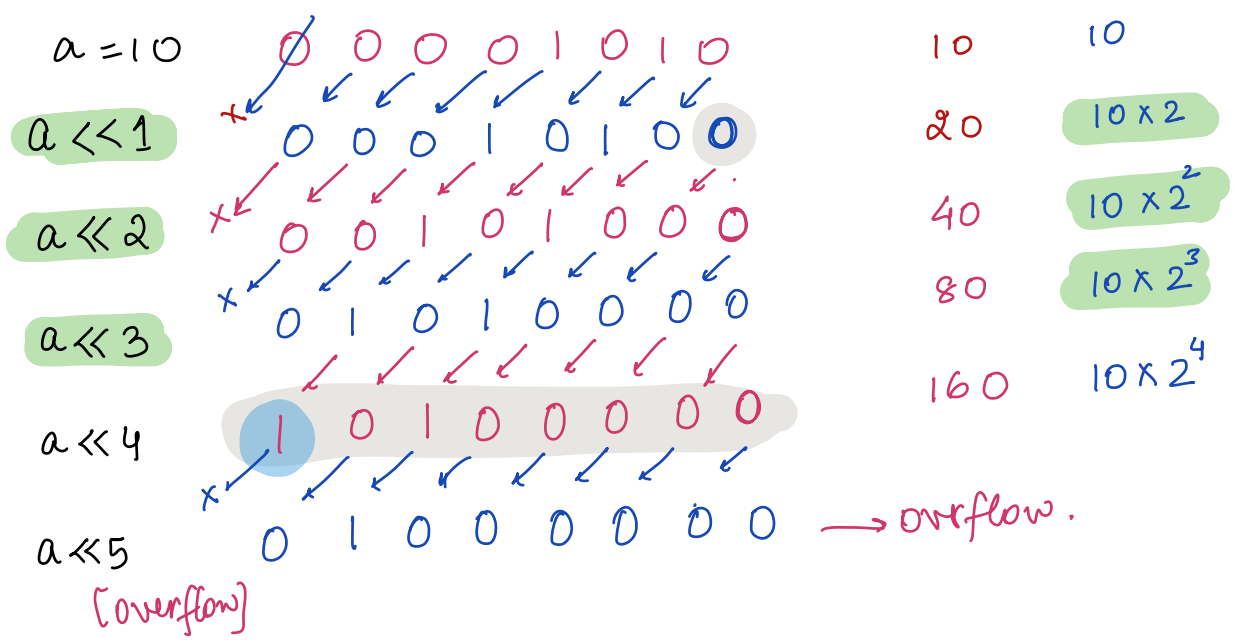
- left shift operator
- right shift operator
- Bit Masking problems
- .

Left Shift Operator. <<

literally shifts bits towards left.

8 bit



In general, $a << n = a * 2^n$

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

$a = 1$

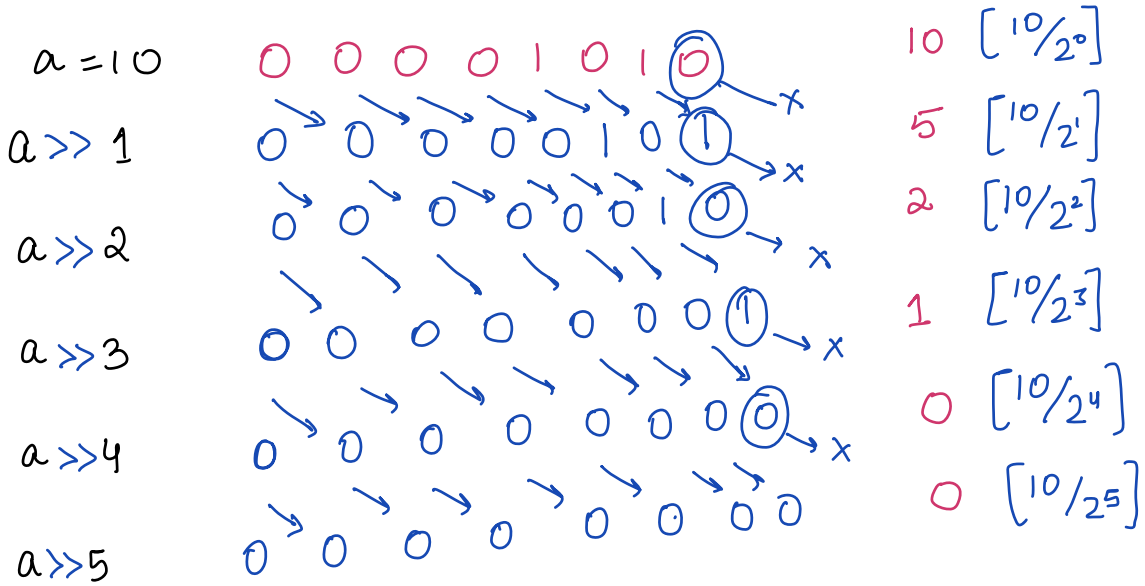
$a << 1 = 1 * 2^1$

$a << 2 = 1 * 2^2$

$a << 3 = 1 * 2^3$

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

Right Shift



$$a >> n = a/2^n$$

QUIZ 1

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

$15 << 2$

$15 * 2^2 = 60$

QUIZ 2

$a >> n = a / 2^n$

$29 >> 2$

$29 / 2^2 = \frac{29}{4} = 7$

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

$1 << n$

$5 << n \rightarrow 5 * 2^n$

QUIZ

$5^n = ?$

- $5 * (1 << n) \longrightarrow 5 * (2^n)$ X
- $5 << n \longrightarrow 5 * (2^n)$ X
- $5 << (n-1) \longrightarrow 5 * 2^{n-1}$ X

Amazon

Question: Given very nos. N & i, check if ith bit is set or not in N.

N = 21 i = 2

	4	3	2	1	0
21 :	1	0	1	0	1

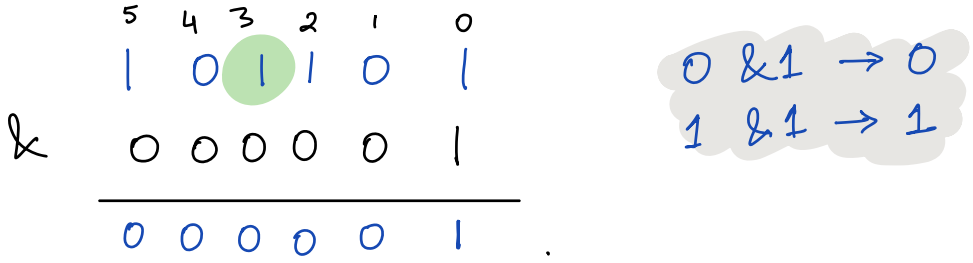
set / true.

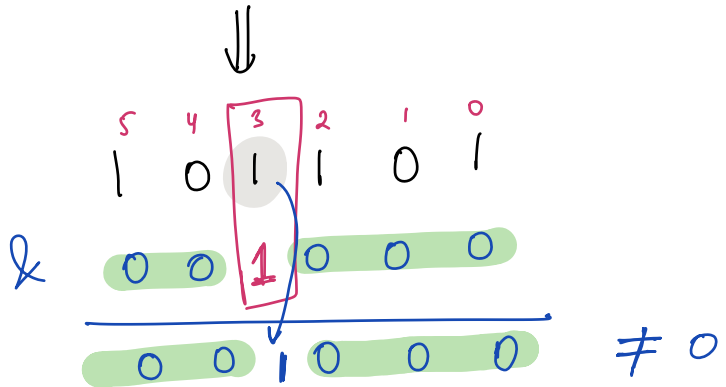
N = 25 i = 1

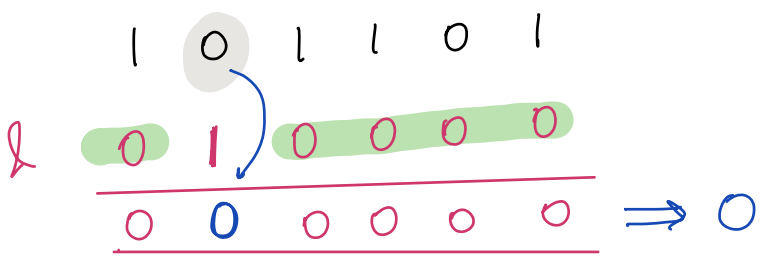
	4	3	2	1	0
25 :	1	1	0	0	1

unset / false

Have we done something to check the rightmost bit?







5 4 3 2 1 0

1 0 1 1 0 1 → N = 45

Left shift

1 << i

0 0 0 0 0 1 → 0th.

0 0 0 0 1 0 → 1st

0 0 0 1 0 0 → 2nd

0 0 1 0 0 0 → 3rd

N & (1 << i) → 0 [unset]

N & (1 << i) → !0 [set]

```
bool checkBit (int N, int i) {
    if( N & (1 << i)) {
        return true
    }
    else {
        return false
    }
}
```

TC → O(1)

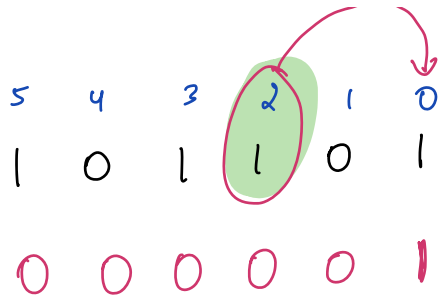
SC → O(1)

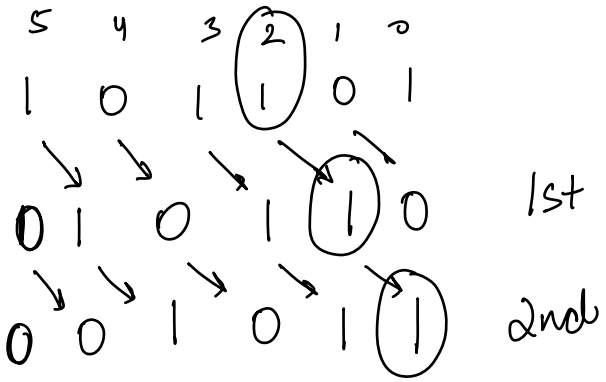
10th

7 6 5 4 3 2 1 0

0 1 0 1 1 0 1

i = 10th

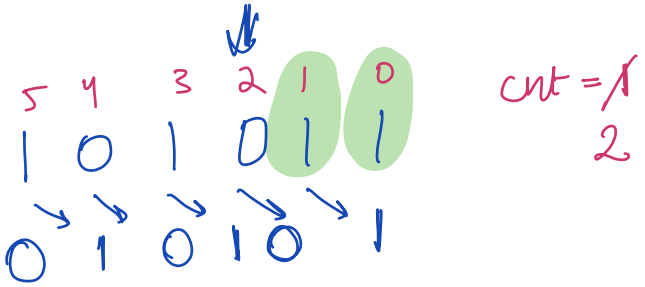




$N >> i$

```
bool checkBit2 ( int N, int i) {
    if ((N >> i) & 1) {
        return true
    }
    else {
        return false
    }
}
```

TC → O(1)
SC → O(1)



(2) Given N, i. set ith bit of N

1 → 1
0 → 1

	7	6	5	4	3	2	1	0	
N = 4	0	0	0	0	0	1	0	0	
i = 0	0	0	0	0	0	1	0	1	⇒ 5
i = 1	0	0	0	0	0	1	1	0	⇒ 6
i = 2	0	0	0	0	0	1	0	0	⇒ 4

Return the resulting number.

0 | 1 → 1

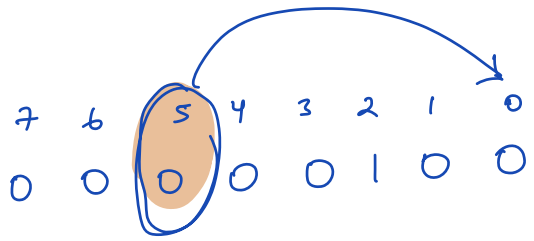
0 | 1 → 1
1 | 1 → 1

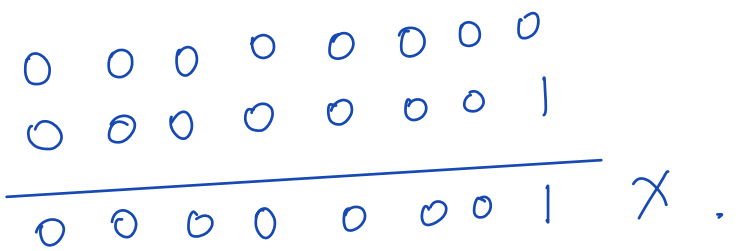
1 | 0 → 1
0 | 0 → 0

	7	6	5	4	3	2	1	0	
	0	0	0	0	0	1	0	0	
(OR)	0	0	1	0	0	0	0	0	
	0	0	1	0	0	1	0	0	⇒ $2^5 + 2^2 = 36$

```
int setBit(N, i){
    return N | (1 << i)
}
```

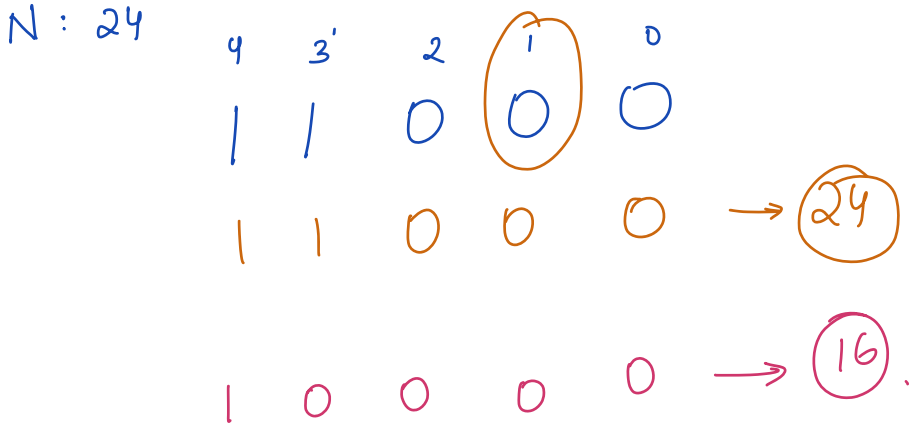
((N >> i) | 1) X





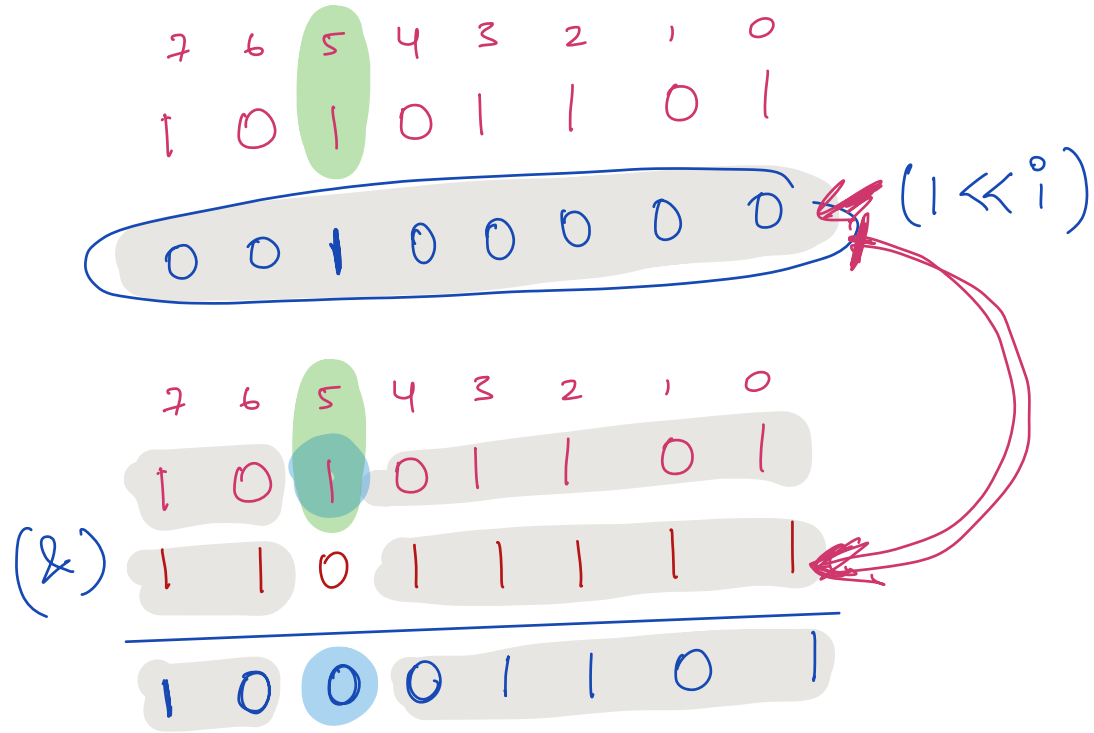
Clearing a Bit

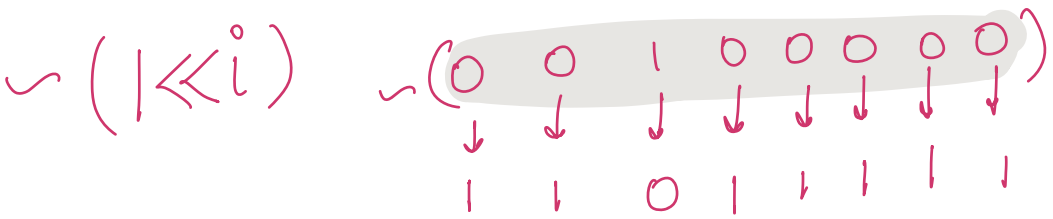
Given N, i ; clear ith bit in N



1 & 0 → 0

0 & 0 → 0






```
int clearBit( N, i) {
    return N & ~(1<<i)
}
```

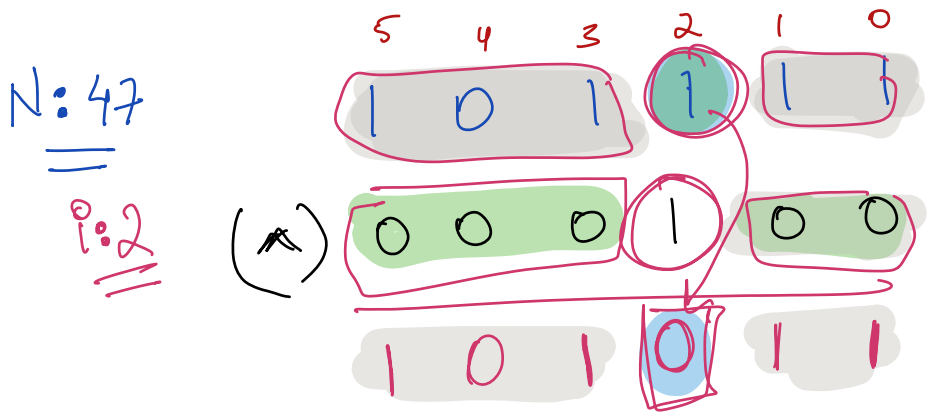
Break → 10:40

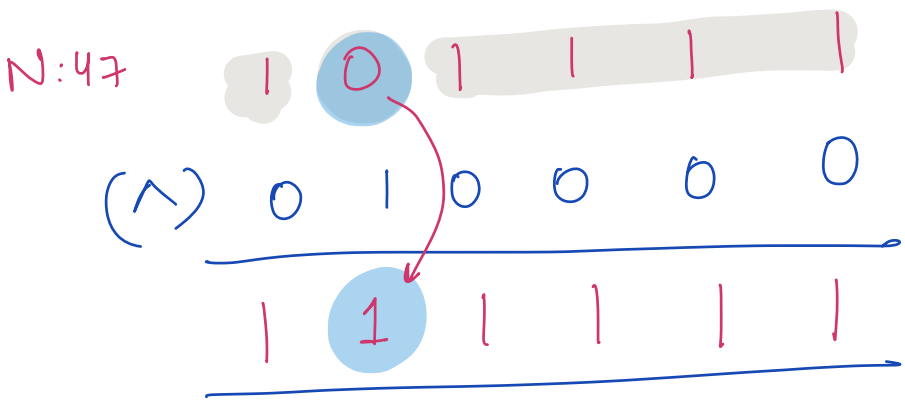
N ^ (1 << i)

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

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

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

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

Qu. N, i ; toggle the ith bit

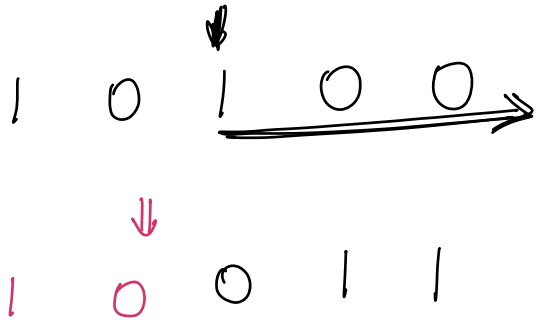




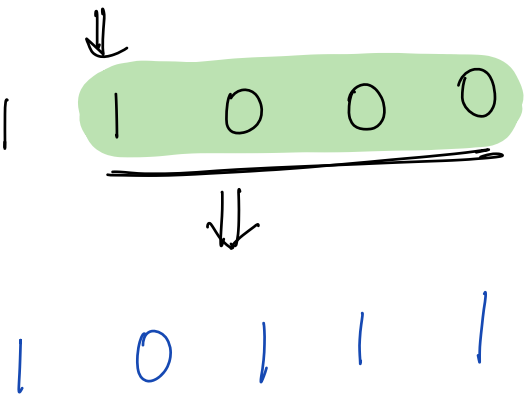

```
int toggleBit (N, i) {
    return N ^ (1 << i)
}
```

Given a ve+ no. N, toggle all the bits starting from the rightmost set Bit.

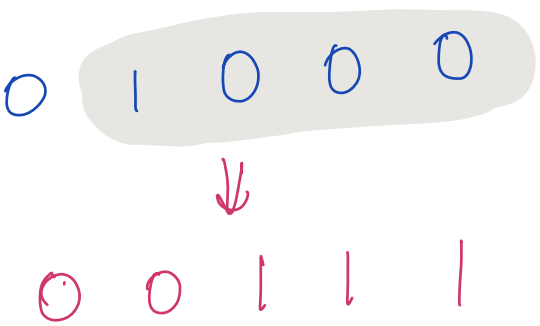
N = 20



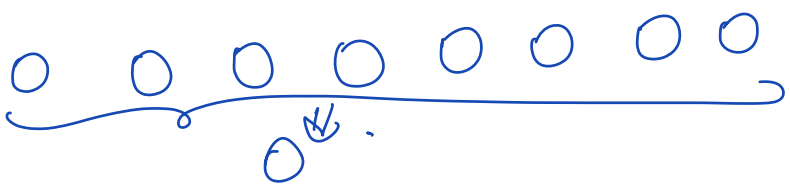
N = (24)



N = 8



N = 0



```
if(N==0) return 0   // edge case
i=0
while((1<<i)<N){
    if(checkBit(N,i)==0){
        setBit(N,i)
    }else{
        toggleBit(N,i)
        break
    }
    i++
}
```

TC

$O(\log_2 N)$

SC

$O(1)$

N = 20

2^4	2^3	2^2	2^1	2^0
1	0	1	0	0
1	0	0	1	1

Observation

N	N-1	N & (N-1)
3: 0 1 1	0 1 0 ⇒ 2	0 1 0
5: 1 0 1	1 0 0 ⇒ 4	1 0 0
15: 1 1 1 1	1 1 1 0 ⇒ 14	1 1 1 0
20: 1 0 1 0 0	1 0 0 1 1 ⇒ 19	1 0 0 0 0
24: 1 1 0 0 0	1 0 1 1 1 ⇒ 23	1 0 1 0 0 0
44: 1 0 1 1 0 0	1 0 1 0 1 1 ⇒ 43	

```
int togglebyRight {
    |    return (N-1)
    }
```

Google | Amazon

Q. Given an int N, count the no. of set bits in N.

N: 44 → 1 0 1 1 0 0

ans: 3

```
ans = 0
while (N > 0) {
    if (N & 1 == 1) ans++
    N = N >> 1
}
```

N = N >> 1 → $N = \frac{N}{2}$

TC = $O(\log_2 N)$

SC = $O(1)$

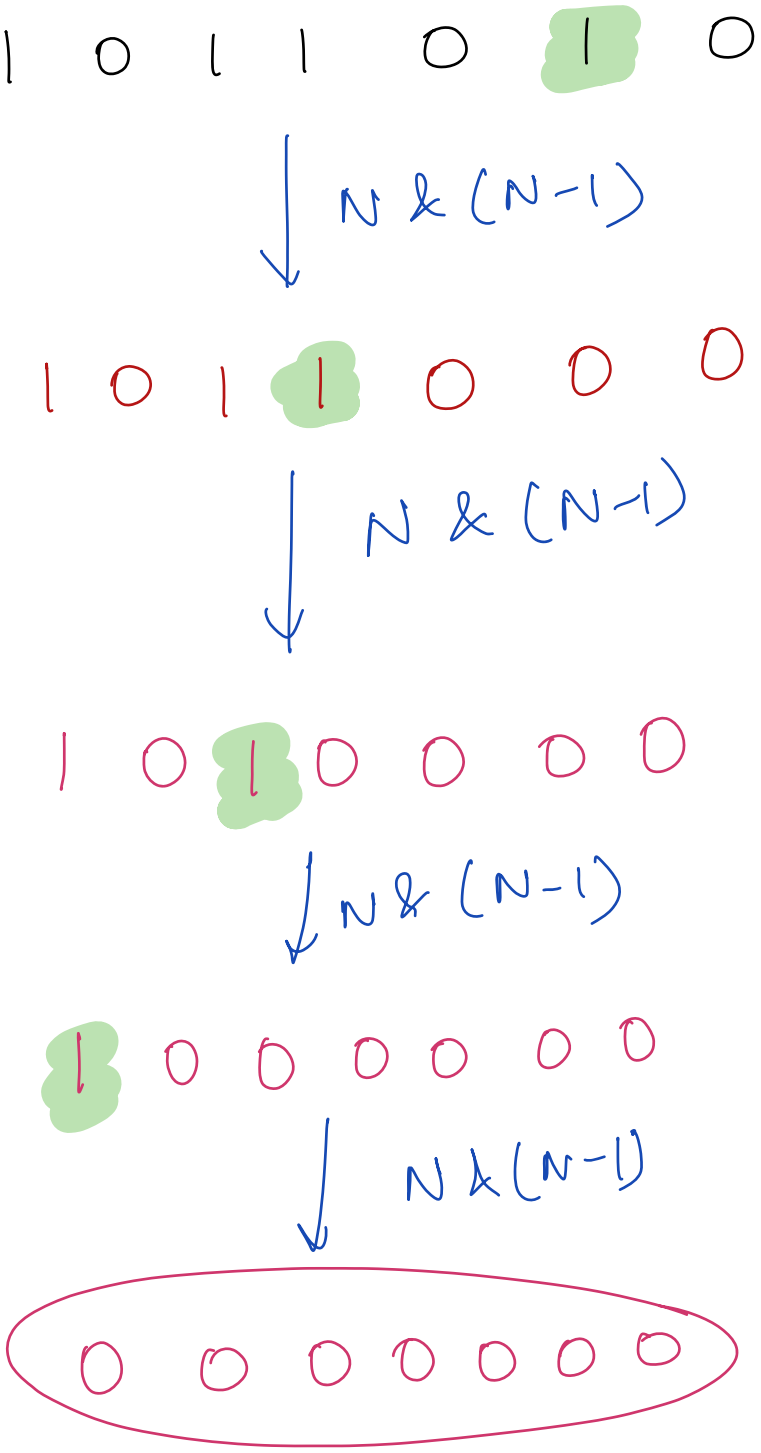
	N	N-1	N & (N-1)
3:	0 1 1	0 1 0 ⇒ 2	0 1 0
5:	1 0 1	1 0 0 ⇒ 4	1 0 0
15:	1 1 1 1	1 1 1 0 ⇒ 14	1 1 1 0
20:	1 0 1 0 0	1 0 0 1 1 ⇒ 19	1 0 0 0 0
24:	1 1 0 0 0	1 0 1 1 1 ⇒ 23	1 0 0 0 0
44:	1 0 1 1 0 0	1 0 1 0 1 1 ⇒ 43	1 0 1 0 0 0

1 & 0 → 0

0 & 1 → 0

Result of the toggled form is always 0.

N & (N-1) ⟹ rightmost bits are no more.



No. of times I'm clearing 1s = Count of 1s.

Previous code

$15 = 1\ 1\ 1\ 1 \Rightarrow 4$

$16 = 1\ 0\ 0\ 0\ 0 \Rightarrow 1$

$17 = 1\ 0\ 0\ 0\ 1 \Rightarrow 2$

$2^{30} = 1\ 0\ 0\ 0 \cdots 0 \Rightarrow 1$

$\log_2 15$

$\log_2 16$

$\log_2 17$

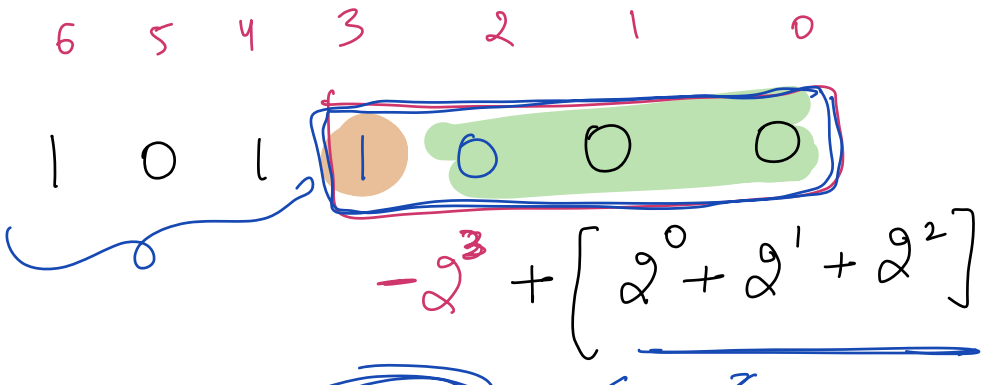
$\log_2(2^{30}) = 30$

```
int countSetBits(N){
    int ans = 0
    while (N > 0){
        N = N & (N-1)
        ans++
    }
    return ans;
}
```

$TC = O(\log_2 N)$

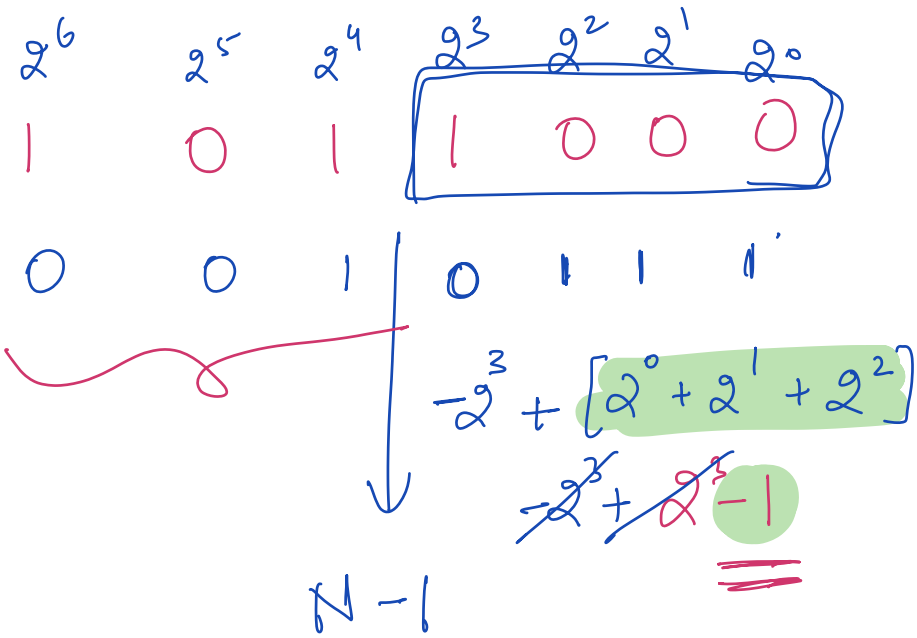
$SC = O(1)$

$N \longrightarrow N-1$



$-2^3 + [2^0 + 2^1 + 2^2]$

$-2^3 + 2^0 - 1$



0 0 0 1 0 0

↓ N * (N-1)

0 0 0 0 0