{1}Power of Two

Given an integer n, return true if it is a power of two. Otherwise, return false.

An integer n is a power of two, if there exists an integer x such that n == 2x.

Example 1:--

Input: n = 1

Output: true

Explanation: 20 = 1

Example 2:-

Input: n = 16

Output: true

Explanation: 24 = 16

Example 3:

Input: n = 3

Output: false

class Solution {

    public boolean isPowerOfTwo(int n) {

       return n>0 && (n&(n-1))==0;

    }

}

{2}Add Digits

Given an integer num, repeatedly add all its digits until the result has only one digit, and return it.

Example 1:

Input: num = 38

Output: 2

Explanation: The process is

38 --> 3 + 8 --> 11

11 --> 1 + 1 --> 2

Since 2 has only one digit, return it.

Example 2:

Input: num = 0

Output: 0

class Solution {

public int addDigits(int num) {

if(num==0){

return 0;

}

if(num%9==0){

return 9;

}

return(num %9);

}

}

{3}. Ugly Number

An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5.

Given an integer n, return true if n is an ugly number.

Example 1:

Input: n = 6

Output: true

Explanation: 6 = 2 × 3

Example 2:

Input: n = 1

Output: true

Explanation: 1 has no prime factors, therefore all of its prime factors are limited to 2, 3, and 5.

Example 3:

Input: n = 14

Output: false

Explanation: 14 is not ugly since it includes the prime factor 7.

class Solution {

public boolean isUgly(int n) {

if(n<1){

return false;

}

while(n%2==0){

n=n/2;

}

while(n%3==0){

n=n/3;

}

while(n%5==0){

n=n/5;

}

return n==1;

}

}

{4}. Nim Game

You are playing the following Nim Game with your friend:

Initially, there is a heap of stones on the table.

You and your friend will alternate taking turns, and you go first.

On each turn, the person whose turn it is will remove 1 to 3 stones from the heap.

The one who removes the last stone is the winner.

Given n, the number of stones in the heap, return true if you can win the game assuming both you and your friend play optimally, otherwise return false.

Example 1:

Input: n = 4

Output: false

Explanation: These are the possible outcomes:

1. You remove 1 stone. Your friend removes 3 stones, including the last stone. Your friend wins.

2. You remove 2 stones. Your friend removes 2 stones, including the last stone. Your friend wins.

3. You remove 3 stones. Your friend removes the last stone. Your friend wins.

In all outcomes, your friend wins.

Example 2:

Input: n = 1

Output: true

Example 3:

Input: n = 2

Output: true

class Solution {

public boolean canWinNim(int n) {

if(n%4==0)

return false;

return true;

}

}

{5}Power of Three

Given an integer n, return true if it is a power of three. Otherwise, return false

An integer n is a power of three, if there exists an integer x such that n == 3x.

Example 1:

Input: n = 27

Output: true

Explanation: 27 = 33

Example 2:

Input: n = 0

Output: false

Explanation: There is no x where 3x = 0.

Example 3:

Input: n = -1

Output: false

Explanation: There is no x where 3x = (-1).

class Solution {

public boolean isPowerOfThree(int n) {

if(n==0)

{

return false;

}

if(n==1)

{

return true;

}

if(n>1)

{

return n%3==0 && isPowerOfThree(n/3);

}

else

{

return false;

}

}

}

{6}Power of Four

Given an integer n, return true if it is a power of four. Otherwise, return false.

An integer n is a power of four, if there exists an integer x such that n == 4x.

Example 1:

Input: n = 16

Output: true

Example 2:

Input: n = 5

Output: false

Example 3:

Input: n = 1

Output: true

class Solution {

public boolean isPowerOfFour(int n) {

if (n < 1)

{

return false;

}

while (n % 4 == 0)

{

n /= 4;

}

return n == 1;

}

}

{7}Count Distinct Numbers on Board

You are given a positive integer n, that is initially placed on a board. Every day, for 109 days, you perform the following procedure:

For each number x present on the board, find all numbers 1 <= i <= n such that x % i == 1.

Then, place those numbers on the board.

Return the number of distinct integers present on the board after 109 days have elapsed.

Note:

Once a number is placed on the board, it will remain on it until the end.

% stands for the modulo operation. For example, 14 % 3 is 2.

Example 1:

Input: n = 5

Output: 4

Explanation: Initially, 5 is present on the board.

The next day, 2 and 4 will be added since 5 % 2 == 1 and 5 % 4 == 1.

After that day, 3 will be added to the board because 4 % 3 == 1.

At the end of a billion days, the distinct numbers on the board will be 2, 3, 4, and 5.

Example 2:

Input: n = 3

Output: 2

Explanation:

Since 3 % 2 == 1, 2 will be added to the board.

After a billion days, the only two distinct numbers on the board are 2 and 3.

class Solution {

public int distinctIntegers(int n) {

return Math.max(n - 1, 1);

}

}

{8} Alternating Digit Sum

You are given a positive integer n. Each digit of n has a sign according to the following rules:

The most significant digit is assigned a positive sign.

Each other digit has an opposite sign to its adjacent digits.

Return the sum of all digits with their corresponding sign.

Example 1:

Input: n = 521

Output: 4

Explanation: (+5) + (-2) + (+1) = 4.

Example 2:

Input: n = 111

Output: 1

Explanation: (+1) + (-1) + (+1) = 1.

Example 3:

Input: n = 886996

Output: 0

Explanation: (+8) + (-8) + (+6) + (-9) + (+9) + (-6) = 0.

class Solution {

public int alternateDigitSum(int n) {

int sum=0;

int num\_digits=0;

while(n!=0)

{

int digits=n%10;

sum=sum+digits\*(num\_digits%2==0?1:-1);

n=n/10;

num\_digits++;

}

return num\_digits%2==0?sum\*-1:sum;

}

}

{9}Difference Between Element Sum and Digit Sum of an Array

You are given a positive integer array nums.

The element sum is the sum of all the elements in nums.

The digit sum is the sum of all the digits (not necessarily distinct) that appear in nums.

Return the absolute difference between the element sum and digit sum of nums.

Note that the absolute difference between two integers x and y is defined as |x - y|.

Example 1:

Input: nums = [1,15,6,3]

Output: 9

Explanation:

The element sum of nums is 1 + 15 + 6 + 3 = 25.

The digit sum of nums is 1 + 1 + 5 + 6 + 3 = 16.

The absolute difference between the element sum and digit sum is |25 - 16| = 9.

Example 2:

Input: nums = [1,2,3,4]

Output: 0

Explanation:

The element sum of nums is 1 + 2 + 3 + 4 = 10.

The digit sum of nums is 1 + 2 + 3 + 4 = 10.

The absolute difference between the element sum and digit sum is |10 - 10| = 0.

class Solution {

public int differenceOfSum(int[] nums) {

int element\_sum=0, digit\_sum = 0;

for(int i=0; i<nums.length; i++){

element\_sum=element\_sum+nums[i];

while(nums[i] != 0){

digit\_sum =digit\_sum+nums[i]%10;

nums[i] /= 10;

}

}

return Math.abs(digit\_sum-element\_sum);

}

}

{10}. Missing Number

Given an array nums containing n distinct numbers in the range [0, n], return the only number in the range that is missing from the array.

Example 1:

Input: nums = [3,0,1]

Output: 2

Explanation: n = 3 since there are 3 numbers, so all numbers are in the range [0,3]. 2 is the missing number in the range since it does not appear in nums.

Example 2:

Input: nums = [0,1]

Output: 2

Explanation: n = 2 since there are 2 numbers, so all numbers are in the range [0,2]. 2 is the missing number in the range since it does not appear in nums.

Example 3:

Input: nums = [9,6,4,2,3,5,7,0,1]

Output: 8

Explanation: n = 9 since there are 9 numbers, so all numbers are in the range [0,9]. 8 is the missing number in the range since it does not appear in nums.

class Solution {

public int missingNumber(int[] nums) {

int n = nums.length;

int ans=0;

for(int i=0;i<n;i++){

ans ^= nums[i] ^ (i+1);

}

return ans;

}

}