Revision - Bit Manipulation

Number System

It is a way to define a variable to fixed amount of quantity. Number of unique symbols in a number system defines that number system. It is the base of the number system.

Common Number Systems

Decimal Number System

Consists of digits from 0 - 9, i.e., 10 unique symbols. These are: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.

Binary Number System

Consists of digits 0 - 1., 2 unique symbols. These are: 0 and 1.

Binary To Decimal Conversion

1. Suppose the binary number is $(1101)_2$.
2. The indexing for binary number is done from right to left. Indexing for $(1101)_2$ is depicted below:

1	**1**	**0**	**1**
3	2	1	0

3. 0 is at index 1 and rest all 1s at index 0, 2 and 3.
4. Now, sum the values of $A_i * 2^i$, where A_i is the i^{th} bit.
5. $(1101)_2 = 1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3 = 13$

Decimal To Binary Conversion

1. Suppose the number is 13
2. Calculate the remainders in the following fashion:

2	13	1
2	6	0
2	3	1
2	1	

3. Now, write the remainders from down to top.
4. 13 = $(1101)_2$

Bitwise Operators

1. These works directly on bits.
2. These take one or more bits as input and produce a bit as an output.

NOT(~) operator

Simply flips the bits.
~0 = 1
~1 = 0

AND (&), OR (|) and XOR(^) operators

Takes two bits as input and produces an output according to the following table.

A	B	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Properties:

1. A&0 = 0
2. A&A = A
3. A|0 = A
4. A|A = A
5. A^0 = A
6. A^A = 0
7. Least significant bit of odd numbers are 1 and for even numbers it is 0.

8. Cumulative:
    - A&B = B&A
    - A|B = B|A
    - A^B = B^A
9. Associative:
    - (A&B)&C = A&(B&C)
    - (A|B)|C = A|(B|C)
    - (A ^ B)^C = A ^ (B^C)

Shift Operators

Right Shift Operator

Remove the least significant bit and shift all higher bits by 1 to lower index. Also, put a 0 at the most significant bit.

Example:

A	0	0	0	1	0	1	0	0
A>>1	0	0	0	1	0	1	0	0
A>>2	0	0	0	0	1	0	1	0
A>>3	0	0	0	0	0	1	0	1
A>>4	0	0	0	0	0	0	1	0
A>>5	0	0	0	0	0	0	0	1
A>>6	0	0	0	0	0	0	0	0

Left Shift Operator

Remove the most significant bit and shift all lower bits by 1 to upper index. Also, put a 0 at the least significant bit.

Example:

A	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**
A<<1	0	0	1	0	1	0	0	0
A<<2	0	1	0	1	0	0	0	0
A<<3	1	0	1	0	0	0	0	0
A<<4	0	1	0	0	0	0	0	0
A<<5	1	0	0	0	0	0	0	0
A<<6	0	0	0	0	0	0	0	0

Note: A<<i = A*2^i

Question - 1

Given an integer N. Check if the i^{th} is set to 1.

Example: N = 20, i = 2
Let’s represent N in binary number system.
20 = $(10100)_2$

We can see the the bit at 2^{nd} index is set. So, we will return true.

Solution:
We can use bit masking to solve this problem. These are:

1. Right shift the number by i.
2. Now, the bit at 0^{th} is the required answer.
3. We can simply find the 0^{th} bit using AND operation with 1.

N = $(10100)_2$
N>>2 = $(00101)_2$
(N>>2)&1 = $(1)_2$

```
function(N, K):
    if (N>>K)&1 > 0:
        return true
    else:
        return false
```

- Time complexity: O(1)
- Space complexity: O(1)

Note:

1. Set ith bit of A: A = A | (1<<i)
2. Unset ith bit of A: A = A & ((1<<max_bit) - 1) ^ (1<<i)
3. Toggle ith bit of A: A = A ^ (1<<i)

Range

Unsigned Integers

For N bit number:
Minimum value = 0
Maximum value = 2^N - 1

Consider a 8 bit number, i.e., N = 32
Minimum value = $(00000000)_2$ = 0
Maximum value = $(11111111)_2$ = 2^{32} - 1

Signed Integers

Most significant bit is assigned for sign, i.e., either positive or negative.

For N bit number:
Minimum value = $-2^{N - 1}$
Maximum value = $2^{N - 1}$ - 1

Consider a 8 bit number i.e., N = 32
Minimum value = -2^{31}
Maximum value = 2^{31} - 1

Question - 2

Given an array A where every element occurs even number of times except one number which occurs odd number of times. Find that number

Example:
A = [3, 1, 3, 3, 4, 7, 4, 3, 7]
Here all numbers except 1, occurs even number of times. So, our answer should be 1.

Solution:

Observation:
We can use the xor property stating that:

- A^A = 0

If we take xor of all the elements in the array, then the resulting number which we get would be our answer.
Since, all elements except one element occurs even number of every, all other number would vanish because of the property. What remains is the number which occured odd number of times.

Pseudo Code:

```
function(A):
    ans = 0
    for element in A:
        ans = ans ^ element
    return ans
```

- Time complexity: O(N)
- Space complexity: O(1)

Question - 3

Given an array A, all numbers occur thrice except one number which occurs once. Find the unique number.

Example:
A = [1, 12, 1, 12, 4, 3, 12, 1, 3, 3]
Here, only 4 occurs once, rest all elements occur thrice. So, our answer should be 4.

Solution:

Observation:
We can work on the count of set bits on each indices, to find if the bit is set in the unique element.
If count = 3 * x, then the bit is unset
If count = 3 * x + 1, then the bit is set
count = 3 * x + 2 is not possible.

Pseudo Code:

```
function(A):
    ans = 0
    for bit from 0 to 31:
        count = 0
        for element in A:
            if ((element>>bit)&1) > 0:
                count = count + 1
        if count%3 == 1:
            ans = ans|(1<<bit)
```

- Time complexity: O(N * log(max(A)))
- Space complexity: O(1)

Question - 4

Given an integer array A of size N, in which every element occurs exactly twice except 2 elements. Find those two elements.

Solution:

Brute force: For every elment check if it is unique or not.

Pseudo Code:

```
function(A):
    N = length of A
    for i from 0 to N - 1:
        flag = 1
        for j from i + 1 to N - 1:
            if A[i] is equal to A[j]:
                A[i] is not unique
                set flag = 0
                break loop

        if flag == 1:
            A[i] is an answer
```

Optimization:
Suppose X and Y are the unique elements. We can think about using properties of XOR. One such property is A ^ A = 0. So, if we xor all the values of the array A, we will have the xor of the element which occurs once. Let it be Z.

Now, lets try to separate the array into two arrays such that X and Y occurs in different array. For this we will find a set bit in Z and separate all the element of A based on this set bit. Now, these two subarray will have a single unique element. We can easily solve these sub problem using xor operation.

Pseudo Code:

```
function(A):
    ini_xor = 0
    for element in A:
        ini_xor = ini_xor ^ element

    set_bit = -1

    find set bit in ini_xor
    for i from 0 to MAX_BIT:
        if (ini_xor>>i)&1 > 0:
            ith bit is set
            set_bit = i
            break from loop

    xor_1 = 0, xor_2 = 0

    for element in A:
        if set_bit is set in element:
            xor_1 = xor_1 ^ element
        else
            xor_2 = xor_2 ^ element

    return (xor_1, xor_2)
```

- Time complexity: O(N)
- Space complexity: O(N)