DAY 4

INTERVIEW BIT PROBLEMS :

1. All Factors

Given a number N, find all factors of N.
Example:
N = 6
factors = {1, 2, 3, 6}
Make sure the returned array is sorted.

CODE:

PYTHON

```
class Solution:
    # @param A : integer
    # @return a list of integers
    import math
    def allFactors(self, A):
        factors=[1]
        if A==1:
            return factors
        large=[]
        for i in range(2,int(pow(A,0.5))+1):
            if A%i==0:
                x=A//i
                factors.append(i)
                if x!=i:
                    large.append(x)
        large=large[::-1]
        factors.extend(large)
        factors.append(A)
        return factors
```

2. Verify Prime

Given a number N, verify if N is prime or not.
Return 1 if N is prime, else return 0.
Example :
Input : 7
Output : True

CODE :

PYTHON

```
class Solution:
    # @param A : integer
    # @return an integer
    def isPrime(self, A):
        p=1
        if A==1:
            return 0
        for i in range(2,int(pow(A,0.5))+1):
            if A%i==0:
                p=0
                break
        return p
```

3. Prime Numbers

Given a number N, find all prime numbers upto N (N included).
Example:
if N = 7,
all primes till 7 = {2, 3, 5, 7}
Make sure the returned array is sorted.

CODE:

PYTHON

```
class Solution:
    # @param A : integer
    # @return a list of integers
    def sieve(self, A):
        output=[]
        primes=[True for i in range(A+1)]
        primes[0]=False
        primes[1]=False
        p=2
        while p*p<=A:
            if primes[p]==True:
                for i in range(p*2,A+1,p):
                    primes[i]=False
            p+=1
        for i in range(A+1):
            if primes[i]:
                output.append(i)
        return output
```

4. Binary Representation

Given a number N >= 0, find its representation in binary.
Example:
if N = 6,
binary form = 110

CODE:

PYTHON

```
class Solution:
    # @param A : integer
    # @return a strings
```

```
def findDigitsInBinary(self, A):
    if A==0:
        return 0
    bin_str=""
    while A>0:
        bin_str+=str(A%2)
        A//=2
    binary=bin_str[::-1]
    return binary
```

5. Prime Sum

Given an even number (greater than 2), return two prime numbers whose sum will be equal to given number.

Input : 4
Output: 2 + 2 = 4

If there are more than one solutions possible, return the lexicographically smaller solution.
If [a, b] is one solution with a <= b,
and [c,d] is another solution with c <= d, then

[a, b] < [c, d]

If a < c OR a==c AND b < d.

CODE:

PYTHON

```
class Solution:
    # @param A : integer
    # @return a list of integers
```

```
def primesum(self, A):
    primes=[True for i in range(A+1)]
    primes[0]=False
    primes[1]=False
    p=2
    while p*p<=A:
        if primes[p]==True:
            for i in range(p*2,A+1,p):
                primes[i]=False
        p+=1
    output=[]
    for i in range(A):
        if primes[i] and primes[A-i]:
            output.extend([i,A-i])
            break
    return output
```

6. Sum of pairwise Hamming Distance

Hamming distance between two non-negative integers is defined as the number of positions at which the corresponding bits are different.

For example,

HammingDistance(2, 7) = 2, as only the first and the third bit differs in the binary representation of 2 (010) and 7 (111).

Given an array of N non-negative integers, find the sum of hamming distances of all pairs of integers in the array.

Return the answer modulo 1000000007.

Example

Let f(x, y) be the hamming distance defined above.

A=[2, 4, 6]

We return,

f(2, 2) + f(2, 4) + f(2, 6) +

f(4, 2) + f(4, 4) + f(4, 6) +
f(6, 2) + f(6, 4) + f(6, 6) =

0 + 2 + 1
2 + 0 + 1
1 + 1 + 0 = 8

CODE:

PYTHON

```
class Solution:
    # @param A : tuple of integers
    # @return an integer
    def hammingDistance(self, A):
        A=list(A)
        sums=0
        for i in range(32):
            c=0
            for j in range(len(A)):
                if ((A[j] & (1<<i))):
                    c+=1
            sums+=(c*(len(A)-c)*2)%1000000007
        return sums%1000000007
```

7. FizzBuzz

Given a positive integer A, return an array of strings with all the integers from 1 to N.
But for multiples of 3 the array should have "Fizz" instead of the number.
For the multiples of 5, the array should have "Buzz" instead of the number.
For numbers which are multiple of 3 and 5 both, the array should have "FizzBuzz" instead of the number.

Look at the example for more details.
Example
A = 5
Return: [1 2 Fizz 4 Buzz]

CODE :

PYTHON

```
class Solution:
    # @param A : integer
    # @return a list of strings
    def fizzBuzz(self, A):
        output=[]
        for i in range(1,A+1):
            if i%15==0:
                output.append('FizzBuzz')
            elif i%5==0:
                output.append('Buzz')
            elif i%3==0:
                output.append('Fizz')
            else:
                output.append(i)
        return output
```

8. Power Of Two Integers

Given a positive integer which fits in a 32 bit signed integer, find if it can be expressed as A^P where P > 1 and A > 0. A and P both should be integers.
Example
Input : 4
Output : True
as 2^2 = 4.

CODE :

PYTHON

```
class Solution:
    # @param A : integer
    # @return an integer
    def isPower(self, A):
        if A==1:
            return 1
        for i in range(2,33):
            p=round(pow(A,(1/i)))
            if pow(p,i)==A:
                return 1
        return 0
```

9. Excel Column Number

Given a column title as appears in an Excel sheet, return its corresponding column number.

Example:

A -> 1

B -> 2

C -> 3

...

Z -> 26

AA -> 27

AB -> 28

CODE :

PYTHON

```
class Solution:
    # @param A : string
    # @return an integer
    def titleToNumber(self, A):
        n=len(A)
        col_num=0
        for l in range(n):
            col_num*=26
            col_num+=ord(A[l])-ord('A')+1
        return col_num
```

10. Excel Column Title

Given a positive integer, return its corresponding column title as appear in an Excel sheet.

For example:

```
1 -> A
2 -> B
3 -> C
...
26 -> Z
27 -> AA
28 -> AB
```

CODE:

PYTHON:

Implementation 1:

```
class Solution:
```

```
    # @param A : integer
    # @return a strings
    def convertToTitle(self, A):
        col_tit=["\0"]*100
        i=0
        while A>0:
            r=A%26
            if r==0:
                col_tit[i]='Z'
                i+=1
                A=(A//26)-1
            else:
                col_tit[i]=chr((r-1)+ord('A'))
                i+=1
                A//=26
        col_tit[i]='\0'
        col_tit=col_tit[::-1]
        output=[]
        for i in col_tit:
            if i:
                output.append(i)
        return "".join(output)
```

Implementation 2 :

```
class Solution:
    # @param A : integer
    # @return a strings
    def convertToTitle(self, A):
        mydict = 'ZABCDEFGHIJKLMNOPQRSTUVWXY'

        s = ''
        while A > 0:
            aux = A%26
            s = mydict[aux]+s
```

```
            A = A//26
            if 0 == aux:
                A -= 1

        return s
```

11. Palindrome Integer

Determine whether an integer is a palindrome. Do this without extra space.
A palindrome integer is an integer x for which reverse(x) = x where reverse(x) is x with its digit reversed.
Negative numbers are not palindromic.
Example :
Input : 12121
Output : True

Input : 123
Output : False

CODE :

PYTHON

```
class Solution:
    # @param A : integer
    # @return an integer
    def isPalindrome(self, A):
        if(str(A)==str(A)[::-1]):
            return 1
        else:
            return 0
```

12. Reverse integer

Reverse digits of an integer.
Example1:
x = 123,
return 321
Example2:
x = -123,
return -321
Return 0 if the result overflows and does not fit in a 32 bit signed integer

CODE :

PYTHON

```
class Solution:
    # @param A : integer
    # @return an integer
    def reverse(self, A):
        neg=False
        temp=''
        rev=0
        if A<0:
            A=-1*A
            neg=True
        temp=str(A)
        temp=temp[::-1]

        if neg==True:
            rev=-1*int(temp)
        else:
            rev=int(temp)
        if (-1*pow(2,31)<=rev<=(pow(2,31)-1)):
            return rev
```

```
        else:
            return 0
```

13. Greatest Common Divisor

Given 2 non negative integers m and n, find gcd(m, n)
GCD of 2 integers m and n is defined as the greatest integer g such that g is a divisor of both m and n.
Both m and n fit in a 32 bit signed integer.

Example

m : 6
n : 9

GCD(m, n) : 3

CODE :

PYTHON

```
class Solution:
    # @param A : integer
    # @param B : integer
    # @return an integer
    def gcd(self, A, B):
        if B>A:
            A, B = B, A
        while B!=0:
            temp = B
            B = A%B
            A = temp
        return A
```