Leet Code Training Day 27 Prime and Divisor

Today we will discuss math problems, we start with Prime and Divisor. This is the most common problem in the math category.

# Prime Numbers

The prime number is a number which can only be divisible by 1 and itself.

The common problem for prime can be in one of the following patterns:

1. Check if a number is prime or not.
2. Find all the prime numbers within a range.
3. Get all the prime factors for a number.

For #1 problem, there is no easy way, you have to check 2 and all the odd numbers until the square root of the number. Please note the largest divisor to test is the square root, round down to integer, itself.

For #2 problem, we can open an array, and check multiple of 2 and all prime numbers, mark them as excluded, when you walk through from 2 to N, the number which is not marked yet are prime.

For #3 problem. It is similar to #1, but the difference is that the number N itself, when we find a prime factor of it, will get divided and get replaced and the number will become smaller and smaller.

## 2521. Distinct Prime Factors of Product of Array

Medium

Given an array of positive integers nums, return the number of distinct prime factors in the product of the elements of nums.

Note that:

A number greater than 1 is called prime if it is divisible by only 1 and itself.

An integer val1 is a factor of another integer val2 if val2 / val1 is an integer.

Example 1:

Input: nums = [2,4,3,7,10,6]

Output: 4

Explanation:

The product of all the elements in nums is: 2 \* 4 \* 3 \* 7 \* 10 \* 6 = 10080 = 25 \* 32 \* 5 \* 7.

There are 4 distinct prime factors, so we return 4.

Example 2:

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

Output: 1

Explanation:

The product of all the elements in nums is: 2 \* 4 \* 8 \* 16 = 1024 = 210.

There is 1 distinct prime factor so we return 1.

Constraints:

1 <= nums.length <= 10^4

2 <= nums[i] <= 1000

### Analysis:

This is a typical type 3 problem. We need to calculate all the prime factors of the number. You can see we keep on dividing the number n and making it smaller.

/// <summary>

/// Leet Code 2521. Distinct Prime Factors of Product of Array

///

/// Medium

///

/// Given an array of positive integers nums, return the number of

/// distinct prime factors in the product of the elements of nums.

///

/// Note that:

///

/// A number greater than 1 is called prime if it is divisible by

/// only 1 and itself.

/// An integer val1 is a factor of another integer val2 if

/// val2 / val1 is an integer.

///

/// Example 1:

/// Input: nums = [2,4,3,7,10,6]

/// Output: 4

/// Explanation:

/// The product of all the elements in nums is:

/// 2 \* 4 \* 3 \* 7 \* 10 \* 6 = 10080 = 25 \* 32 \* 5 \* 7.

/// There are 4 distinct prime factors so we return 4.

///

/// Example 2:

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

/// Output: 1

/// Explanation:

/// The product of all the elements in nums is:

/// 2 \* 4 \* 8 \* 16 = 1024 = 210.

/// There is 1 distinct prime factor so we return 1.

///

/// Constraints:

/// 1. 1 <= nums.length <= 10^4

/// 2. 2 <= nums[i] <= 1000

/// </summary>

int LeetCodeMath::distinctPrimeFactors(vector<int>& nums)

{

unordered\_set<int> primes;

for (size\_t i = 0; i < nums.size(); i++)

{

int n = nums[i];

for (int j = 2; j <= (int)sqrt(n); j++)

{

if (n % j == 0)

{

while (n % j == 0) n /= j;

primes.insert(j);

}

}

if (n != 1) primes.insert(n);

}

return primes.size();

}

## 2523. Closest Prime Numbers in Range

Medium

Given two positive integers left and right, find the two integers num1 and num2 such that:

left <= nums1 < nums2 <= right .

nums1 and nums2 are both prime numbers.

nums2 - nums1 is the minimum amongst all other pairs satisfying the above conditions.

Return the positive integer array ans = [nums1, nums2]. If there are multiple pairs satisfying these conditions, return the one with the minimum nums1 value or [-1, -1] if such numbers do not exist.

A number greater than 1 is called prime if it is only divisible by 1 and itself.

Example 1:

Input: left = 10, right = 19

Output: [11,13]

Explanation: The prime numbers between 10 and 19 are 11, 13, 17, and 19.

The closest gap between any pair is 2, which can be achieved by [11,13] or [17,19].

Since 11 is smaller than 17, we return the first pair.

Example 2:

Input: left = 4, right = 6

Output: [-1,-1]

Explanation: There exists only one prime number in the given range, so the conditions cannot be satisfied.

Constraints:

1 <= left <= right <= 10^6

### Analysis:

This is a typical type 2 problem. We need to find all the prime numbers between left and right.

/// <summary>

/// Leet Code 2523. Closest Prime Numbers in Range

///

/// Medium

///

/// Given two positive integers left and right, find the two integers num1

/// and num2 such that:

///

/// left <= nums1 < nums2 <= right .

/// nums1 and nums2 are both prime numbers.

/// nums2 - nums1 is the minimum amongst all other pairs satisfying the

/// above conditions.

/// Return the positive integer array ans = [nums1, nums2]. If there are

/// multiple pairs satisfying these conditions, return the one with the

/// minimum nums1 value or [-1, -1] if such numbers do not exist.

///

/// A number greater than 1 is called prime if it is only divisible by 1

/// and itself.

///

/// Example 1:

///

/// Input: left = 10, right = 19

/// Output: [11,13]

/// Explanation: The prime numbers between 10 and 19 are 11, 13, 17,

/// and 19.

/// The closest gap between any pair is 2, which can be achieved by

/// [11,13] or [17,19].

/// Since 11 is smaller than 17, we return the first pair.

///

/// Example 2:

/// Input: left = 4, right = 6

/// Output: [-1,-1]

/// Explanation: There exists only one prime number in the given range,

/// so the conditions cannot be satisfied.

///

/// Constraints:

/// 1. 1 <= left <= right <= 10^6

/// </summary>

vector<int> LeetCodeMath::closestPrimes(int left, int right)

{

vector<int> dp(right + 1, 1);

dp[0] = dp[1] = 0;

for (int i = 2; i <= right; i++)

{

if (dp[i] == 0) continue;

for (int j = 2 \* i; j <= right; j += i)

{

dp[j] = 0;

}

}

int prev = -1;

vector<int> result = { -1, -1 };

int min\_diff = INT\_MAX;

for (int i = left; i <= right; i++)

{

if (dp[i] == 1)

{

if (prev != -1)

{

if (i - prev < min\_diff)

{

result[0] = prev;

result[1] = i;

min\_diff = i - prev;

}

}

prev = i;

}

}

return result;

}

# Greatest Common Divisor

First, we need to see how to calculate the common divisor. Assume you have two positive numbers a and b, and a > b, if a %b == 0 then b is the common divisor, otherwise you replace a by a % b and exchange a and b keep on above test until you get a perfect division. The code example looks like this:

while (b != 0 && a % b != 0)

{

a = a % b;

swap(a, b);

}

Also note the common divisor can be transmitted, so if you have three number a, b and c and want to calculate the greatest common divisor, you can calculate the greatest common divisor between a and b to get x, then calculate greatest common divisor between x and c.

## 2436. Minimum Split Into Subarrays With GCD Greater Than One

Medium

You are given an array nums consisting of positive integers.

Split the array into one or more disjoint subarrays such that:

Each element of the array belongs to exactly one subarray, and

The GCD of the elements of each subarray is strictly greater than 1.

Return the minimum number of subarrays that can be obtained after the split.

Note that:

The GCD of a subarray is the largest positive integer that evenly divides all the elements of the subarray.

A subarray is a contiguous part of the array.

Example 1:

Input: nums = [12,6,3,14,8]

Output: 2

Explanation: We can split the array into the subarrays: [12,6,3] and [14,8].

- The GCD of 12, 6 and 3 is 3, which is strictly greater than 1.

- The GCD of 14 and 8 is 2, which is strictly greater than 1.

It can be shown that splitting the array into one subarray will make the GCD = 1.

Example 2:

Input: nums = [4,12,6,14]

Output: 1

Explanation: We can split the array into only one subarray, which is the whole array.

Constraints:

1 <= nums.length <= 2000

2 <= nums[i] <= 10^9

### Analysis:

We keep on calculate group greatest common divisor in the array, when we get 1, we stop and split array.

/// <summary>

/// Leet Code 2436. Minimum Split Into Subarrays With GCD Greater Than One

///

/// Medium

///

/// You are given an array nums consisting of positive integers.

///

/// Split the array into one or more disjoint subarrays such that:

///

/// Each element of the array belongs to exactly one subarray, and

/// The GCD of the elements of each subarray is strictly greater than 1.

/// Return the minimum number of subarrays that can be obtained after

/// the split.

///

/// Note that:

/// The GCD of a subarray is the largest positive integer that evenly

/// divides all the elements of the subarray.

/// A subarray is a contiguous part of the array.

///

/// Example 1:

/// Input: nums = [12,6,3,14,8]

/// Output: 2

/// Explanation: We can split the array into the

/// subarrays: [12,6,3] and [14,8].

/// - The GCD of 12, 6 and 3 is 3, which is strictly greater than 1.

/// - The GCD of 14 and 8 is 2, which is strictly greater than 1.

/// It can be shown that splitting the array into one subarray will make

/// the GCD = 1.

///

/// Example 2:

/// Input: nums = [4,12,6,14]

/// Output: 1

/// Explanation: We can split the array into only one subarray, which

/// is the whole array.

///

/// Constraints:

/// 1. 1 <= nums.length <= 2000

/// 2. 2 <= nums[i] <= 10^9

/// </summary>

int LeetCodeMath::minimumSplits(vector<int>& nums)

{

int result = 1;

int prev = nums[0];

for (size\_t i = 1; i < nums.size(); i++)

{

int a = prev;

int b = nums[i];

if (a < b) swap(a,b);

while (a % b != 0)

{

a = a % b;

swap(a, b);

}

prev = b;

if (prev == 1)

{

prev = nums[i];

result++;

}

}

return result;

}

## 2447. Number of Subarrays With GCD Equal to K

Medium

Given an integer array nums and an integer k, return the number of subarrays of nums where the greatest common divisor of the subarray's elements is k.

A subarray is a contiguous non-empty sequence of elements within an array.

The greatest common divisor of an array is the largest integer that evenly divides all the array elements.

Example 1:

Input: nums = [9,3,1,2,6,3], k = 3

Output: 4

Explanation: The subarrays of nums where 3 is the greatest common divisor of all the subarray's elements are:

- [9,3,1,2,6,3]

- [9,3,1,2,6,3]

- [9,3,1,2,6,3]

- [9,3,1,2,6,3]

Example 2:

Input: nums = [4], k = 7

Output: 0

Explanation: There are no subarrays of nums where 7 is the greatest common divisor of all the subarray's elements.

Constraints:

1 <= nums.length <= 1000

1 <= nums[i], k <= 10^9

### Analysis:

We keep on testing additional element in a subarray, to see if we can get greatest common divisor as K, if we get a common divisor which is not multiple of K, we break out because we can not get GCD as K any more..

At same time we can not determine if we remove an element from sub array, we will be able to get a GCD as K. So we have to re-establish the subarray chain test from the next element.

/// <summary>

/// Leet Code 2447. Number of Subarrays With GCD Equal to K

///

/// Medium

///

/// Given an integer array nums and an integer k, return the number of

/// subarrays of nums where the greatest common divisor of the subarray's

/// elements is k.

///

/// A subarray is a contiguous non-empty sequence of elements within an

/// array.

///

/// The greatest common divisor of an array is the largest integer that

/// evenly divides all the array elements.

///

/// Example 1:

/// Input: nums = [9,3,1,2,6,3], k = 3

/// Output: 4

/// Explanation: The subarrays of nums where 3 is the greatest common

/// divisor of all the subarray's elements are:

/// - [9,3,1,2,6,3]

/// - [9,3,1,2,6,3]

/// - [9,3,1,2,6,3]

/// - [9,3,1,2,6,3]

///

/// Example 2:

/// Input: nums = [4], k = 7

/// Output: 0

/// Explanation: There are no subarrays of nums where 7 is the greatest

/// common divisor of all the subarray's elements.

///

/// Constraints:

/// 1. 1 <= nums.length <= 1000

/// 2. 1 <= nums[i], k <= 10^9

/// </summary>

int LeetCodeMath::subarrayGCD(vector<int>& nums, int k)

{

int result = 0;

for (size\_t i = 0; i < nums.size(); i++)

{

int a = nums[i];

for (int j = i; j >= 0; j--)

{

int b = nums[j];

while (a % b != 0)

{

if (a < b) swap(a, b);

else

{

a = a - b;

swap(a, b);

}

}

a = b;

if (a == k) result++;

else if (a % k != 0) break;

}

}

return result;

}

## 2183. Count Array Pairs Divisible by K

Hard

Given a 0-indexed integer array nums of length n and an integer k, return the number of pairs (i, j) such that:

0 <= i < j <= n - 1 and

nums[i] \* nums[j] is divisible by k.

Example 1:

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

Output: 7

Explanation:

The 7 pairs of indices whose corresponding products are divisible by 2 are

(0, 1), (0, 3), (1, 2), (1, 3), (1, 4), (2, 3), and (3, 4).

Their products are 2, 4, 6, 8, 10, 12, and 20 respectively.

Other pairs such as (0, 2) and (2, 4) have products 3 and 15 respectively, which are not divisible by 2.

Example 2:

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

Output: 0

Explanation: There does not exist any pair of indices whose corresponding product is divisible by 5.

Constraints:

1 <= nums.length <= 105

1 <= nums[i], k <= 105

### Analysis:

We can calculate the great common divisor with every number and K and find out in the existing divisors already calculated so far, how many of them times the current one will be divisible by K.

Because gcd of any number and K must be a divisor of K, so the number of them are limited, we can track it in hash table.

/// <summary>

/// Leet Code 2183. Count Array Pairs Divisible by K

///

/// Hard

///

/// Given a 0-indexed integer array nums of length n and an integer k,

/// return the number of pairs (i, j) such that:

///

/// 0 <= i < j <= n - 1 and

/// nums[i] \* nums[j] is divisible by k.

///

/// Example 1:

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

/// Output: 7

/// Explanation:

/// The 7 pairs of indices whose corresponding products are divisible

/// by 2 are

/// (0, 1), (0, 3), (1, 2), (1, 3), (1, 4), (2, 3), and (3, 4).

/// Their products are 2, 4, 6, 8, 10, 12, and 20 respectively.

/// Other pairs such as (0, 2) and (2, 4) have products 3 and 15

/// respectively, which are not divisible by 2.

///

/// Example 2:

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

/// Output: 0

/// Explanation: There does not exist any pair of indices whose

/// corresponding product is divisible by 5.

///

/// Constraints:

/// 1. 1 <= nums.length <= 10^5

/// 2. 1 <= nums[i], k <= 10^5

/// </summary>

long long LeetCodeMath::countPairs(vector<int>& nums, int k)

{

unordered\_map<int, int> gcd\_count;

long long result = 0;

for (size\_t i = 0; i < nums.size(); i++)

{

long long g = gcd(nums[i], k);

for (auto itr : gcd\_count)

{

if ((long long)itr.first \* (long long)g % (long long)k == 0)

{

result += itr.second;

}

}

gcd\_count[(int)g]++;

}

return result;

}

## 2197. Replace Non-Coprime Numbers in Array

Hard

You are given an array of integers nums. Perform the following steps:

Find any two adjacent numbers in nums that are non-coprime.

If no such numbers are found, stop the process.

Otherwise, delete the two numbers and replace them with their LCM (Least Common Multiple).

Repeat this process as long as you keep finding two adjacent non-coprime numbers.

Return the final modified array. It can be shown that replacing adjacent non-coprime numbers in any arbitrary order will lead to the same result.

The test cases are generated such that the values in the final array are less than or equal to 108.

Two values x and y are non-coprime if GCD(x, y) > 1 where GCD(x, y) is the Greatest Common Divisor of x and y.

Example 1:

Input: nums = [6,4,3,2,7,6,2]

Output: [12,7,6]

Explanation:

- (6, 4) are non-coprime with LCM(6, 4) = 12. Now, nums = [12,3,2,7,6,2].

- (12, 3) are non-coprime with LCM(12, 3) = 12. Now, nums = [12,2,7,6,2].

- (12, 2) are non-coprime with LCM(12, 2) = 12. Now, nums = [12,7,6,2].

- (6, 2) are non-coprime with LCM(6, 2) = 6. Now, nums = [12,7,6].

There are no more adjacent non-coprime numbers in nums.

Thus, the final modified array is [12,7,6].

Note that there are other ways to obtain the same resultant array.

Example 2:

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

Output: [2,1,1,3]

Explanation:

- (3, 3) are non-coprime with LCM(3, 3) = 3. Now, nums = [2,2,1,1,3,3].

- (3, 3) are non-coprime with LCM(3, 3) = 3. Now, nums = [2,2,1,1,3].

- (2, 2) are non-coprime with LCM(2, 2) = 2. Now, nums = [2,1,1,3].

There are no more adjacent non-coprime numbers in nums.

Thus, the final modified array is [2,1,1,3].

Note that there are other ways to obtain the same resultant array.

Constraints:

1 <= nums.length <= 10^5

1 <= nums[i] <= 10^5

The test cases are generated such that the values in the final array are less than or equal to 10^8.

### Analysis:

The algorithm is to use stack, process the last two numbers, if non-coprime, merge them.

/// <summary>

/// Leet Code 2197. Replace Non-Coprime Numbers in Array

///

/// Hard

///

/// You are given an array of integers nums. Perform the following steps:

///

/// Find any two adjacent numbers in nums that are non-coprime.

/// If no such numbers are found, stop the process.

/// Otherwise, delete the two numbers and replace them with their LCM

/// (Least Common Multiple).

/// Repeat this process as long as you keep finding two adjacent

/// non-coprime numbers.

/// Return the final modified array. It can be shown that replacing

/// adjacent non-coprime numbers in any arbitrary order will lead to the

/// same result.

///

/// The test cases are generated such that the values in the final array

/// are less than or equal to 108.

///

/// Two values x and y are non-coprime if GCD(x, y) > 1 where GCD(x, y)

/// is the Greatest Common Divisor of x and y.

///

/// Example 1:

/// Input: nums = [6,4,3,2,7,6,2]

/// Output: [12,7,6]

/// Explanation:

/// - (6, 4) are non-coprime with LCM(6, 4) = 12. Now,

/// nums = [12,3,2,7,6,2].

/// - (12, 3) are non-coprime with LCM(12, 3) = 12. Now,

/// nums = [12,2,7,6,2].

/// - (12, 2) are non-coprime with LCM(12, 2) = 12. Now,

/// nums = [12,7,6,2].

/// - (6, 2) are non-coprime with LCM(6, 2) = 6. Now, nums = [12,7,6].

/// There are no more adjacent non-coprime numbers in nums.

/// Thus, the final modified array is [12,7,6].

/// Note that there are other ways to obtain the same resultant array.

///

/// Example 2:

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

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

/// Explanation:

/// - (3, 3) are non-coprime with LCM(3, 3) = 3. Now, nums = [2,2,1,1,3,3].

/// - (3, 3) are non-coprime with LCM(3, 3) = 3. Now, nums = [2,2,1,1,3].

/// - (2, 2) are non-coprime with LCM(2, 2) = 2. Now, nums = [2,1,1,3].

/// There are no more adjacent non-coprime numbers in nums.

/// Thus, the final modified array is [2,1,1,3].

/// Note that there are other ways to obtain the same resultant array.

///

/// Constraints:

/// 1. 1 <= nums.length <= 10^5

/// 2. 1 <= nums[i] <= 105

/// 3. The test cases are generated such that the values in the final

/// array are less than or equal to 10^8.

/// </summary>

vector<int> LeetCodeMath::replaceNonCoprimes(vector<int>& nums)

{

vector<int> result;

for (size\_t i = 0; i < nums.size(); i++)

{

int right = nums[i];

while (!result.empty())

{

int left = result.back();

int common = (int)gcd((long long)left, (long long)right);

if (common != 1)

{

right = (left / common) \* right;

result.pop\_back();

}

else

{

break;

}

}

result.push\_back(right);

}

return result;

}