## **Tarea 1** de Soluciones Algorítmicas Avanzadas, ITESO, Luis Gatica

### Fecha de entrega: 25 de agosto de 2024

Nombre: Yael Alejandro Rodríguez Barreto

# Teoría de números

Use el lenguaje de programación de su preferencia para resolver los siguientes problemas. Cuídese especialmente de errores por desbordar los tipos de datos utilizados. Además de su código, incluya una captura de pantalla del último resultado que le dio el juez virtual. Considere que las búsquedas lineales en arreglos o vectores pueden ser muy lentas en caso de querer utilizarlas.

1. **Comprobación de la conjetura de Goldbach**

<https://vjudge.net/problem/UVA-543>

Calcule la criba de Eratóstenes una sola vez para resolver todos los casos.

### Creación de la criba y obtención de primos

primes = []

max = 0

def fillSieve(size):

    sieve = [0] \* size

    for i in range(2, size, 1):

        if(sieve[i]):

            continue

        for j in range(i\*2, size, i):

            sieve[j] = 1

        primes.append(i)

    return sieve

### Verificación de conjetura de goldbach

def goldbach(result):

    for p1 in range(max):

        sum = result - primes[p1]

        if(sieve[sum] == 0):

            return "{} = {} + {}".format(result, primes[p1], sum)

    return "Goldbach's conjecture is wrong."

### Obtener y devolver datos

numbers = []

result = int(input())

while(result != 0):

    max = max if max > result else result

    numbers.append(result)

    result = int(input())

sieve = fillSieve(max)

for i in numbers:

    print(goldbach(i))

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. **Números perfectos**

<https://vjudge.net/problem/UVA-1180>

La cantidad de números perfectos que el problema menciona es muy pequeña; además, presenta un valor límite.

¿Es posible precomputar información que ayude a resolver cada caso individual en tiempo constante?

### Entrada de datos

n = int(input())

numbers = input().split(",")

results = [2,3,5,7,13,17,19]

for i in range(n):

    r = int(numbers[i])

    if(r in results):

        print("Yes")

    else:

        print("No")

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. **Divisor primo más grande**

<https://vjudge.net/problem/UVA-11466>

El número límite es demasiado grande para crear un arreglo y aplicar la criba. Puede adaptar el algoritmo de factorización de números para resolver cada caso individualmente; considere atentamente el caso especial de este algoritmo en que el último factor se agrega fuera de los ciclos.

El problema indica que la entrada será un número entero. ¿Hay algún subconjunto infinito de enteros que requiera un tratamiento especial?

### Buscar el divisor primo más grande

def maxFactor(n):

    n = n if n > 0 else n \* -1

    factor = -1

    change = 0

    i = 2

    while i\*i <= n:

        while(n % i == 0):

            if(i != factor):

                factor = i

                change += 1

            n /= i

        i += 1

    if(n > 1):

        factor = n

        change += 1

    return int(factor) if change > 1 else -1

### Obtener y devolver datos

numbers = []

n = int(input())

while(n != 0):

    numbers.append(n)

    n = int(input())

for i in numbers:

    print(maxFactor(i))

Texto

Descripción generada automáticamente

1. **Ceros a la derecha en un factorial**

<https://leetcode.com/problems/factorial-trailing-zeroes/>

Utilice la fórmula de Legendre, que sirve para calcular la multiplicidad *v* de un número primo *p* en el factorial de un entero no negativo *n*.

Observe que la fórmula de Legendre sólo admite números primos como valores del parámetro *p*. ¿Cómo se puede usar su resultado para números compuestos (números no primos)?

### Legendre específico para 5

def trailingZeroes(n):

    count = 0

    while(n > 0):

        n //= 5

        count += n

    return count

a = trailingZeroes(10)

print(a)

Captura de pantalla de un celular

Descripción generada automáticamente

1. **Ceros a la derecha en un factorial y su cantidad de dígitos (en bases distintas)**

<https://vjudge.net/problem/UVA-10061>

En el problema anterior había que buscar la multiplicidad del factor 10, porque la base en la que se expresaría el resultado del factorial era justo la base 10; en este problema la base puede ser distinta.

¿Qué tienen en común los enteros con ceros a la derecha en la base 10? ¿Qué tienen en común los enteros expresados en base 2 que tienen ceros a la derecha en esa representación? ¿Y en base 16? ¿En base 3?

¿Cómo se puede contar la cantidad de dígitos en la representación de un número x en una base *b* >= 2? ¿Qué tiene que ver esto con el método de las divisiones largas usado para cambiar la representación de un número desde la base 10 a otra?

¿Cuáles leyes matemáticas (externas a la teoría de números en sí misma) se aplican a la respuesta sobre el conteo de dígitos y cómo pueden facilitar la implementación de la solución?

Captura de pantalla de computadora

Descripción generada automáticamente

import math

### Suma de los logaritmos de cada miembro del factorial

def numDigits(n, b):

    sum = 1

    for i in range(1, n+1):

        sum += math.log(i) / math.log(b)

    return int(sum)

### Formula de legendre para para multiplicidad de primos en factoriales

def legendre(n, p):

    count = 0

    while(n > 0):

        n //= p

        count += n

    return int(count)

### Factores primo más grande y sus apariciones

def bigFactor(n):

    l\_factors = []

    i = 1

    while i\*i <= n:

        i += 1

        count = 0

        while(n % i == 0):

            count += 1

            n /= i

        if(count > 0):

            l\_factors.append([i, count])

    if(n > 1):

        l\_factors.append([n, 1])

    return l\_factors

### Entrada de datos

args = input()

inp = []

while(args):

    args = args.split()

    inp.append(args)

    args = input()

for i in inp:

    num = int(i[0])

    base = int(i[1])

    f = bigFactor(base)

    zeros = 0

    for i in f:

        spec = legendre(num, i[0])//i[1]

        zeros = spec if spec < zeros or zeros <= 0 else zeros

    digits = numDigits(num, base)

    print(str(zeros) + " " + str(digits))

|  |  |
| --- | --- |
| Criterio de evaluación | Puntos |
| Ejercicio 1 | 20 |
| Ejercicio 2 | 20 |
| Ejercicio 3 | 20 |
| Ejercicio 4 | 15 |
| Ejercicio 5 | 25 |
| TOTAL | **100** |