S02

Sunday, November 6, 2022 4:01 PM

Continuation of bitwise operators -

Q. `int pow2 (int N) { // 2^N`

≡

}

How to correct this?

a) `long long int x = 1`
`return x << N`

b) `return (long long int) 1 << N`
(Type casting)

c) `return 1 unsigned long long int << N;`

Method 1:

```
int ans = 1;
for(int i=1; i<=N; i++)
  ans = ans * i;
```

N = 25 ✓
N = 43 ×
use long long.
N = 253 ×
N = 60 ✓

Method 2:

```
return 1<<N;
```

N = 63
use unsigned long long

(a << i = a * 2^i)

32-bits

N = 25 ✓
N = 43 ×

Cannot left shift by 43 bits.

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

≡

}

① `return ((N >> i) & 1) == 1;`

a

1 0 1 1 0
3 2 1 0

right shift

a & 1 → 1 odd
a & 1 → 0 even

0th bit after right shift

also the same as

`return ((N >> i) % 2) == 1;`

② `return (N & (1 << i)) != 0;`

↓

(a << i = a * 2^i)

$0 \le N \le 10^9$
$0 \le i \le 30$

N	i	O/P
5 (101) ↓ 0th bit	1	F
5	2	T
12	0	F
12	3	T

(0 → F, 1 → T)

$(a << i = a * 2^i)$

Q. int setBits (int N) {

(count number of 1's in binary form)

```
int c = 0;
for(int i = 0; i <= 30; i++)
    if(checkBit(N, i) == T)
        c++;
return c;
}
```

$0 \le N \le 10^9$

N		o/p
(101)	5	2
(1111)	15	4
(10101)	21	3

$10^9 \approx 2^{30}$

Alter:

```
while(N != 0) {
    if((N & 1) == 1)
        c++;
    N = N >> 1;
}
return c;
```

N		N-1		N & (N-1)	
25	11001	11000	24	11000	24
20	10100	10011	19	10000	16
12	1100	1011	11	1000	8
16	10000	01111	15	00000	0
↳ 2^4 (power of 2)					
42	10101 0	101001	41	101000	40

(N & (N-1)) : unsets the least significant set bit
(makes it 0) (1)

To count set bits:

```
int c = 0;
while(N != 0) {
    N = N & (N-1);
    c++;
}
return c;
```

Q. Check power of 2.

```
if((N & (N-1)) == 0)
        cout << "True"
else
        cout << "False"
```

Q. int create (int x, int y)

(create a number whose x^{th} bit & y^{th} bit are 1s & remaining are 0)

$0 \le x, y \le 25$

x	y	o/p
3	1	1 0 1 0 (10)
5	2	1 0 0 1 0 0 (36)
0	2	1 0 1

```
{
return 2^x + 2^y   (1<<x) | (1<<y)
                   | ← bitwise or
```

↓

```
return (1<<x) | (1<<y)
```

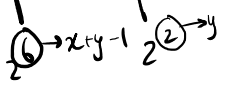
x=2, y=2

4 + 4 = 8 ✗ should be 4.

Q. x 1's y 0's

```
int solve(int x, int y) {
    int v=0;
    for(int i=y; i<=x+y-1; i++)
        v= v+ 2^i
              ↓
           (1<<i)
    return v;
}
```

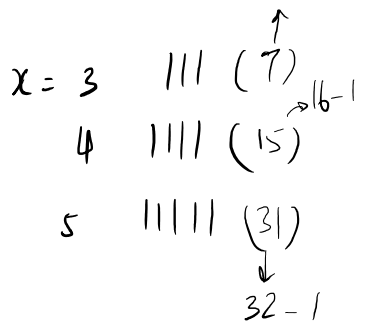
x	y	o/p
2	5	1 1 0 0 0 0 (96)
1	3	1 0 0 0 (8)
5	2	1 1 1 1 1 0 0 (124)



Alter: ① return $((1<<x)-1)<<y$;

② return $(1<<(x+y)-1) \wedge ((1<<y)-1)$;

③ return $(1<<(x+y)) - (1<<y)$;



① is evaluated ✓ & verified.

Then append 0's $<<y$.

$(2^x-1)<<y$

or $((1<<x)-1)<<y$ ✓

③ $(1<<(x+y)) - (1<<y)$

$= 2^{x+y} - 2^y$

$= 2^y(2^x-1)$

same.

③ works too.

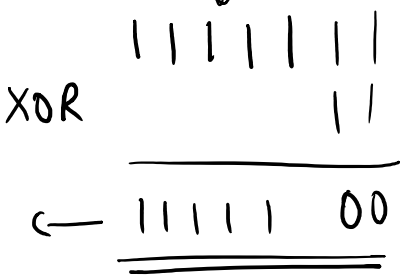
$-(x+y)$ no. of 1s

$= 2^{(x-1)}$

② $a = (1 << (x+y)) - 1 = 2^{x+y} - 1$ → $(x+y)$ no. of 1s

$b = (1 << y) - 1 = 2^{y} - 1$ → y no. of 1s

x = 5, y = 2



x 1s
y 0s
← 11111 00

∴ ② works too

Q. int pwr (int a, int N) { a^N

```
int p = 1;
for (int i = 1; i <= N; i++)
  p = p * a;
return p;
}
```

a	N
2	25 ✓
3	40 ✗ use long long int.
35	2128 ✗

```
a = a % M;
loop...
p = (p * a) % M
return p;
```

for cases where answer > range of primitive data types,

(HR SI Primary Pg. 3)

do not print the exact answer. Use ans % M (only when M is given in the question)

in general, $M = 10^9 + 7$. $x \% M = [0, M-1]$

Modulo Arithmetic

$(10-5)\%7$

$(a*b)\%M = ((a\%M)*(b\%M))\%M$

$(a+b)\%M = ((a\%M)+(b\%M))\%M$

$(a-b)\%M = ((a\%M)-(b\%M)+M)\%M$

$(a/b)\%M = ((a\%M)*(b^{-1}\%M))\%M$

To-do

① Euclid's GCD algo (SI Primary HR)

Inverse Modulo

② Extended Euclid's Algo

LCM & GCD question.

NOTE:

$30005 = 3e^4+5$

double y = .00003 = 3e-5 ↓ 3×10^{-5}

int ar[10^6] = int ar[1e6]

Complexity Analysis of Algorithms

problem → Solution s1, s2, s3

Time complexity (more weightage)

space complexity

Q. Divisors of n.

$N = 10^9$

4. Divisors of n.

```
int divisors(int n) {
    int c=0;
    for(int i=1; i<=n; i++)
    {
      if( n%i == 0)
          c++;
    }
    return c;
}
```

$N = 10^7$

no. of iterations $= 10^9$

no of instructions per iteration = around 4-5 (1) (assume 1)

total no of instructions = 5×10^9 (1×10^9)

time = 1s

$N = 10^{18}$

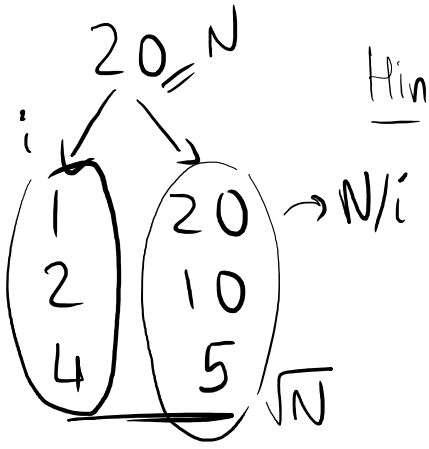
no. of iterations = 10^{18}

no. of instructions/iteration = 4-5 (1)

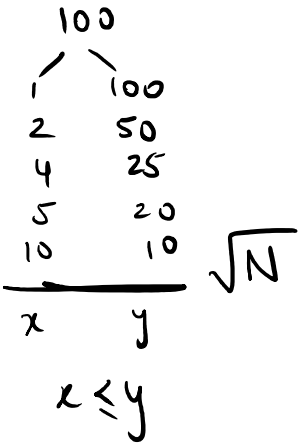
total no. of instructions = 5×10^{18} (1×10^{18})

Time = 10^9 seconds $\approx$ 31.7 years

Divisors



Hint: Use the fact that a number is a divisor of itself to optimize the code & work out a better logic.



$i \leq \frac{N}{i}$

$i^2 \leq N$

$i \leq \sqrt{N}$

Optimized code has reduced iterations!

→ for $N = 10^{18}$, only 10^9 iterations

↳ for $N = 10^{18}$, only 10^9 iterations

⇒ 1 instruction/iteration

⇒ 1×10^9 instructions

⇒ 1 seconds

(1 → assumption for better calculations)