Bit - Manipulation

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

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

MSB LSB

$(105)_{10} = (1101001)_2$

2	105	LSB
2	52	1
2	26	0
2	13	0
2	6	1
2	3	0
2	1	1
	0	1
		MSB

6 5 4 3 2 1 0

$(1101001)_2 = (?)_{10}$

$2^6\times1 + 2^5\times1 + 2^4\times0 + 2^3\times1 + 2^2\times0$
$+ 2^1\times0 + 2^0\times1$

$64+32+0+8+0+0+1$

$96+8+1 = 105$

(*) 1s Compliment of a Number. (32 bits)

$(12)_{10} = (1100)_2$

flip all bits ($1 \to 0$, $0 \to 1$)

0000 0000 0000 0000 0000 0000
0000 1100

=> 1111 1111 1111 1111 1111 1111 1111
0011

① Two's Complement

→ find one's compliment & then add 1 to it.

$(12)_{10} = (0000\ 1100)_2$

1s $(1111\ 0011)$

2's $+1$

$1111\ 0100$

$(12) = (1100)$

$(-12) \Rightarrow (1111\ 0011)_2$ 1s complement

$(0)_{10} = (0000\ 0000)_2$

1s $= (1111\ 1111)$

+0
−0

$+1$

0

$+ 0000 : 0000$

$1000\ 0000 \Rightarrow -0 \Rightarrow -2^7$

$\Rightarrow 2^7$ Signed bit

$+ (0000, 0000)$

(Notation)

(8-bit)

$(+12)_{10} = (0\ 000\ 1100)$

Signed bit

(0 → positive
1 → negative)

1 1111111

$2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$

$2^7 - 1$

32

-2^{31}

$+2^{31} - 1$

0 → (+12)

1 → (−12)

-2^8

-2^7

0000 0000 = 0

1000 0000 = 0 ⇒ -2^7

-2^{31} $\quad$ $2^{31}-1$

⊙ Operators. (bitwise)

① OR (|) if either bit is set final bit will be also set

A	B	A|B
0	1	= (1)
1	0	= (1)
0	0	= (0)
1	1	= (1)

Eg: 12 | 5

$(12) = (1100)_2$

$(5) = (0101)_2$

$(12|5) = (1101)_2$

$= (8+4+0+1) = 13$

② bitwise And (&) if both are set then final bit will be set.

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

12 = (1100)

5 = (0101)

$(12 \& 5) = (0100)_2$

$= (4)_{10}$

③ X-OR (Exclusive OR) (^)

if both bits are same result → 0
otherwise 1.

A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

12 → (1 1 0 0)
5 → (0 1 0 1)
12^5 → $(1\ 0\ 0\ 1)_2$
= $(9)_{10}$

④ Left Shif (<<):

(x << y) : move the bits of x to the left by y' places & apend 0s at the end.

(5 << 3)

(1 0 1)

0 0 0 0 0 1 0 1

0 0 1 0 1 0 0 0

(40)

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

⑤ Right Shift: (>>)

(a >> b) : move the bits of a to right by b places.

(7 >> 2) = $(0\ 0\ 0\ 0\ 0\ 1\ \boxed{1\ 1})_2$
$(0\ 0\ 0\ 0\ 0\ 0\ 0\ 1)_2 = 1$

$(a>>b) = \left\lfloor \frac{a}{2^b} \right\rfloor$

Logical Not (!)

⊙ bitwise Not operation ($\sim$)

x

$\boxed{\sim x = -x-1}$

$\sim 5 = -5-1 = -6$

⊙ Check i^{th} bit is set or not?

a i

$\boxed{((1<<i)\ \&\ a)\ != 0}$

i^{th} bit set

$((a>>i)\ \&\ 1) == 1$

i^{th} bit set.

17

13

3 2 1 0

1 1 0 1

0 1 0 0

(0 1 0 0)

2^{nd}

$1<<2$

0 0 0 1

0 1 0 0

1 1 0 1 $>>2$

0 0 0 1

0 0 0 0 1

1

① Count set bits in n :

Integer 32

```
cnt=0;
for (int i=31 ; i>=0; i--)
    if ((n>>i)&1) cnt++;
return cnt;
```

$O(32)$

$O(\text{Max No. of bits})$

②

13

1 1 0 1

```
while (n>0){
    if(n&1) cnt++;
    n>>=1;
}
```

$O(\log n)$

$\boxed{\log_2 n}$

⊙ Brian Kernighan Algorithm : $O(1)$

worst : $O(\log N)$.

$O(\text{set bits})$

②

$n = n \,\&\, (n-1)$

5 4 3 2 1 0				
1 0 0 1 0 0	⇒ 36			
1 0 0 0 1 1	35			
n = 1 0 0 0 0 0	= 32			
0 1 1 1 1 1	31			
0 0 0 0 0 0	= 0			

111111

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

$O(\text{setbits})$

avg Case $O(1)$

worst Case $O(\log n)$

$\boxed{n \& (n-1)}$

$n \rightarrow$ LSB (set) unset

$$
\begin{array}{rccccccl}
 & 4 & 3 & 2 & 1 & 0 & & \\
 & 1 & 0 & 1 & 0 & 1 & & 21 \\
\& & 1 & 0 & 1 & 0 & 0 & & \\
\hline
 & 1 & 0 & 1 & 0 & 0 & = & 20 \\
 & 1 & 0 & 0 & 1 & 1 & & 19 \\
\hline
 & 1 & 0 & 0 & 0 & 0 & & \\
 & 0 & 1 & 1 & 1 & 1 & & \\
\hline
 & & & & & & = & 0
\end{array}
$$

XOR- Properties:

$$a \wedge 0 = a$$
$$a \wedge a = 0$$
$$a \wedge b = b \wedge a$$
$$a \wedge (b \wedge c) = (a \wedge b) \wedge c$$
$$a \wedge a \wedge a = a$$

$$\boxed{a+b = (a \wedge b) + 2(a \& b)}$$

flip i^{th} bit

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

$$\boxed{a = (1 << i) \wedge (a)}$$

set i^{th} bit:

$$a = ((1 << i) | a)$$

① Check n is power of 2 or not.

$((n \& (n-1)) == 0)$ return 2;

$16+8+2+1$

$\boxed{① \text{ Power of } 3:}$

27 $\boxed{11011}$

O(1)

$\alpha > K$

3^K

$(3^\alpha \% 3^K == 0)$

10^9

10^9

$\alpha = 1;$

$3^\alpha > 10^9$

while (α < max){

α *= 3

}

K

log max

$3^\alpha \% k == 0$ (pow of 3)

else (not).

~~X = (11001110032~~

MAX 18 + 0 =

17 = 18

3^α

$3^\alpha = 18$

$k \approx 3^x$ $(27) = 3^3$

$$\boxed{k = 3^x}$$

```
while(k % 3 == 0)
    k /= 3;
return k == 1;
```

$\log(n)$

(3^k) $3^\alpha > 3^k$

$\alpha >= k$

$6^\alpha \% k$ $(3^\alpha \% 3^k == 0)$

10^9

y k

return $(y \% k == 0)$

① Majority $\lfloor \frac{n}{2} \rfloor$

[1, 2, 3, 7, 7, 7, 7]

$\frac{7}{2} = 3$

0 0 0 1 2

3	2	1	0
0	5	5	6

1 1 1

$\frac{n}{2}$

$= 7$

1. Sorting
2. Hash
3. Moore's Voting
4. Bits

i^{th}

i

X

ans |= (1<<i)

$\boxed{X > \frac{N}{2}}$

Majority elem

Non-Majority:

$= \frac{N}{2}$ $< \frac{N}{2}$ $> \frac{N}{2}$

[2 1 2 1 2 2 2]

cnt = 0
cand = -1

cnt == 0
cand: a[i]

if(cand == a[i])
cnt++;
else cnt--;

M N M N M N M

M M M M N N N

$\frac{N}{2}$

```
class Solution {
public:
    int majorityElement(vector<int>& nums) {
        int ans  = 0;
        for(int i = 31; i >= 0 ; i--){
            int cnt = 0;
            for(int x: nums){
                if((x >> i)&1)cnt++;
            }
            if(cnt > (int)nums.size() / 2){
                ans |= (1<<i);
            }
        }
        return ans;
    }
};
```

O(32 · N)

O(No. of bits × N)

1. Missing Number: (XOR).

Divide Integers

Medium 104 154

Asked In: Microsoft Amazon

Divide two integers without using multiplication, division and mod operator.

Return the floor of the result of the division.

Example:

```
5 / 2 = 2
```

Also, consider if there can be overflow cases. For overflow case, return INT_MAX.

Note: INT_MAX = 2^31 - 1

X

$\lfloor \frac{17}{3} \rfloor = 5$

$\lfloor 17/3 \rfloor = 5$

$a << b = a * 2^b$

$17 = 3*5 + 2$

$= 3*(4+1) + 2$

$= [3*4 + 3\times 1] + 2$

$= [3 * (pow(2,2)) + 3 \times pow(2,0)] + 2$

$= [(3<<2) + (3<<0)] + 2$

$= \{(1<<2) + (1<<0)\} = 5$

a b temp = 0 ans = 0 -ve

$\left(\frac{a}{b}\right) = \left(\frac{a}{b}\right)$

```
for(int i=30; i>=0; i--){
    if((temp + (b<<i)) <= a){
        ans |= (1<<i);
        temp += (b<<i);
    }
}
return ans;
```

(a<0 && b>=0) ||

(b<0 && a>=0)

sign = -1;

ans 3 2 1 0

0 1 0 1 = 5

17 / 3

temp = 12

$12 + (3<<0)$ 15

$(3<<x)$

2 $3<<1$ 3

12

18

$17/3 = 5$

$17 = 3\cdot 5 + 2$

$3 * 5$

$= 3\cdot(4+1)$

$= 3(4) + 3(1)$

$= 3(1<<2) + 3(1<<0)$

$17 = \boxed{(3<<2) + (3<<0)} + 2$

$(1<<2) + (1<<0)$

$\sum (3<<x)$

$(6<<x) + (6<<y)$

$(1<<x) + (1<<y)$

```
ans=0, temp=0
for( int i=30; i>=0; i--)
while( temp+(6<<i) <= a)
{
    temp += (6<<i);
    ans += (1<<i);
}
return sign * ans;
```

0

ans = 2 14

30 29 28 27 ... + 7

21

$23/7 =$

0 $21 <= 23$

$14 < 23$

$0 + (7<<x) <= 23$

$(1<<x)$

$7\cdot(2)$

$\boxed{7 * (1<<1)}$

temp = 21

ans = 3

temp = 21

2^3

2^1

2

Programming / Bit Manipulation / Count Total Set Bits

Count Total Set Bits

Hard 237 20

Asked In: Amazon

Problem Description

Given a positive integer A, the task is to count the total number of set bits in the binary representation of all the numbers from 1 to A.

Return the count modulo $10^9 + 7$.

Problem Constraints

$1 <= A <= 10^9$

Input Format

First and only argument is an integer A.

0000 = 0
0001 = 1
0010 = 1
0011 = 2
0100 = 1
0101 = 2
0110 = 2
0111 = 3
1000 = 1
13

$N = 8$

$N = 2^i$

$N = 8 = 2^3$

$\frac{N}{2} + \frac{N}{2} + \frac{N}{2}$

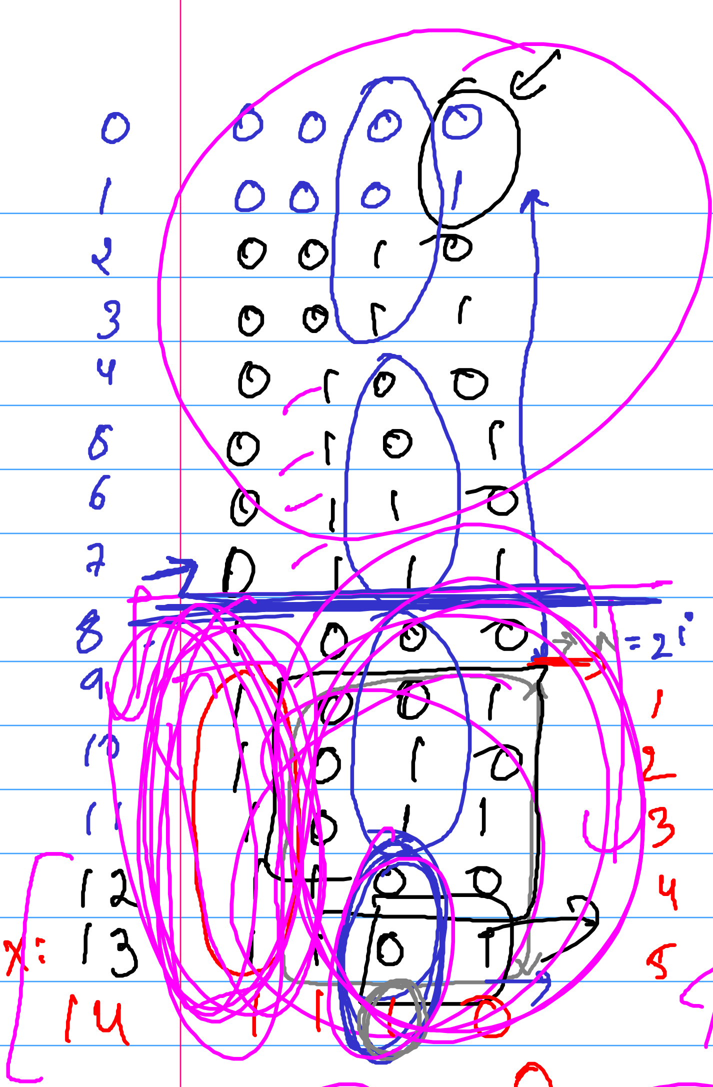
$\frac{N}{2} + \left(\frac{N}{2^2}\right) \cdot 2^1 + \left(\frac{N}{2^3}\right) \cdot 2^2$

$= \frac{N}{2} + \frac{N}{2} + \frac{N}{2}$

$2^i - 1 = \left(\frac{N}{2}\right) * i$

$= \left(\frac{N}{2}\right) * i + 1$

$2^i = \left(\frac{N}{2}\right) * i + 1$



$\left(\frac{N}{2}\right) \times 1 =$

$\left(\frac{N}{4}\right) \times 2$

x

$$\boxed{\left(\frac{N}{2}\right) * i + 1 + \left(x - 2^i\right) + \text{Setbit}\left(x - 2^i\right)}$$

$\log N + \log$

$3 + 2 = 1 \quad 0$

$\frac{\cancel{x}}{2}$

$\frac{4}{2} = 2$

$14 = \left(\frac{14}{8}\right) \times 4 = 4$

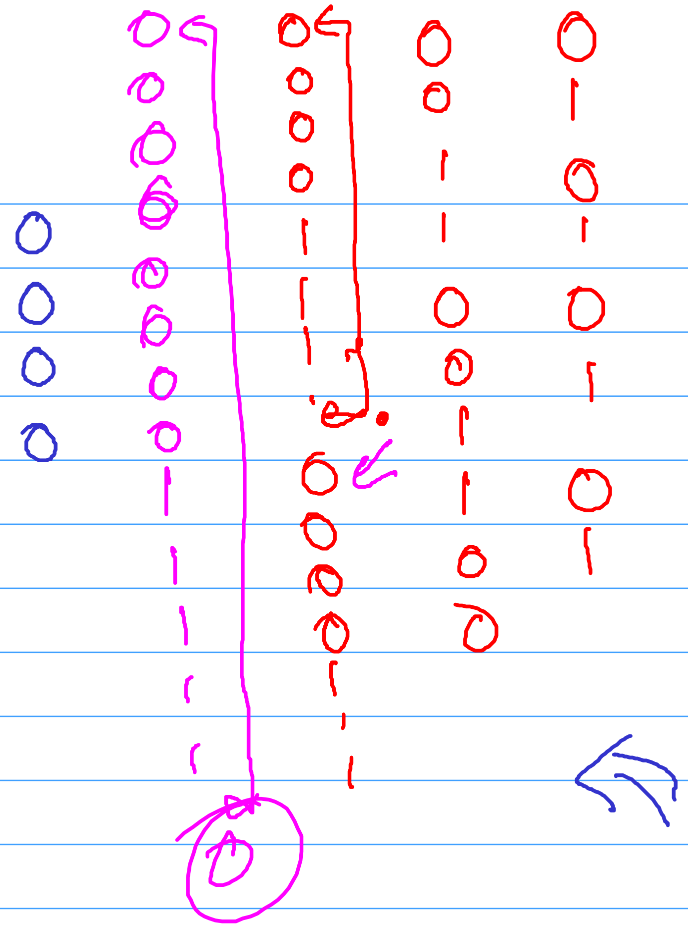
$14 - 8 = \cancel{6} - \left(\frac{8}{4}\right) = 4$

$14 - 7 = 7$

$\cancel{6} - 4 = 2$

7

$\left(\frac{\cancel{x}}{2}\right) = 0$



x

$$Rem \le \frac{x}{2}$$

$$\left(Rem - \frac{x}{2}\right)$$

1

16

$\frac{6}{10} \times 8 = 0$

grp = k

$\frac{k}{2} \Rightarrow 0$

$x \% 2$

$\frac{k}{2} = 1$

2°

	16	8	4	2
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0

$\frac{6}{2} = 3 \times 1 = 3$

$7 \% 16 = 7 - \frac{16}{2}\ 8$

$\max(0, -1)$; 3

$\frac{6}{4} = 1 \times 2 = 2$

$\frac{6}{8} = 0 \times 4 = 0$

$(6+1) \% 4 = 3$

$3 - \frac{4}{2} = 1$

8

$7 \% 8 = 7 - \left(\frac{8}{2}\right) = 3$

1

3

$x - \frac{k}{2}$

1st

rem = (x+1) % grpsz ;

rem_set_bit = max(0, rem − gp/2);

32

0 — 31

$(L - R) \rightarrow (1 - L{-}1) = X$

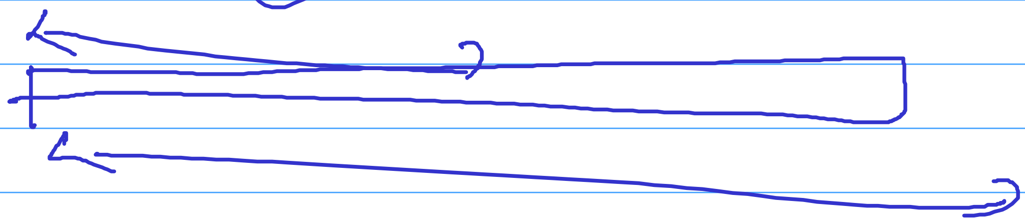
$\rightarrow (1 - R) = Y$

$Y - X$

$(L+1, R)$

L − R

L



Nth palindromic binary

Programming / Bit Manipulation / Palindromic Binary Representation

Palindromic Binary Representation

Hard 123 10

Asked In: Amazon

Problem Description

Given an integer **A** find the **A**th number whose binary representation is a palindrome.

NOTE:

- Consider the 1st number whose binary representation is palindrome as 1, instead of 0
- Do not consider the leading zeros, while considering the binary representation.

Problem Constraints

1 <= A <= $2*10^4$ $= 2 \times 10^4$.

	bin	Dec
1 →	1	1
2 →	11	3
3 →	101	5
4 →	111	7
5 →	1001	9
6 →	1111	15
7 →	10001	17
8 →	10101	21.
9 →	11011	27.

5 digit

1 x x . 1

2^2

len = 1 binary.

1	1	} 1
2	1 1	} 2
3	1 0 1 1 1 1	} 2
4	1 0 0 1 1 1 1 1	} 2
5	1 0 0 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1	} 4
6	1 0 0 0 0 1 1 0 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1	} 4

l = 1 = 1
l = 2 = 1 = 2
l = 3 = 2 = 4
l = 4 = 2 ⇒ 6
l = 5 = 4 ⇒ 10
l = 6 ⇒ 4 ⇒ 14

14

$4 \quad 2^{4/2} = 2^2 = 4$

⇐ 10

l = 6

1 1 1 1 1 1

$2^{5/2} \quad 2^2 = 4$

3 = 11

$2^{\frac{(3-1)}{2}}$

sum = 4 = 10

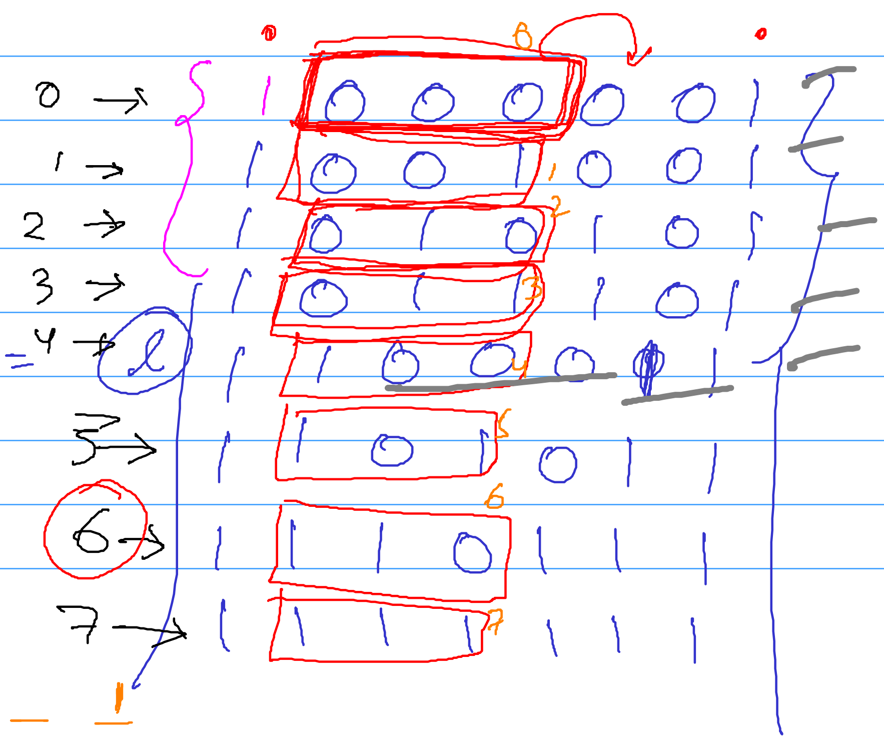
A

14 − 4 10 = 4 = 3rd

$2^{\frac{7-1}{2}} = 2^{6/2} = 2^3$

7

7



n^{th} = l

8

	Count	Cumulative
L=1	1	1
L=2	1	2
L=3	2	4
L=4	2	6
L=5	4	10
L=6	4	14
$\boxed{L=7}$	8	22

19

22

22 - 8 = 14

19 - 14 = 5 → 4 100

1 1 0 $\boxed{0}$ 0 1 1

l = 7

7

l =

1 0 0 0 0 0 0

1 0 0 0 0 0

$\boxed{1 << (L-1)}$

offset = 4 = 100

100 << $\frac{len}{2}$

4 << 3

$\boxed{1\ 1\ 0\ 0}$ ~~0~~ ~~0~~ ~~0~~

=>

1 1 0 0 0 0 0

1 0 0 0 0 0

1 1 ~~0~~ ~~0~~

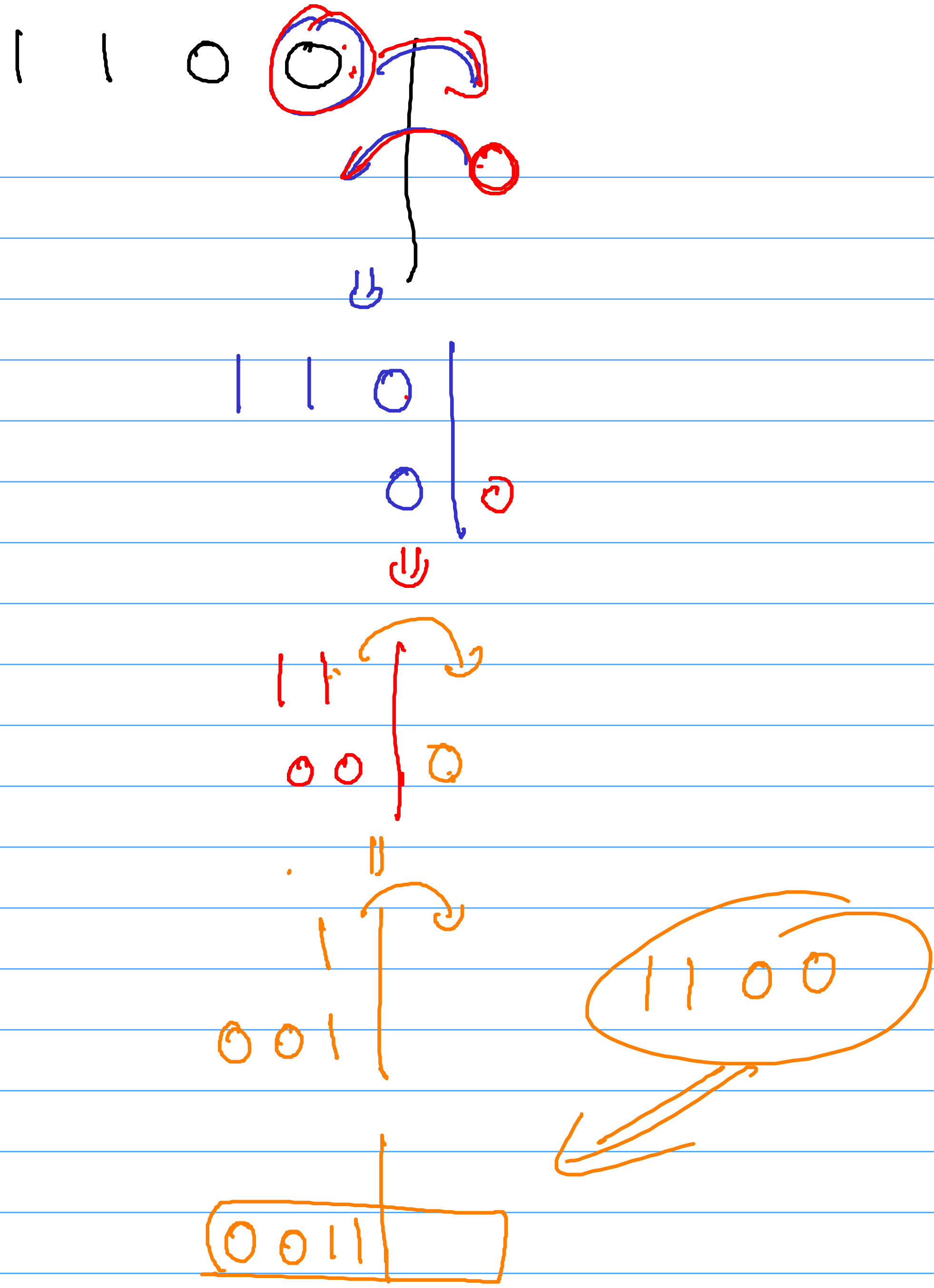
1 1 ~~0~~ ~~0~~ 0

0

0 0 1 1 0 0 0 0

ans: ~~0 0 0 0~~ 0 ~~0~~ 0 0 1 1

0 0 1 1



(1100) => stoi(s, 2);

stoi(1101)

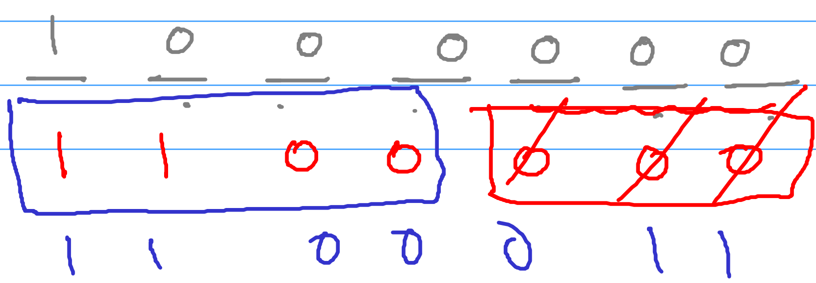
1101

stoi(1101, 2, 0)

13

$l = 7$

4



offset : 100

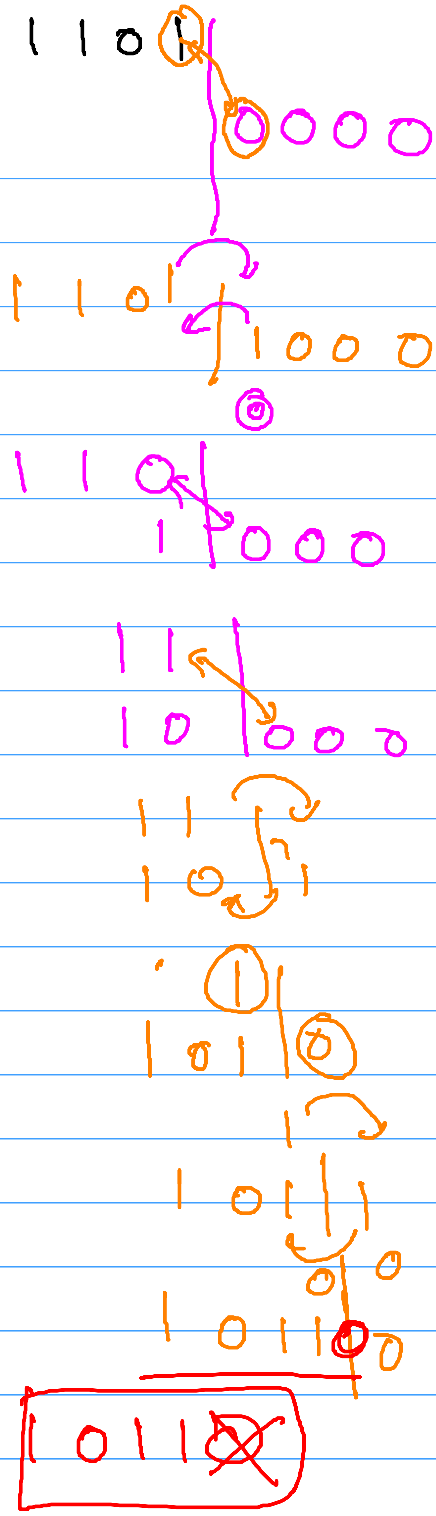
100000

```cpp
int getRev(int n){
    int rev = 0;
    while(n > 0){
        int low_bit = (n & 1);
        rev |= (low_bit);
        rev <<= 1;
        n >>= 1;
    }
}
```

```cpp
int getRev(int n){
    int rev = 0;
    while(n > 0){
        int low_bit = (n & 1);
        rev |= (low_bit);
        rev <<= 1;
        n >>= 1;
    }
    rev >>= 1;
    return rev;
}
int Solution::solve(int A) {
    int len = 1;
    int count = 1;
    while(count < A){
        len++;
        count += (1<<((len - 1) / 2));
    }
    count -= (1<<((len - 1) / 2));
    int offset = A - count - 1;
    int ans = (1<<(len - 1));
    ans |= (offset << (len / 2));
    int val = (ans >> (len / 2));
    int rev = getRev(val);
    return (ans | rev);
}
```

O(log n)

O(log A)



$a < b < c$

$a \wedge b < a \wedge c$

Min sorted $a \wedge b$

Max Trie

Programming / Bit Manipulation / Single Number II

Single Number II

Medium 184 10

Asked In: Google Amazon

Given an array of integers, every element appears thrice except for one which occurs once.

Find that element which does not appear thrice.

Note: Your algorithm should have a linear runtime complexity.

Could you implement it without using extra memory?

Input Format:

```
First and only argument of input contains an integer array A
```

Output Format:

```
return a single integer.
```

Constraints:

```
2 <= N <= 5 000 000
0 <= A[i] <= INT_MAX
```

ith → x

3x + 1

3 time   single

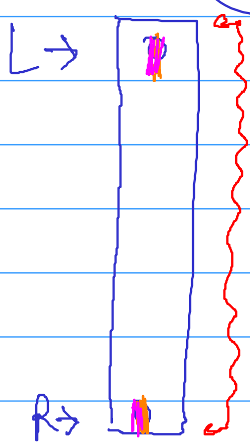
3x ⇒ 3 time

(1) (3x + 1) % 3 == 1

3 time   single.

bit wise

(1) L - R   And   ith



ith

L = 0 & R = 0

ans = (

ith → 0

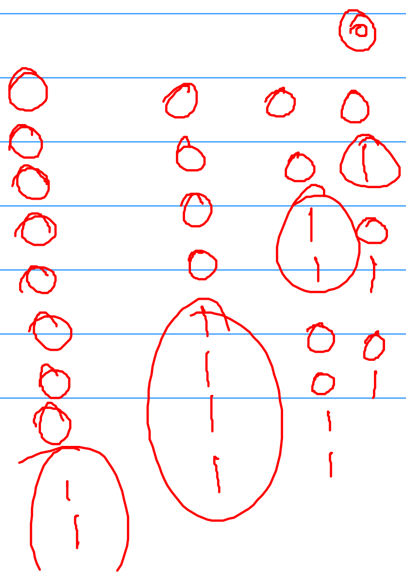
ans = 1 ith may be 1

ith → ((L >> i) & 1)   ((R >> i) & 1)

ith bit → (2^i)

L — R

(R - L + 1) <= 2^i



$0 \le L \le R \le 10^8$ $= 64$

2^0 <= Total

$L = 4$

$R = 7$

1	0	1	→ L
1	1	0	
1	1	1	→ R
		1	4

4	3	2	1	0	
1	1	1	0	0	= 28
1	0	1	0	1	= 29
1	0	1	1	0	= 30
1	0	1	1	1	= 31
	0	1			
	0				
	0				
	0				
	0				

4

$2^2 = 4$

$2^3 = 8$

4

$L \to i^{th}$ & $R \to i^{th}$

& $(R-L+1) <= (2^i)$

ans1 = (1<<i);

2^0

0	0]
0	1]
1	0
1	1]

$i^{th} \to \left\{ \begin{array}{l} 2^i \to 0 \\ 2^i \to 1 \end{array} \right\}$

$L - R \le 2^i$

$L \to 1$

$R \to 1$

⇒ L – R bit-wise And :-

$a_i \le 10^9$: 32

a_0 a_1 a_2 a_3 ..... a_{n-1}

L R

0 → 1

1 →

2 → R-L+1

3 →

0 →
1 →
2 →
3 →
4 →
5 →

L – R

$0 \le L \le R \le 10^{18}$

⇓

~~log(N)~~

L	a_0
$L+1$	a_1
$L+2$	a_2
$L+3$	a_3
$L+4$	a_4
⋮	
R	

$\le N \le 10^5$

$O(\log N)$

R → R (L-1)

L-1

L – R = [..]

R-L+1

⊙ MAX SUB-ARRAY XOR.

⇓

MAX-PAIR → O(N·length of trie)

O(N log N)

⇓

bits

MAX-PAIR

[2 3 4 7 1 3 9]

[0 [2, 2^3, 2^3^4, 2^3^4^7, 2^3^4^1^7, 2^3^4^7^1^3, 2^3^4^7^1^3^9]

⊙ A - N

⇓

a_i

N = 6

0 {1^1, 2^2, 3^3, 4^4, 5^5, 6^6}

⊙ a_0 a_1 a_2 a_3

a a b b c c d d e f

$e \wedge f = \boxed{X} \Rightarrow$ ith bit set

i^{th} bit = e

i^{th} unset xor = f

$1 \wedge 1 = 0$

$0 \wedge 0 = 0$

$1 \wedge 0 \quad 1$

$0 \wedge 1 = 1$

$e \wedge f = x$ $\quad$ i^{th} be set

$e = 1 \qquad f = 0$

$f = 0 \qquad e = 1$

i^{th} bit : $((e) \wedge b \wedge \cancel{b} \wedge \cancel{c} \wedge c)$

0 $\quad$ 1 0 $\boxed{1\ 0}$ $\qquad$ k = 2

1 $\boxed{0\ 0}$ 1

1 1 1 1

$\boxed{1\ 1\ 0}$ $\qquad$ k = 2

0 $\boxed{1\ 0}$

$\boxed{0\ 0\ 1}$ =

$\boxed{1\ 1\ 1\ 0}$ $\qquad$ k = 2

0 $\quad$ 1

[0,0,0,1,0,1,1,0] $\qquad$ k = 3

(1, 1, 0)

Minimum Subset OR. (having exactly k element).

4

1 2 3 4 4 4 4 5 6 7

k = 3

7

1 2 3 4 6 6 4 4 4

k = 4

0 0 1
0 1 0
0 1 1
1 0 0 0
1 1 1 = 7

6

$A_i \cap A_i \cap A_i \cap A_i$ = A_i

3 3 4 3 3 ⇒ A_i

⇒ 3

4

a_i a_i a_i a_i 4 a_i

a_1 a_2 a_3

1100
0

a b

c d = X

3

bitwise (And) of arr.

A_i = X

X

X

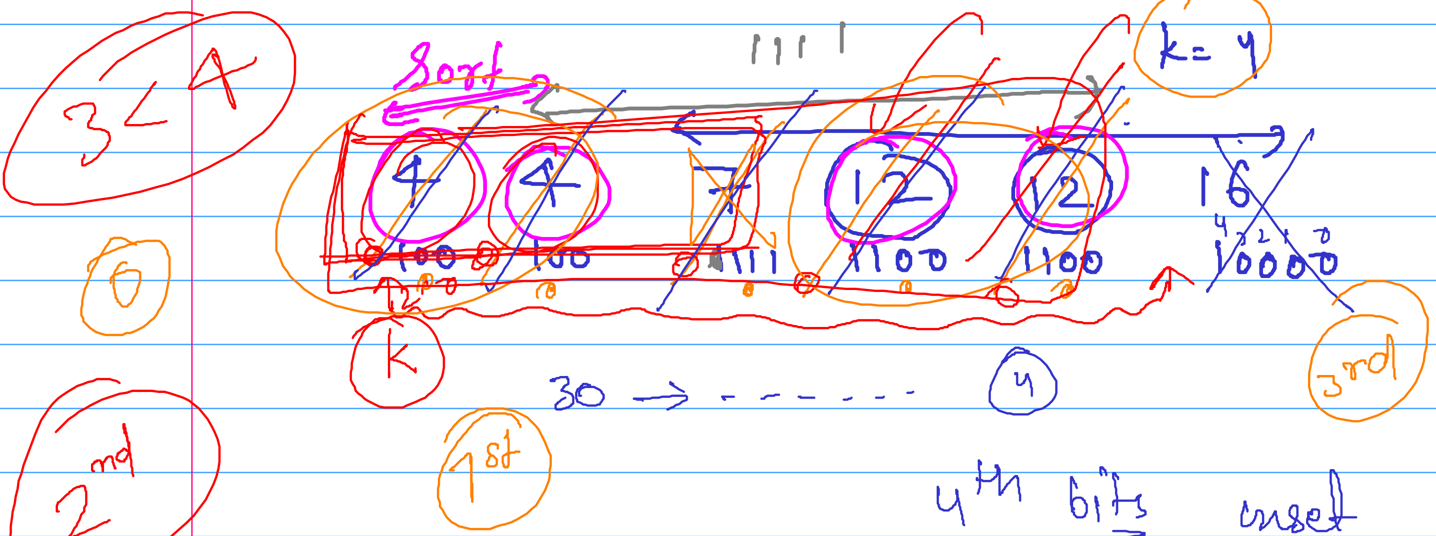
k len

a_0 a_1 a_2 ... a_{k-1} = X

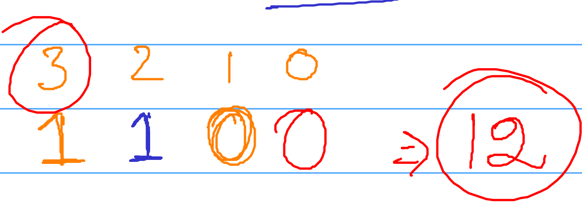
k−1

k−1

① min OR (k elements) :-



4^{th} bits unset



i^{th} bit unset

ans i^{th} bit - unset

i^{th} bit

m' edges

m' edges

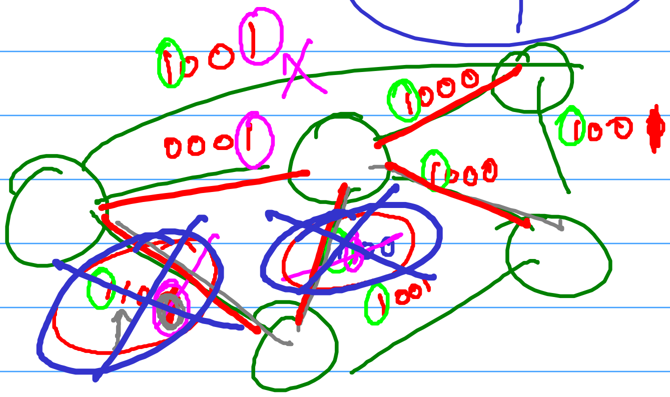
```
for ( int i = 30; i >= 0; i-- )
    ~~vector<pair<int, int>> M;~~   New Edges
    for (auto &e: edges){
        if ((e.w >> i) & 1) continue
        New-edge.pb({e});
```

}

if (Connect (NewEdges))
Edge = NewEdges ;

}.

Connected → Graph → dfs → 1 call



vis →