PROBLEM SOLVING

SmartCliff Career Mobility Solutions

SDE Readiness Training

Exercise No. : **1**

Topics Covered : **Basic Math, Control Flow, Arrays, Functions, Bit Manipulation**

Date : **08-05-2024**

Solve the following problems

Q No.	Question Detail	Level
1	**Add Digits** **Problem statement:** 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 **Example 2:** **Input:** num = 0 **Output:** 0 **Constraints:** 0 <= num <= 2^31 – 1	Easy
2	**Alternating Digit Sum** **Problem statement:** 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 digit has an opposite sign to its adjacent digits. Return the sum of all digits with their corresponding sign. **Example 1:** **Input:** n = 521	Easy

Output: 4

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

Example 2:

Input: n = 111

Output: 1

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

Constraints:

1 <= n <= 10^9

3 **Divisor Game** Easy

Problem statement: Alice and Bob take turns playing a game, with Alice starting first.

Initially, there is a number n on the chalkboard. On each player's turn, that player makes a move consisting of:

- Choosing any x with 0 < x < n and n % x == 0.
- Replacing the number n on the chalkboard with n - x.

Also, if a player cannot make a move, they lose the game.

Return true if and only if Alice wins the game, assuming both players play optimally.

Example 1:

Input: n = 2

Output: true

Explanation: Alice chooses 1, and Bob has no more moves.

Example 2:

Input: n = 3

Output: false

Explanation: Alice chooses 1, Bob chooses 1, and Alice has no more moves.

Constraints:

1 <= n <= 1000

4	**Happy Number** **Problem statement:** Write an algorithm to determine if a number n is happy. A happy number is a number defined by the following process: • Starting with any positive integer, replace the number by the sum of the squares of its digits. • Repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. • Those numbers for which this process ends in 1 are happy. Return true if n is a happy number, and false if not. **Example 1:** **Input:** n = 19 **Output**: true **Explanation:** $1^2 + 9^2 = 82$ $8^2 + 2^2 = 68$ $6^2 + 8^2 = 100$ $1^2 + 0^2 + 0^2 = 1$ **Example 2:** **Input:** n = 2 **Output**: false **Constraints:** 1 <= n <= 2^31 - 1	Easy
5	**Ugly Number** **Problem statement:** 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	Easy

	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. **Constraints:** • -2^31 <= n <= 2^31 – 1	
6	**Find the Pivot Integer** **Problem statement:** Given a positive integer n, find the pivot integer x such that: • The sum of all elements between 1 and x inclusively equals the sum of all elements between x and n inclusively. Return the pivot integer x. If no such integer exists, return -1. It is guaranteed that there will be at most one pivot index for the given input. **Example 1:** **Input**: n = 8 **Output**: 6 **Explanation:** 6 is the pivot integer since: 1 + 2 + 3 + 4 + 5 + 6 = 6 + 7 + 8 = 21. **Example 2:** **Input**: n = 1 **Output**: 1 **Explanation:** 1 is the pivot integer since: 1 = 1. **Constraints:** • 1 <= n <= 1000	Easy
7	**Count of Matches in Tournament** **Problem Statement:** You are given an integer n, the number of teams in a tournament that has strange rules:	Easy

	• If the current number of teams is even, each team gets paired with another team. A total of n / 2 matches are played, and n / 2 teams advance to the next round. • If the current number of teams is odd, one team randomly advances in the tournament, and the rest gets paired. A total of (n - 1) / 2 matches are played, and (n - 1) / 2 + 1 teams advance to the next round. Return the number of matches played in the tournament until a winner is decided. **Example 1:** **Input:** n = 7 **Output**: 6 **Explanation:** Details of the tournament: - 1st Round: Teams = 7, Matches = 3, and 4 teams advance. - 2nd Round: Teams = 4, Matches = 2, and 2 teams advance. - 3rd Round: Teams = 2, Matches = 1, and 1 team is declared the winner. Total number of matches = 3 + 2 + 1 = 6. **Example 2:** **Input:** n = 14 **Output:** 13 **Explanation:** Details of the tournament: - 1st Round: Teams = 14, Matches = 7, and 7 teams advance. - 2nd Round: Teams = 7, Matches = 3, and 4 teams advance. - 3rd Round: Teams = 4, Matches = 2, and 2 teams advance. - 4th Round: Teams = 2, Matches = 1, and 1 team is declared the winner. Total number of matches = 7 + 3 + 2 + 1 = 13. **Constraints:** • 1 <= n <= 200	
8	**Find the Maximum Achievable Number** **Problem statement:** You are given two integers, num and t. An integer x is called **achievable** if it can become equal to num after applying the following operation no more than t times:	Easy

Increase or decrease x by 1, and simultaneously increase or decrease num by 1.

Return the maximum possible achievable number. It can be proven that there exists at least one achievable number.

Example 1:

Input: num = 4, t = 1

Output: 6

Explanation: The maximum achievable number is x = 6; it can become equal to num after performing this operation:

1- Decrease x by 1, and increase num by 1. Now, x = 5 and num = 5.

It can be proven that there is no achievable number larger than 6.

Example 2:

Input: num = 3, t = 2

Output: 7

Explanation: The maximum achievable number is x = 7; after performing these operations, x will equal num:

1- Decrease x by 1, and increase num by 1. Now, x = 6 and num = 4.

2- Decrease x by 1, and increase num by 1. Now, x = 5 and num = 5.

It can be proven that there is no achievable number larger than 7.

Constraints:

- 1 <= num, t <= 50

9 **Number of handshakes** Easy

Problem statement: The user is asked to take a number as integer n and find out the possible number of handshakes. For example, if there are n number of people in a meeting and find the possible number of handshakes made by the person who entered the room after all were settled.

- For the first person, there will be **N-1** people to shake hands with
- For second person, there will be **N -1** people available **but** as he has already shaken hands with the first person, there will be **N-1-1 = N-2** shake-hands
- For third person, there will be **N-1-1-1 = N-3,** and So On…

	Therefore, the total number of handshake = (**N − 1 + N − 2 + …+ 1 + 0**) = **((N-1) *N) / 2.** **Example:** **Input** : 3 **Output** : Possible number of handshakes: 3 **Input** : 5 **Output :** Possible number of handshakes: 10	
10	**Climbing Stairs** **Problem Statement** : You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top? **Example 1:** **Input**: n = 2 **Output:** 2 **Explanation:** There are two ways to climb to the top. 1. 1 step + 1 step 2. 2 steps **Example 2:** **Input:** n = 3 **Output:** 3 **Explanation:** There are three ways to climb to the top. 1. 1 step + 1 step + 1 step 2. 1 step + 2 steps 3. 2 steps + 1 step **Constraints:** 1 <= n <= 45	Easy
11	**Encrypt the given number** **Problem statement:** Micheal has created his own encryption technique to encrypt a number. He makes use of the logic behind factorial. In a factorial, we multiply the number by its previous number and so on but if we want to encrypt a number we don't multiply in every step like in the	

case of factorial but multiply, divide, add and subtract and repeat in the same order.

So your task is to find the encrypted form of a number using the Micheal's encryption technique and you were being provided with the number.

Sample Input 1:

2

5

8

Sample Output 1:

7

9

Explanation For Sample Input 1:

Test Case 1:

For the first test case, given number is '5' so using the encryption technique we follow the steps: (5 * 4 / 3 + 2 - 1) = 7

Test Case 2:

For the first test case, the given number is '8' so using the encryption technique we follow the steps: (8 * 7 / 6 + 5 - 4 * 3 / 2 - 1) = 9

Sample Input 2:

1

12

Sample Output 2:

13

12 Cube Sum Pairs

Problem statement: You are given a positive integer N, and you have to find the number of ways to represent N as a sum of cubes of two integers(let's say A and B), such that:

N = A^3 + B^3.

Note:

1. A should be greater than or equal to one (A>=1).
2. B should be greater than or equal to zero (B>=0).
3. (A, B) and (B, A) should be considered different solutions, if A is not equal to B, i.e (A, B) and (B, A) will not be distinct if A=B.

Sample Input 1:

9

Sample Output 1:

2

Explanation For Sample Input 1:

There are 2 ways to represent N in the (A^3 + B^3) form ie. {(1, 2), (2, 1)}.

Eg. 1^3 + 2^3 = 9 and 2^3 + 1^3 = 9.

Sample Input 2:

27

Sample Output 2:

1

Explanation For Sample Input 2:

There is only 1 way to represent N in the (A^3 + B^3) form ie. {(3, 0)}.

Eg. 3^3 + 0^3 = 27.

Constraints:

1 <= T <= 10^3

1 <= N <= 10^8

13 Star pattern

Problem statement: Print the following pattern

Pattern for N = 4

```
...*
..***
.*****
*******
```

The dots represent spaces.

Sample Input 1 :

2

1

2

Sample Output 1 :

```
*
*
```

```
  ***
```

Sample Input 2 :

2

3

4

Sample Output 2 :

```
    *
   ***
  *****
    *
   ***
  *****
 *******
```

Constraints :

1 <= N <= 900

14 **Square pattern**

Problem statement : Given an integer N, print a square pattern consisting of N rows and N columns. Each row should contain N characters. The pattern should be such that it forms a square shape with numbers

Example: Pattern for N = 4

4444444444444444

Sample Input 1:

7

Sample Output 1:

```
7777777
7777777
7777777
7777777
7777777
7777777
7777777
```

Sample Input 2:

6

Sample Output 2:

```
666666
666666
666666
666666
```

	666666 666666 **Constraints** 0 <= N <= 50	
15	**Number of 1 Bits** **Problem statement:** Write a function that takes an integer and returns the number of 1 bit it has. **Example Input** **Input1:** 11 **Output1:** 3 **Explanation:**11 is represented as 1101 in binary **Constraints:** • 1<= A <= 4294967295	
16	**Decode XORed Array** **Problem Statement:** There is a hidden integer array arr that consists of n non-negative integers. It was encoded into another integer array encoded of length n - 1, such that encoded[i] = arr[i] XOR arr[i + 1]. For example, if arr = [1,0,2,1], then encoded = [1,2,3]. You are given the encoded array. You are also given an integer first, that is the first element of arr, i.e. arr[0].Return the original array arr. It can be proved that the answer exists and is unique. **Example 1:** **Input:** encoded = [1,2,3], first = 1 **Output:** [1,0,2,1] **Explanation:** If arr = [1,0,2,1], then first = 1 and encoded = [1 XOR 0, 0 XOR 2, 2 XOR 1] = [1,2,3] **Example 2:** **Input:** encoded = [6,2,7,3], first = 4 **Output:** [4,2,0,7,4] **Constraints:** • 1<= n <= 10^4	

17	**Number of Even and Odd Bits** **Problem Statement:** You are given a **positive** integer n.Let even denote the number of even indices in the binary representation of n (**0-indexed**) with value 1.Let odd denote the number of odd indices in the binary representation of n (**0-indexed**) with value 1.Return an integer array answer where answer = [even, odd]. **Example 1:** **Input:** n = 17 **Output:** [2,0] **Explanation:** The binary representation of 17 is 10001. It contains 1 on the 0th and 4th indices. There are 2 even and 0 odd indices. **Example 2:** **Input:** n=2 **Output:** [0,1] **Explanation:** The binary representation of 2 is 10. It contains 1 on the 1st index. There are 0 even and 1 odd indices. **Constraints:** • 1 <= n <= 1000	
18	**Hamming Distance** **Problem Statement:** The Hamming distance between two integers is the number of positions at which the corresponding bits are different. Given two integers x and y, return the Hamming distance between them. **Example 1:** **Input:** x = 1, y = 4 **Output:** 2 **Explanation:** 1 (0 0 0 1) 4 (0 1 0 0) ↑ ↑ The above arrows point to positions where the corresponding bits are different.	

Example 2:

Input: x = 3, y = 1

Output: 1

Constraints:

- 0 <= x, y <= 2^31 – 1

19 **Check if Bitwise OR Has Trailing Zeros**

Problem Statement: You are given an array of **positive** integers nums. You have to check if it is possible to select **two or more** elements in the array such that the bitwise OR of the selected elements has **at least** one trailing zero in its binary representation. Return true if it is possible to select two or more elements whose bitwise OR has trailing zeros, return false otherwise.

Example 1:

Input: nums = [1,2,3,4,5]

Output: true

Explanation: If we select the elements 2 and 4, their bitwise OR is 6, which has the binary representation "110" with one trailing zero.

Example 2:

Input: nums = [2,4,8,16]

Output: true

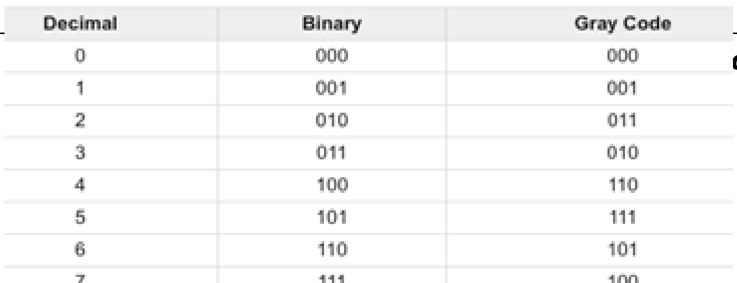
Explanation: If we select the elements 2 and 4, their bitwise OR is 6, which has the binary representation "110" with one trailing zero.
Other possible ways to select elements to have trailing zeroes in the binary representation of their bitwise OR are: (2, 8), (2, 16), (4, 8), (4, 16), (8, 16), (2, 4, 8), (2, 4, 16), (2, 8, 16), (4, 8, 16), and (2, 4, 8, 16).

Constraints:

- 2 <= nums.length <= 100
- 1 <= nums[i] <= 100

20 **Longest Consecutive 1's**

Problem Statement: Given a number N. Find the length of the longest consecutive 1s in its binary representation.



Decimal	Binary	Gray Code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

ory

Example 1:

Input: N = 14

Output: 3

Explanation:

Binary representation of 14 is 1110, in which 111 is the longest consecutive set bits of length is 3.

Example 2:

Input: N = 222

Output: 4

Explanation:

Binary representation of 222 is 11011110, in which 1111 is the longest consecutive set bits of length 4.

Constraints:

1 <= N <= 10^6

21 **Gray to Binary equivalent**

Problem Statement: Given a integer number n in Gray Code, find the binary equivalent of the number n. Return the decimal representation of the binary equivalent.

Example 1:

Input:

n = 4

Output:

7

Explanation:

Given 4, its gray code = 110.

Binary equivalent of the gray code 110 is 100.

Return 7 representing gray code 100.

Example 2:

Input:

n = 15

Output:

10

Explanation:

Given 15 representing gray code 1000.

Binary equivalent of gray code 1000 is 1111.

Return 10 representing gray code 1111

ie binary 1010.

Constraints:

0 <= n <= 10^8

22 **Minimum Bit Flips to Convert Number**

Problem Statement: A bit flip of a number x is choosing a bit in the binary representation of x and flipping it from either 0 to 1 or 1 to 0. Given two integers start and goal, return the minimum number of bit flips to convert start to goal.

Example 1:

Input: start = 10, goal = 7

Output: 3

Explanation: The binary representation of 10 and 7 are 1010 and 0111 respectively. We can convert 10 to 7 in 3 steps:

- Flip the first bit from the right: 1010 -> 1011.
- Flip the third bit from the right: 1011 -> 1111.
- Flip the fourth bit from the right: 1111 -> 0111.

It can be shown we cannot convert 10 to 7 in less than 3 steps. Hence, we return 3.

Example 2:

Input: start = 3, goal = 4

Output: 3

Explanation: The binary representation of 3 and 4 are 011 and 100 respectively. We can convert 3 to 4 in 3 steps:

- Flip the first bit from the right: 011 -> 010.

- Flip the second bit from the right: 010 -> 000.

- Flip the third bit from the right: 000 -> 100.

It can be shown we cannot convert 3 to 4 in less than 3 steps. Hence, we return 3.

Constraints:

- 0 <= start, goal <= 10^9

23 **Toeplitz Matrix**

Problem Statement: Given an m x n matrix, return true if the matrix is Toeplitz. Otherwise, return false. A matrix is Toeplitz if every diagonal from top-left to bottom-right has the same elements.

Example 1:

Input: matrix = [[1,2,3,4],[5,1,2,3],[9,5,1,2]]

Output: true

Explanation:

In the above grid, the diagonals are:

"[9]", "[5, 5]", "[1, 1, 1]", "[2, 2, 2]", "[3, 3]", "[4]".

In each diagonal all elements are the same, so the answer is True.

Example 2:

Input: matrix = [[1,2],[2,2]]

Output: false

Explanation:

The diagonal "[1, 2]" has different elements.

Constraints:

m == matrix.length

n == matrix[i].length

1 <= m, n <= 20

0 <= matrix[i][j] <= 99

24 **Build Array from Permutation**

Problem Statement: Given a zero-based permutation nums (0-indexed), build an array ans of the same length where ans[i] = nums[nums[i]] for each 0 <= i < nums.length and return it.

A zero-based permutation nums is an array of distinct integers from 0 to nums.length - 1 (inclusive).

Example 1:

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

Output: [0,1,2,4,5,3]

Explanation: The array ans is built as follows:

ans = [nums[nums[0]], nums[nums[1]], nums[nums[2]], nums[nums[3]], nums[nums[4]], nums[nums[5]]]

= [nums[0], nums[2], nums[1], nums[5], nums[3], nums[4]]

= [0,1,2,4,5,3]

Example 2:

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

Output: [4,5,0,1,2,3]

Explanation: The array ans is built as follows:

ans = [nums[nums[0]], nums[nums[1]], nums[nums[2]], nums[nums[3]], nums[nums[4]], nums[nums[5]]]

= [nums[5], nums[0], nums[1], nums[2], nums[3], nums[4]]

= [4,5,0,1,2,3]

Constraints:

- 1 <= nums.length <= 1000
- 0 <= nums[i] < nums.length
- The elements in nums are distinct.

25 Concatenation of Array

Problem Statement: Given an integer array nums of length n, you want to create an array ans of length 2n where ans[i] == nums[i] and ans[i + n] == nums[i] for 0 <= i < n (0-indexed).

Specifically, ans is the concatenation of two nums arrays.

Return the array ans.

Example 1:

Input: nums = [1,2,1]

Output: [1,2,1,1,2,1]

Explanation: The array ans is formed as follows:

- ans = [nums[0],nums[1],nums[2],nums[0],nums[1],nums[2]]
- ans = [1,2,1,1,2,1]

Example 2:

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

Output: [1,3,2,1,1,3,2,1]

Explanation: The array ans is formed as follows:

- ans = [nums[0],nums[1],nums[2],nums[3],nums[0],nums[1],nums[2],nums[3]]
- ans = [1,3,2,1,1,3,2,1]

Constraints:

n == nums.length

1 <= n <= 1000

1 <= nums[i] <= 1000

26 Find N Unique Integers Sum up to Zero

Problem statement : Given an integer n, return any array containing n unique integers such that they add up to 0. The integers for sum can be either positive or negative.

Example 1:

Input: n = 5

Output: [-7,-1,1,3,4]

Explanation: These arrays also are accepted [-5,-1,1,2,3] , [-3,-1,2,-2,4].

Example 2:

Input: n = 3

Output: [-1,0,1]

Constraints:

1 <= n <= 1000

27 Matrix Diagonal Sum

Problem statement: Given a square matrix mat, return the sum of the matrix diagonals. Only include the sum of all the elements on the primary diagonal and all the elements on the secondary diagonal that are not part of the primary diagonal.

Example 1:

Input: mat = [[1,2,3],
[4,5,6],
[7,8,9]]

Output: 25

Explanation: Diagonals sum: 1 + 5 + 9 + 3 + 7 = 25

Notice that element mat[1][1] = 5 is counted only once.

Example 2:

Input: mat = [[1,1,1,1],
[1,1,1,1],
[1,1,1,1],
[1,1,1,1]]

Output: 8

Constraints:

n == mat.length == mat[i].length

1 <= n <= 100

1 <= mat[i][j] <= 100

28 Modify the Matrix

Problem statement : Given a 0-indexed m x n integer matrix matrix, create a new 0-indexed matrix called answer. Make answer equal to matrix, then replace each element with the value -1 with the maximum element in its respective column.

Return the matrix answer.

Example 1:

Input: matrix = [[1,2,-1],[4,-1,6],[7,8,9]]

Output: [[1,2,9],[4,8,6],[7,8,9]]

Explanation: The diagram above shows the elements that are changed (in blue).

- We replace the value in the cell [1][1] with the maximum value in the column 1, that is 8.
- We replace the value in the cell [0][2] with the maximum value in the column 2, that is 9.

Example 2:
Input: matrix = [[3,-1],[5,2]]
Output: [[3,2],[5,2]]
Explanation: The diagram above shows the elements that are changed (in blue).

Constraints:

- m == matrix.length
- n == matrix[i].length
- 2 <= m, n <= 50
- -1 <= matrix[i][j] <= 100
- The input is generated such that each column contains at least one non-negative integer.

29 **Number of Students Doing Homework at a Given Time**

Problem statement: Given two integer arrays startTime and endTime and given an integer queryTime.The ith student started doing their homework at the time startTime[i] and finished it at time endTime[i].Return the number of students doing their homework at time queryTime. More formally, return the number of students where queryTime lays in the interval [startTime[i], endTime[i]] inclusive.

Example 1:
Input: startTime = [1,2,3], endTime = [3,2,7], queryTime = 4
Output: 1
Explanation: We have 3 students where:
The first student started doing homework at time 1 and finished at time 3 and wasn't doing anything at time 4.
The second student started doing homework at time 2 and finished at time 2 and also wasn't doing anything at time 4.
The third student started doing homework at time 3 and finished at time 7 and was the only student doing homework at time 4.

Example 2:

Input: startTime = [4], endTime = [4], queryTime = 4

Output: 1

Explanation: The only student was doing their homework at the queryTime.

Constraints:

startTime.length == endTime.length

1 <= startTime.length <= 100

1 <= startTime[i] <= endTime[i] <= 1000

1 <= queryTime <= 1000

30 **Count Hills and Valleys in an Array**

Problem statement : You are given a 0-indexed integer array nums. An index i is part of a hill in nums if the closest non-equal neighbors of i are smaller than nums[i]. Similarly, an index i is part of a valley in nums if the closest non-equal neighbors of i are larger than nums[i]. Adjacent indices i and j are part of the same hill or valley if nums[i] == nums[j].
Note that for an index to be part of a hill or valley, it must have a non-equal neighbor on both the left and right of the index.Return the number of hills and valleys in nums.

Example 1:

Input: nums = [2,4,1,1,6,5]

Output: 3

Explanation:

At index 0: There is no non-equal neighbor of 2 on the left, so index 0 is neither a hill nor a valley.

At index 1: The closest non-equal neighbors of 4 are 2 and 1. Since 4 > 2 and 4 > 1, index 1 is a hill.

At index 2: The closest non-equal neighbors of 1 are 4 and 6. Since 1 < 4 and 1 < 6, index 2 is a valley.

At index 3: The closest non-equal neighbors of 1 are 4 and 6. Since 1 < 4 and 1 < 6, index 3 is a valley, but note that it is part of the same valley as index 2.

At index 4: The closest non-equal neighbors of 6 are 1 and 5. Since 6 > 1 and 6 > 5, index 4 is a hill.
At index 5: There is no non-equal neighbor of 5 on the right, so index 5 is neither a hill nor a valley.
There are 3 hills and valleys so we return 3.

Example 2:
Input: nums = [6,6,5,5,4,1]
Output: 0
Explanation:
At index 0: There is no non-equal neighbor of 6 on the left, so index 0 is neither a hill nor a valley.
At index 1: There is no non-equal neighbor of 6 on the left, so index 1 is neither a hill nor a valley.
At index 2: The closest non-equal neighbors of 5 are 6 and 4. Since 5 < 6 and 5 > 4, index 2 is neither a hill nor a valley.
At index 3: The closest non-equal neighbors of 5 are 6 and 4. Since 5 < 6 and 5 > 4, index 3 is neither a hill nor a valley.
At index 4: The closest non-equal neighbors of 4 are 5 and 1. Since 4 < 5 and 4 > 1, index 4 is neither a hill nor a valley.
At index 5: There is no non-equal neighbor of 1 on the right, so index 5 is neither a hill nor a valley.
There are 0 hills and valleys so we return 0.

Constraints:

3 <= nums.length <= 100
1 <= nums[i] <= 100