power > 0) {  
            if ((power & 1) == 1) {  
                ans *= base;  
            }  
  
            base *= base;  
            power = power >> 1;  
        }  
  
        System.out.println(ans);  
    }  
}
```

Q]. Given a number, find the number of set bits in it.

Solu.:-

Generic method,

Do, $n \& 1$

If $n \& 1 == 1$, counter ++,
 $n \gg 1$

Till $n == 0$. (i.e. $n > 0$)

Formula:-

We know, to get rightmost set bit, we used, $[N \& (-N)]$

Eg:- 1001

$$N \& (-N) \Rightarrow 0001$$

So, If we do $N - [N \& (-N)] = 1000$
(counter ++)

in loop until

$n > 0$,

counter gives no of set bits.

No of set bits = No of iterations.

(OR)

```
while (n > 0) {  
    counter++  
    n = n & (n - 1);  
}
```

Also
correct.

for example:-

$$\begin{aligned} n &= 1010 = 10 \\ n-1 &= \cancel{1}001 = 9 \\ \text{Now } n &= \cancel{1}000 = 8 \\ n-1 &= 0111 = 7 \\ n &= 0\cancel{0}000 \end{aligned}$$

\therefore
 $n=0$
terminate.
counter ≥ 2 .

```
11  
12     private static int setBits(int n) {  
13         int count = 0;  
14         while (n > 0) {  
15             count++;  
16             n -= (n & -n);  
17         }  
18  
19         while (n > 0) {  
20             count++;  
21             n = n & (n-1);  
22         }  
23  
24         return count;
```

Q). Find XOR of numbers from 0 to a.

Soln

a	XOR from 0 to a
0	0
1	$0^{\wedge} 1 = 1$
2	$0^{\wedge} 1^{\wedge} 2 = 0^{\wedge} 1^{\wedge} 10 = 11 = 3$
3	$0^{\wedge} 1^{\wedge} 2^{\wedge} 3 = 11^{\wedge} 11 = 0$
4	$0^{\wedge} 1^{\wedge} 2^{\wedge} 3^{\wedge} 4 = 0^{\wedge} 4 = 4$
5	1
6	7
7	0
8	8
9	1

Observe:-

$$\text{If } (a \% 4 = 0) \quad \text{XOR} = 0$$

$$a \% 4 = 1 \quad \text{XOR} = 1$$

$$a \% 4 = 2$$

$$\text{XOR} = a + 1$$

$$a \% 4 = 3$$

$$\text{XOR} = 0.$$

Loop this from 0 to a.

a) XOR of all numbers between
a & b.

Solu:-

$$\text{If } a = 3 \quad \& \quad b = 9.$$

$$3^1 4^1 5^1 6^1 7^1 8^1 9.$$

We already studied 0 to a.
so, to find 0 to b.

Do, 0 to b - 0 to a-1

$$\begin{array}{r} 0^1 1^1 2^1 3^1 4^1 5^1 6^1 7^1 8^1 9 \\ - 0^1 1^1 2 \\ \hline \end{array}$$

NOTE:- \wedge (XOR) operation in binary
is equivalent to subtraction.

So Logic :- $\text{ans} = \text{xor}(b) \wedge \text{xor}(a-1)$
+ ill b from 0 to a-1

```
3 public class RangeXOR {  
4     public static void main(String[] args) {  
5         // range xor for a, b = xor(b) ^ xor(a-1)  
6         int a = 3;  
7         int b = 9;  
8  
9         int ans = xor(b) ^ xor(a-1);  
10  
11         System.out.println(ans);  
12     }  
  
// this gives xor from 0 to a  
static int xor(int a) {  
    if (a % 4 == 0) {  
        return a;  
    }  
  
    if (a % 4 == 1) {  
        return 1;  
    }  
  
    if (a % 4 == 2) {  
        return a + 1;  
    }  
  
    return 0;  
}
```

Maths for DSA-II
Next Notes