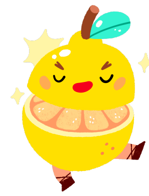
Bit Manipulation 2

> The first step to achieving your goal, is to take a moment to respect your goal. Know what it means to you to achieve it.
>
> DWAYNE JOHNSON



Good Evening

Today's content

01. Power of left shift
    (a) Set i^{th} bit
    (b) Flip i^{th} bit
    (c) Check i^{th} bit

02. Check bit is set or not
03. Count set bits
04. Negative number
05. Ranges
06. Importance of constraints

A Good news & a bad news today

Left shift Operator

$$n = \overset{5}{1}\overset{4}{0}\overset{3}{1}\overset{2}{0}\overset{1}{1}\overset{0}{0} \Rightarrow 2^5 + 2^3 + 2^1 = 32 + 8 + 2 = 42$$

$$n << 1 = \cancel{x}\,0\,1\,0\,1\,0\,0 \rightarrow 2^4 + 2^2 = 16 + 4 = 20$$

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

Quiz $7 << 3$

$$\underbrace{\overset{31}{0}00 \ldots 0\,0\,0\,1\,1\,1}_{} << 3$$

$$\ldots 0\,0\,1\,1\,1\,0\,0\,0$$

x

$$\Rightarrow 2^5 + 2^4 + 2^3 = 56$$

Quiz $15 << \boxed{2} \Rightarrow 15 * 2^2$

$$= 15 * 4 = 60$$

$$\underbrace{00}000 \ldots 01111$$

$$000 \ldots 0111100$$

- $1 << 0 = 1$ (No shifting)
- $1 << 1 = \underbrace{0}\,000000 \ldots 0001 << 1$

$$00000 \ldots 000\overset{}{1}0 \longrightarrow 2^1 \Rightarrow 2$$

$$ \,4\;3\;2\;1\;0$$

- $1 << 2 = 4$
- $1 << 3 = 1 * 2^3 = 8$

$$2^n = 1 << n$$

One thing ⤵

$2 << (-1)$

$=$

$2 >> 1$

returns 0

* Power of left shift

* $45 = (101101)_2$

01. Left shift with OR

45	=	1	0	1	1	0	1
OR $1<<2$	=	0	0	0	1	0	0
		1	0	1	1	0	1

45	=	1	0	1	1	0	1
OR $1<<4$	=	0	1	0	0	0	0
		1	1	1	1	0	1

45	=	1	0	1	1	0	1
OR $1<<1$	=	0	0	0	0	1	0
		1	0	1	1	1	1

Left shift with OR operator will set the particular bit

$N \mid (1<<i) \rightarrow$ set the i^{th} bit

02. Left shift with XOR → same same puppy shame

0 0 = 0

1 0 = 1

$$\begin{array}{lrcccccc}
 & 45 = & 1 & 0 & 1 & 1 & 0 & 1 \\
\text{XOR} & & & & & & & \\
 & 1<<2 = & 0 & 0 & 0 & 1 & 0 & 0 \\
\hline
 & & 1 & 0 & 1 & 0 & 0 & 1
\end{array}$$

$$\begin{array}{lrcccccc}
 & 45 = & 1 & 0 & 1 & 1 & 0 & 1 \\
\text{XOR} & & & & & & & \\
 & 1<<4 = & 0 & 1 & 0 & 0 & 0 & 0 \\
\hline
 & & 1 & 1 & 1 & 1 & 0 & 1
\end{array}$$

$$\begin{array}{lrcccccc}
 & 45 = & 1 & 0 & 1 & 1 & 0 & 1 \\
\text{XOR} & & & & & & & \\
 & 1<<1 = & 0 & 0 & 0 & 0 & 1 & 0 \\
\hline
 & & 1 & 0 & 1 & 1 & 1 & 1
\end{array}$$

Toggling the bits with the help of XOR & left shift

$N \,\hat{}\, (1<<i) \rightarrow$ toggle the i^{th} bit

03. AND with left shift operator

$$
\begin{array}{rcl}
45 & = & 1\ 0\ 1\ 1\ 0\ 1 \\
\text{AND} & & \\
1 \ll 2 & = & 0\ 0\ 0\ 1\ 0\ 0 \\
\hline
 & & 0\ 0\ 0\ 1\ 0\ 0
\end{array}
$$

$$
\begin{array}{rcl}
45 & = & 1\ 0\ 1\ 1\ 0\ 1 \\
\text{AND} & & \\
1 \ll 4 & = & 0\ 1\ 0\ 0\ 0\ 0 \\
\hline
 & & 0\ 0\ 0\ 0\ 0\ 0
\end{array}
$$

$$
\begin{array}{rcl}
45 & = & 1\ 0\ 1\ 1\ 0\ 1 \\
\text{AND} & & \\
1 \ll 1 & = & 0\ 0\ 0\ 0\ 1\ 0 \\
\hline
 & & 0\ 0\ 0\ 0\ 0\ 0
\end{array}
$$

$N \,\&\, (1 \ll i)$

- $\rightarrow 0 \rightarrow$ i^{th} bit is unset
- $\rightarrow 1 \ll i \rightarrow$ i^{th} bit is set

Q1. Unset the i^{th} bit of a given number if it is set, otherwise do nothing

$n = 45$
$i = 2$

$$\begin{array}{cccccc} 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{array} \longrightarrow 101001$$

$n = 45$
$i = 4$

$$\begin{array}{cccccc} 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{array} \longrightarrow 101101$$

Idea → Check if the i^{th} bit is set, if yes, then toggle it

```
if (checkbit(n, i) == true)
{
    n = n ^ (1<<i);
}
else {
    // do nothin
}
```

Q How to check if i^{th} bit is set or unset?

01. AND

$N \,\&\, (1<<i) == (1<<i) \rightarrow$ i^{th} bit is set

02 OR

$N \mid 1 \rightarrow N$ = last bit is set

$N \mid 1 \rightarrow\, > N$ = last bit is unset

$N \mid (1<<i) == N \rightarrow$ i^{th} bit is set

$N \mid (1<<i) > N \rightarrow$ i^{th} bit is unset

$N = 1\,0\,1\,0$

$1<<2 = 0\,1\,0\,0$

Ans $1\,1\,1\,0$

<u>Ans > N</u>

$N = 1\,0\,1\,1$

$1<<3 = 1\,0\,0\,0$

Ans $= 1\,0\,1\,1$

Ans == N

<u>03</u> XOR operator $\rightarrow$ {TODO}

* Right shift operator $\Rightarrow$ $a >> x = \dfrac{a}{2^x}$

01. $10 >> 2 \Rightarrow \dfrac{10}{2^2} = \dfrac{10}{4} \Rightarrow \boxed{2}$

$0\,0\,0\,0\,1\,0\,\underbrace{1\,0} >> 2$

$\times$

$0\,0\,0\,0\,0\,0\,1\,0$

$\Downarrow$

$\boxed{2}$

02. $8 >> 3 = \dfrac{8}{2^3} = \dfrac{8}{8} = 1$

03 $8 >> 4 = \dfrac{8}{2^4} = 0$

04. $1 >> 2 = 0$

$N \,\&\, 1 \rightarrow$
- 0 {last bit is unset}
- 1 {last bit is set}

$\Rightarrow n = 1010$

$$
\begin{array}{rcl}
n >> 0 & = & 1\,0\,1\,0 \\
\& \quad 1 & = & 0\,0\,0\,1 \\
\hline
 & & 0\,0\,0\,0
\end{array}
$$

claim → 0^{th} bit is unset

$$
n = 45 \rightarrow 1\,0\,1\,1\,0\,1
$$

$$
\begin{array}{rcl}
n >> 0 & \rightarrow & 1\,0\,1\,1\,0\,1 \\
\& \quad 1 & \rightarrow & 0\,0\,0\,0\,0\,1 \\
\hline
 & & 0\,0\,0\,0\,0\,1
\end{array}
$$

claim → 0^{th} bit is set

$$
\begin{array}{rcl}
n & \rightarrow & 1\,0\,1\,1\,0\,1 \\
n >> 1 & \rightarrow & 0\,1\,0\,1\,1\,0 \quad \times
\end{array}
$$

$$
\begin{array}{rcl}
n >> 1 & \rightarrow & 0\,1\,0\,1\,1\,0 \\
\& \quad 1 & \rightarrow & 0\,0\,0\,0\,0\,1 \\
\hline
 & & 0\,0\,0\,0\,0\,0
\end{array}
$$

$n >> 2$ 0 0 1 0 1 1

&   1   0 0 0 0 0 1

0 0 0 0 0 1

Check if i^{th} bit is set or not

$(n >> i) \& 1$ → 0 (i^{th} bit is unset)
→ 1 (i^{th} bit is set)

Q Count the no. of set bits in a no.

$n = 45$ 1 0 1 1 0 1   Ans = 4

$n = 8$ 0 0 1 0 0 0   Ans = 1

Idea 1 → Check for every bit

set	unset
$c = c + 1$	// do nothing

```
count = 0
for (i=0 ; i<32 ; i++)
|    if (checkbit(n, i) == true)
|    |    count += 1
|    }
}
return count ;
```

Not a Generalised code

$n = 2 \Rightarrow$ 0 0 0 0 0 . . . . 0 0 0 1 0

31 30 29 . . . . 4 3 2 1 0

Issues with the code

01. Unnecessary iterations

02. Hard code by taking 32

int $i : 0 \to 31$

long $i : 0 \to 63$

byte $i : 0 \to 7$

Idea 2

count = 0

$n = \ldots 1\ 0\ 1\ 0$		0
$n>>1 =$ 0 1 0 1	×	1
$n>>2 =$ 0 0 1 0	×	0
$n>>3 =$ 0 0 0 1	×	1
$n>>4 =$ 0 0 0 0		2

└ if n==0 return count

```
count = 0
while (n>0)
|   if (n & 1 == 1) {
|   |   count = count + 1;
|   }
|   n = n>>1;  ──→ n/2
}
return count;
```

TC: $O(\log_2 n)$

SC: $O(1)$

Break time : 10:20 → 10:30 pm

* **Negative numbers**

$(-45)_{10} \longrightarrow (\quad)_2$

int = __ __ __ __ . . . __ __ __ __

31	30	29	28		3	2	1	0
2^{31}	2^{30}	2^{29}	2^{28}	...	2^3	2^2	2^1	2^0

$$= \underbrace{2^{30} + 2^{29} + 2^{28} + \cdots + 2^{3} + 2^{2} + 2^{1} + 2^{0}}_{2^{0} + 2^{1} + 2^{2} + \cdots + 2^{30}}$$

$$\frac{a(r^n - 1)}{r - 1} \Rightarrow \frac{2^{0} * (2^{31} - 1)}{2 - 1} = \boxed{2^{31} - 1}$$

$$\underbrace{2^{31}} > 2^{30} + 2^{29} + 2^{28} + \cdots + 2^{2} + 2^{1} + 2^{0}$$

MSB → Most significant bit

MSB → 0 { no. is positive }

MSB → 1 { no is negative }

$(-45)_{10} \rightarrow (\qquad\qquad)_2$

Assumption → int → 8 bits

$(45)_{10}$ = 0 0 1 0 1 1 0 1

01. 1's compliment ⇒ Toggle each & every bit

02 2's compliment ⇒ 1's compliment + 1

(45) = 0 0 1 0 1 1 0 1

1's compli = 1 1 0 1 0 0 1 0

+ 1 = 0 0 0 0 0 0 0 1

2's compliment =	1	1	0	1	0	0	1	1
	7	6	5	4	3	2	1	0

$$-2^7 + 2^6 + 2^4 + 2^1 + 2^0$$

$$\Rightarrow -128 + 64 + 16 + 2 + 1$$

$$= -128 + 83$$

$$\Rightarrow -45$$

rem = sum % 2

carry = sum / 2

02. $(-12)_{10} \rightarrow (\qquad\qquad)_2$

12 =	0	0	0	0	1	1	0	0
						1	1	
1's compliment =	1	1	1	1	0	0	1	1
+ 1	0	0	0	0	0	0	0	1
2's compliment =	1	1	1	1	0	1	0	0
	7	6	5	4	3	2	1	0

$$Ans = -2^7 + 2^6 + 2^5 + 2^4 + 2^2$$

$$= -128 + 64 + 32 + 16 + 4$$

$$= -128 + 116$$

$$\Rightarrow -12$$

* **Ranges**

01. Byte = 8 bits

Min num = 1 0 0 0 0 0 0 0 = -128

$\Rightarrow -2^7$

max num = 0 1 1 1 1 1 1 1 $\Rightarrow 2^7 - 1$

-128 to 127

02 int $\longrightarrow$ 32 bits

min num = 1 0 0 0 0 0 . . . . 0 0 0 0 0 $\rightarrow -2^{31}$

31 30 29 — — — . . 4 3 2 1 0

max num = 0 1 1 1 1 1 . . . 1 1 1 1 1 $= 2^{31} - 1$

Range $= -2^{31}$ to $2^{31} - 1$

Range $= -2 * 10^9$ to $2 * 10^9$

03. long = 64 bits

min num = $\underset{63}{\underline{1}}$ $\underline{0}$ $\underline{0}$ $\underline{0}$ $\underline{0}$ . . . $\underline{0}$ $\underline{0}$ $\underline{0}$ $\Rightarrow -2^{63}$

max num = $\underline{0}$ 1 1 1 1 1 . . . 1 1 1 1 $\Rightarrow 2^{63} - 1$

Range $= -2^{63}$ to $2^{63} - 1$

Q Given two integers a & b. Return $a * b$

constraints $\begin{cases} a \le 2 * 10^9 \\ b \le 2 * 10^9 \end{cases}$

01 int ans = a * b → incorrect

return ans

overflow

02 long ans = a * b → incorrect

return ans;

Overflow

$$\underbrace{\text{int} * \text{int}}_{\text{int}} \rightarrow \text{garbage value}$$

03. long ans = long (a * b) → incorrect

overflow

$$\underbrace{\text{int} * \text{int}}_{\text{int}} \rightarrow (\text{garbage value})$$

04. long ans = long(a) * b $\longrightarrow$ Correct

$$\underbrace{\text{long} * \text{int}}_{\text{long}}$$

05 long ans = a;
ans *= b;
return ans

Correct

———— & ———— & ———— & ———— & ————

Doubts

Signed integer - MSB → 0 positive
→ 1 negative

unsigned integer -

⇒ Unique elements when ele are repeating

$\boxed{a \wedge a = 0}$

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

⇒ array = 2 3 4 5 3 5 2

⇒ $2 \wedge 3 \wedge 4 \wedge 5 \wedge 3 \wedge 5 \wedge 2 \longrightarrow 4$

= $2 \wedge 2 \wedge 3 \wedge 3 \wedge 5 \wedge 5 \wedge 4 \longrightarrow 4$

* **Subarray with OR = 1**

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

⇒

Subarray whose OR = 1

Total no. of subarrays. $= \dfrac{n(n+1)}{2} \rightarrow \dfrac{9 * 10}{2} = \underline{\underline{45}}$

<u>Obs</u> → Even if single 1 is present, the OR of complete subarray is 1.

* count the no. of subarrays only having zeroes

Ans = Total no. of subarrays — <u>subarrays only having zero</u>

```
count = 0
for (i=0; i<n; i++)
{
    if (arr[i] == 0)
    {
        zeroes ++;
    }
    else {
        count = count + (zeroes * (zeroes+1)/2)
        zeroes = 0
    }
}
count = count + (zeroes * (zeroes+1)/2)
```

$$Ans = \frac{n*(n+1)}{2} - count;$$