Today's Content

- → >> Right Shift
- → Check Bit / Count Set Bits
- → Set x^{th} & y^{th} bit
- → Unset i^{th} bit
- → Set continious x & y Bits
- → MSB

/ → Integer division

$10/4 \rightarrow 5/2 \rightarrow 2$

8 bit

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		Decimal
$n = 10$	0	0	0	0	1	0	1	0		$= 2^3 + 2^1 = 10$
$n >> 1$:	0	0	0	0	0	1	0	1	discard	$= 2^2 + 2^0 = 5 = 10/2$
$n >> 2$:	0	0	0	0	0	0	1	0	discard	$= 2^1 = 2 = 10/2^2$
$n >> 3$:	0	0	0	0	0	0	0	1	discard	$= 2^0 = 1 = 10/2^3$
$n >> 4$:	0	0	0	0	0	0	0	0	discard	$= 0 = 10/2^4 = 0$
$n >> 5$:										$10/2^5 = 0$

// generalize :

$n >> 1 \Rightarrow n/2$

$n >> 2 \Rightarrow n/2^2$

$n >> 3 \Rightarrow n/2^3$

$\boxed{n >> i \Rightarrow n/2^i}$

bit positions

int a → 4 Bytes → 32 bits : [31 - 0]

long a → 8 Bytes → 64 bits : [63 - 0]

Given N, i, check if i^{th} bit pos in N is set or Not?

set → 1, Not → 0

	4	3	2	1	0
N = 21 :	1	0	1	0	1

i = 2

Set

	5	4	3	2	1	0
N = 34 :	1	0	0	0	1	0

i = 3

not set

// check if i^{th} bit in N is set or not?

decimal number

```
bool checkBit(int N, int i){
```

Ideal: Convert decimal → Binary & store it & check if its ith bit == 1 or not

with above logic: loop

//

N: 82

	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
i = 0 :	1	0	1	0	0	1	0	(N & 1) == 1 : Set
i = 1, N>>1 :	—	1	0	1	0	0	1	(N>>1) & 1 == 1 : Set
i = 2, N>>2 :	—	—	1	0	1	0	0	(N>>2) & 1 == 1 : Set
i = 3, N>>3 :	—	—	—	1	0	1	0	(N>>3) & 1 == 1 : Set
i = 4, N>>4 :	—	—	—	—	1	0	1	(N>>4) & 1 == 1 : Set

i, N>>i : { i^{th} in N >> 0th bit pos } : $\boxed{(N>>i) \,\&\, 1 == 1 : \text{Set}}$

{int}

```
bool checkBit( int N, int i){
    if( (N >> i) & 1 == 1) {
        return True
    }
    else { return False }
}
```

a == b
↳ comparison

```
return (N >> i) & 1 == 1
```

→ TC: O(1) time → SC: O(1)

N = 82

1st bit pos

	2^6	2^5	2^4	2^3	2^2	$\boxed{2^1}$	2^0	
N :	1	0	1	0	0	1	0	
: i = 0 :	0	0	0	0	0	0	1	: (N & 1) == 1 : Set
1<<i : i = 1 :	0	0	0	0	0	1	0	
N :	1	0	1	0	0	1	0	
(N & (1<<i)) :	0	0	0	0	0	1	0	= 2

	6	5	4	3	2	1	0
N :	1	0	1	0	0	1	0
1 :	0	0	0	0	0	0	1
1<<2 :	0	0	0	0	1	0	0

$= N \& (1<<2) =$ Not set

	6	5	4	3	2	1	0
N :	1	0	1	0	0	1	0
1 :	0	0	0	0	0	0	1
1<<3 :	0	0	0	1	0	0	0
N&(1<<3) :	0	0	0	0	0	0	0

$= N \& (1<<3) = 0$: Not Set

4th bit pos

	6	5	4	3	2	1	0
N :	1	0	1	0	0	1	0
1 :	0	0	0	0	0	0	1
1<<4 :	0	0	1				
N&(1<<4) :	0	0	1	0	0	0	0

$= N \& (1<<4) == 2^4$: Set

2^4

// check Bit using left shift

N & (1<<i) == 0 : Unset Else Set

```
bool checkBit(N, i) {
    if (N & (1<<i) == 0) {
        return false
    }
    else return True
}
```

logical operators

or

return !(N&(1<<i) == 0)

T

return !(T) → F

return !(F) → T

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

i^{th} bit set

// if (N & (1<<i)) == (2^i) = (1<<i)

```
→ if (N & (1<<i) == (1<<i)) {
    i^th bit set
    return True
}
else { return False }
```

return (N&(1<<i)) == (1<<i)

→ 8:58 PM : lom pos break

Count Set Bits : → {Todays Assignment}

→ given N integer calculate how many set bits in N

32 bits : [0 31]

N = 10 : 1 0 1 0 : output 2

N = 27 : 1 1 0 1 1 : 4

Idea : Convert N → Binary & store & count no: of 1's

Iterations : 32

TC: O(1)

```
int countSetBits(int N)
{
    int c = 0
    i = 0; i < 32; i = i+1){
        if (checkBit(N, i)){
            c = c+1
        }
    }
    return c;
}
```

N = 45 :

c = 0

(N&1) == 1 :

5	4	3	2	1	0	
1	0	1	1	0	1	: c++, c = 1
0	1	0	1	1	0	: c = 1
0	0	1	0	1	1	: c++, c = 2
0	0	0	1	0	1	: c++, c = 3
0	0	0	0	1	0	: c = 3
0	0	0	0	0	1	: c++, c = 4
0	0	0	0	0	0	: c = 4

return c

```
int countSetBits(int N)
{
    int c = 0;
    while( N > 0 ){
        if (N&1 == 1){
            c = c+1
        }
        N = N>>1;
        // N = N/2
    }
    return c;
}
```

will it update N value?

TC: $(\log_2 N)$

```
int countSetBits(int N)
    int c=0
    i=0; i<32; i=i+1){
        if(checkBit(N, i)){
            c=c+1
        }
    }
    return c;
}
```

// N=10

Iterations: 32 iterations

N=100

// Example

$N = 2^{32}$

32 iterations

```
int countSetBits(int N)
    int c=0;
    while( N>0 ){
        if(N&1 ==1){
            c=c+1
        }
        N= N>>1;
        // N=N/2
    }
    return c;
}
```

will it update N value?

TC: $(\log_2 N)$ → is better when compared to O(1)

N=10

Iterations:

10: 1010 → 4 iterations

101

10

1

0

→ $\log_2 100$ → 7 iterations

$N=2^{32}$ → $\log_2 N$ → $\log_2 2^{32}$ → 32

32 iterations

30) Given x, y set x^{th} bit & y^{th} bit in {value 0}

$x = 3$, $y = 1$:

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

→ decimal $2^3 + 2^1 = 10$

$x = 5$, $y = 2$:

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

→ $2^5 + 2^2 = 36$

$x = 7$, $y = 3$

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

→ $2^7 + 2^3 = 128 + 8 = 136$

$x = 2$, $y = 6$

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

→ $2^2 + 2^6 = 66$

Set x at y^{th} bit

			2^y												
			y					$x+1$	x			2	1	0	
0	0	0	~~0~~	0	0	0	0	~~0~~	0	0	0	0	0	→	$2^x + 2^y$
0	0	0	1	0	0	0	0	1	0	0	0	0	0		

								2^{x+1}	2^x			2^2	2^1	2^0	

$x = 3$, $y = 3$ } Edge case

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

→ 2^3

$x = 0$, $y = 0$

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

→ $2^0 = 1$

Edge case

```
int setbits(int x, int y) {
    // return (2^x + 2^y)
    return (1<<x + 1<<y)
}
```

→ set x, y in 0

$1<<n = 2^n$

$1<<x = 2^x$

$1<<0 = 1$

$1<<1 = 2$

$1<<2 = 4$

```
int setbits(int x, int y) {
    if (x==y)
        return (1<<x)
    else
        return (1<<x) + (1<<y)
}
```

// Without if/else → ternary operator

If x==y : We need to set only once

If x!=y : Set x^{th}, y^{th} bit

```
int setbits(int x, int y){
    return (1<<x) | (1<<y)
}
```

$x = 6$, $y = 3$

	7	6	5	4	3	2	1	
1<<6 :	0	1	0	0	0	0	0	0
1<<3 :	0	0	0	0	1	0	0	0
72 =	0	1	0	0	1	0	0	0

$x = 5$, $y = 5$

	7	6	5	4	3	2	1	0
1<<5 :	0	0	1	0	0	0	0	0
1<<5 :	0	0	1	0	0	0	0	0
1<<5 =	0	0	1	0	0	0	0	0

// given N, i set i^{th} bit in N

2nd bit not set

	3	2	1	0
N = 10 :	1	0	1	0
i = 2 : 1<<2	0	1	0	0
10 | (1<<2) :	1	1	1	0

2nd bit already set

	4	3	2	1	0
N = 23 :	1	0	1	1	1
i = 2 :	0	0	1	0	0
(N | (1<<i)) =	1	0	1	1	1

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

if i bit is already set in N

N = 21

	4	3	2	1	0
i = 3	1	0	1	0	1
	0	1	0	0	0
(N | (1<<i))	1	1	1	0	1

$21 + 2^3 = 29$

TODO 1) given N, i : unset i^{th} bit in N

N=23 : 1 0 1 1 1 (4 3 2 1 0)

i=2 : 0

10011

N = 21 : 1 0 1 0 1 (4 3 2 1 0)

i=3 : 0

21 = 10101

2) given x, y set x continuous bits & y unset bits

x=3, y=2 : 2^4 2^3 2^2 2^1 2^0 : 1 1 1 0 0 → 28

x=4, y=2 : 2^5 2^4 2^3 2^2 2^1 2^0 : 1 1 1 1 0 0 → 60

x=2, y=4 : 2^5 2^4 2^3 2^2 2^1 2^0 : 1 1 1 1 1 1 → 48

x=3, y=3 : 2^5 2^4 2^3 2^2 2^1 2^0 : 1 1 1 0 0 0 → 32+16+8 → 56

// today: Revision

↳ {datatype range}

Doubts:

S_1 : 1 0 0 1 0 1

S_2 : 1 0 1 0

S_3 : 1 0 1 1 1 1

reverse S_1 & S_2 →

$c =$ 0 0 0 0 0

S_1 : 1 0 1 0 0 1

S_2 : 0 1 0 1 0 1

S_3 : 1 1 1 1 0 1

reverse : 1 0 1 1 1 1

//