DAY 9

INTERVIEW BIT PROBLEMS :

1. Valid Number
Validate if a given string is numeric.
Examples:
"0" => true
" 0.1 " => true
"abc" => false
"1 a" => false
"2e10" => true
Return 0 / 1 (0 for false, 1 for true) for this problem

Is 1u (which may be a representation for unsigned integers valid?
For this problem, no.

Is 0.1e10 valid?
Yes

-01.1e-10?
Yes

Hexadecimal numbers like 0xFF?
Not for the purpose of this problem

3. (. not followed by a digit)?
No

Can exponent have decimal numbers? 3e0.1?
Not for this problem.

Is 1f (floating point number with f as prefix) valid?
Not for this problem.

How about 1000LL or 1000L (C++ representation for long and long long numbers)?
Not for this problem.

How about integers preceded by 00 or 0? like 008?
Yes for this problem

CODE :

PYTHON

```
class Solution:
    # @param A : string
    # @return an integer
    def isNumber(self, A):
```

```
A = A.strip()
n=len(A)
if n==0:
    return 0
if A[0]=='+' or A[0]=='-':
    A=A[1:]
    n-=1
    if n==0:
        return 0
i=0
dot=False
e_p=False
while i<n:
    if A[i]>='0' and A[i]<='9':
        i+=1
        continue
    if A[i]=='.':
        if dot:
            return 0
        dot=True
        i+=1
        if i>=n:
            return 0
        elif A[i]=='e':
            return 0
    elif A[i]=='e':
        if e_p:
            return 0
        e_p=True
        dot=True
        i+=1
        if i<n and (A[i]=='-' or A[i]=='+'):
            i+=1
    else:
        return 0
return 1
```

2. Valid Ip Addresses

Given a string containing only digits, restore it by returning all possible valid IP address combinations.
A valid IP address must be in the form of A.B.C.D, where A,B,C and D are numbers from 0-255. The numbers cannot be 0 prefixed unless they are 0.
Example:
Given "25525511135",
return ["255.255.11.135", "255.255.111.35"]. **(Make sure the returned strings are sorted in order)**

CODE:

PYTHON

```
class Solution:
    # @param A : string
    # @return a list of strings
    def isvalid(self,s):
        s=s.split(".")
        for i in s:
            if len(i)>3 or int(i)<0 or int(i)>255:
                return False
            if len(i)>1 and int(i)==0:   #for i=000 or i=00
                return False
            if len(i)>1 and int(i)>0 and i[0]=="0":
                return False
        return True

    def restoreIpAddresses(self, A):
        n=len(A)
        if n>12 or n<4:
            return []
        l=[]
        s=A
        #loop for inserting 3 "." in the string.
        for i in range(1,n-2):
            for j in range(i+1,n-1):
                for k in range(j+1,n):
                    s=s[:i]+"."+s[i:]
                    s=s[:j+1]+"."+s[j+1:]
                    s=s[:k+2]+"."+s[k+2:]
                    if self.isvalid(s):
                        l.append(s)
                    s=A
         return l
```

C++

```
bool isValid(string s) {
    if (s.size() > 1 && s[0]   ==   '0')
        return false;
    if (stoi(s) <= 255 && stoi(s) >= 0)
        return true;
    else
        return false;
}
vector<string> Solution::restoreIpAddresses(string s) {
    vector<string> ans;
    if (s.size() > 12 || s.size() < 4)
        return ans;

    for (int i = 1; i < 4; i++) {
        string first   =   s.substr(0, i);
```

```
            if (!isValid(first))
                continue;
            for (int j = 1; i + j < s.size() && j < 4; j++) {
                string second = s.substr(i, j);
                if (!isValid(second))
                    continue;
                for (int k = 1; i + j + k < s.size() && k < 4; k++) {
                    string third = s.substr(i + j, k);
                    string fourth = s.substr(i + j + k);
                    if (isValid(third) && isValid(fourth)) {
                        string current = first + "." + second + "." + third + "." + fourth;
                        ans.push_back(current);
                    }
                }
            }
        }

        return ans;
}
```

3. Number of 1 Bits

Write a function that takes an unsigned integer and returns the number of 1 bits it has.
Example:
The 32-bit integer 11 has binary representation
00000000000000000000000000001011
so the function should return 3.

CODE :

PYTHON

```
class Solution:
    # @param A : integer
    # @return an integer
    def numSetBits(self, A):
        lis=[]
        c=0
        while A>0:
            r=A%2
            lis.append(r)
            A/=2
        for i in range(len(lis)):
            if lis[i]==1:
                c+=1
        return c
```

C++

```
int Solution::numSetBits(unsigned int A) {
```

```
    // Do not write main() function.
    // Do not read input, instead use the arguments to the function.
    // Do not print the output, instead return values as specified
    vector<int>lis;
    int c=0;
    int r;
    while(A>0){
        r=A%2;
        lis.push_back(r);
        A/=2;
    }
    for(int i=0;i<lis.size();i++){
        if(lis[i]==1){
            c+=1;
        }
    }
    return c;
}
```

4. Reverse Bits

Reverse the bits of an 32 bit unsigned integer A.

Input Format:

First and only argument of input contains an integer A

Output Format:

return a single unsigned integer denoting minimum xor value

Constraints:

0 <= A < 2^32

For Examples :

Example Input 1:

A = 0

Example Output 1:

0

Explanation 1:

00000000000000000000000000000000
=> 00000000000000000000000000000000

Example Input 2:

A = 3

Example Output 2:

3221225472

Explanation 2:

00000000000000000000000000000011
=> 11000000000000000000000000000000

CODE :

PYTHON

```
class Solution:
    # @param A : unsigned integer
```

```
    # @return an unsigned integer
    def reverse(self, A):
        lis=[0]*32
        i=0
        while A>0:
            r=A%2
            lis[i]=r
            i+=1
            A//=2
        result=0
        lis=lis[::-1]
        for i in range(len(lis)-1,-1,-1):
            result+=lis[i]*pow(2,i)
        return result
```

5. Min XOR value

Given an integer array A of N integers, find the pair of integers in the array which have minimum XOR value. Report the minimum XOR value.

Input Format:

First and only argument of input contains an integer array A

Output Format:

return a single integer denoting minimum xor value

Constraints:

2 <= N <= 100 000

0 <= A[i] <= 1 000 000 000

For Examples :

Example Input 1:

A = [0, 2, 5, 7]

Example Output 1:

2

Explanation:

0 xor 2 = 2

Example Input 2:

A = [0, 4, 7, 9]

Example Output 2:

3

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
    # @return an integer
    class Solution:
    # @param A : list of integers
    # @return an integer
    def findMinXor(self, A):
        A.sort()
```

```
n=len(A)
min_xor=0
min_xor=A[0]^A[1]
for i in range(1,n-1):
    x_or=A[i]^A[i+1]
    min_xor=min(min_xor,x_or)
return min_xor
```

6. Single Number

Given an array of integers, every element appears twice except for one. Find that single one.

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 denoting single element

Constraints:

2 <= N <= 2 000 000

0 <= A[i] <= INT_MAX

For Examples :

Example Input 1:

A = [1, 2, 2, 3, 1]

Example Output 1:

3

Explanation:

3 occurs only once

Example Input 2:

A = [1, 2, 2]

Example Output 2:

1

CODE :

C++

```
int Solution::singleNumber(const vector<int> &A) {
    int ans=0;
    int i;
    for(i=0;i<A.size();i++){
        ans=ans^A[i];
    }
    return ans;
}
```

PYTHON

```
from functools import reduce
class Solution:
```

```
    # @param A : tuple of integers
    # @return an integer
    def singleNumber(self, A):
        return reduce(lambda x,y : x^y, A)
```

(OR)

```
    def singleNumber(self, A):
        x = 0
        for n in A:
            x = x ^ n
        return x
```

7. Single Number II

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
For Examples :
Example Input 1:
A = [1, 2, 4, 3, 3, 2, 2, 3, 1, 1]
Example Output 1:
4
Explanation:
4 occur exactly once
Example Input 2:
A = [0, 0, 0, 1]
Example Output 2:
1

CODE :

PYTHON

```
class Solution:
    # @param A : tuple of integers
    # @return an integer
    def singleNumber(self, A):
        from collections import Counter
        A=list(A)
        A=Counter(A)
        for i,j in A.items():
            if j==1:
                return i
```

C++

```
int Solution::singleNumber(const vector<int> &A) {
    int seen_once = 0, seen_twice = 0;
    for (int num: A) {
        seen_once = ~seen_twice & (seen_once ^ num);
        seen_twice = ~seen_once & (seen_twice ^ num);
    }
    return seen_once;
}
```