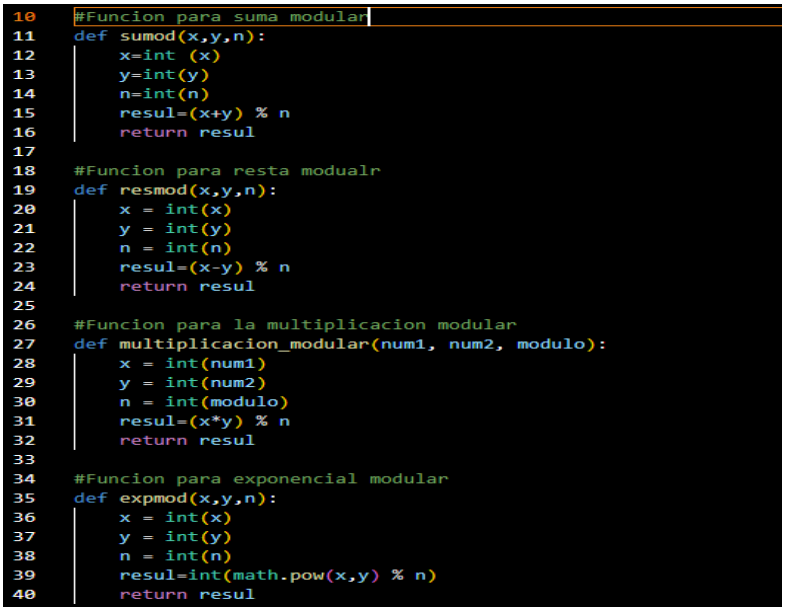
INSTITUTO POLITÉCNICO NACIONAL

“ESCUELA SUPERIOR DE CÓMPUTO”



```python
#Funcion para suma modular
def sumod(x,y,n):
    x=int (x)
    y=int(y)
    n=int(n)
    resul=(x+y) % n
    return resul

#Funcion para resta modualr
def resmod(x,y,n):
    x = int(x)
    y = int(y)
    n = int(n)
    resul=(x-y) % n
    return resul

#Funcion para la multiplicacion modular
def multiplicacion_modular(num1, num2, modulo):
    x = int(num1)
    y = int(num2)
    n = int(modulo)
    resul=(x*y) % n
    return resul

#Funcion para exponencial modular
def expmod(x,y,n):
    x = int(x)
    y = int(y)
    n = int(n)
    resul=int(math.pow(x,y) % n)
    return resul
```

“RESOLUTOR DE PROBLEMAS MODULARES”

UNIDAD ACADÉMICA: MATEMÁTICAS DISCRETAS

ALUMNOS:

- Gregorio Ramirez Luz Arely
- Miranda Hernandez Zaira Fernanda
- Pineda Godinez Abel
- Silva Lucio Ruben

GRUPO: 1CM5

Documentación: Resolutor de Problemas Modulares

Objetivo del Proyecto

Diseñar y desarrollar una herramienta interactiva digital que facilite a los usuarios la ejecución de operaciones modulares básicas (como suma, resta, multiplicación y exponenciación modular), la resolución de ecuaciones modulares lineales y la representación visual de los resultados.

Descripción de proceso del proyecto

-OPERACIONES ARITMÉTICAS MODULARES BÁSICAS

SUMA

Este código implementa una función para realizar la suma modular, una operación matemática que suma dos números y obtiene el residuo al dividir la suma por un módulo especificado. La estructura del programa está diseñada de manera clara y robusta para garantizar la correcta ejecución incluso si el usuario proporciona entradas no válidas. El programa consta de dos funciones principales:

`suma_modular`: Esta función realiza la operación principal. Toma tres parámetros: `num1` y `num2`, que son los números a sumar, y `módulo`, que define el rango del resultado. La operación que realiza es sencilla: (num1+num2)mod modulo(num1 + num2) \mod modulo(num1+num2)mod modulo, asegurando que el resultado siempre esté dentro del rango definido por el módulo.

`main`: Es la función principal que coordina la interacción con el usuario. Primero, solicita los valores de entrada: los dos números (`num1` y `num2`) y el módulo (`módulo`). Luego valida que el módulo sea un número positivo; si no lo es, muestra un mensaje de error y detiene la ejecución. A continuación, llama a la función `suma_modular` con los valores proporcionados y muestra el resultado al usuario. El programa también maneja entradas inválidas mediante un bloque `try-except`, que muestra un mensaje amigable si el usuario introduce algo que no sea un número.

RESTA

El código implementa una sección dentro de las diferentes operaciones, que es la resta, dado dos números el minuendo y el sustraendo y su respectivo mod, con el cual se trabajará.

La función resmod se utiliza en la sección del programa que permite al usuario realizar operaciones básicas de forma manual. Esto ocurre cuando el usuario selecciona la opción correspondiente en el menú principal.

Al igual que la función suma, este funciona exactamente igual, sin embargo se toma en cuenta que el signo de la operación cambia.

MULTIPLICACIÓN

Este código implementa una función para realizar la multiplicación modular básica, que calcula el residuo de la división de un producto entre un módulo dado. El objetivo principal del código es permitir realizar operaciones aritméticas en sistemas modulares de manera eficiente y precisa. La multiplicación modular es especialmente útil en escenarios donde se manejan números muy grandes, ya que ayuda a evitar problemas como el desbordamiento numérico, al limitar los valores del sistema. El diseño del código está estructurado de manera modular, lo que permite encapsular la lógica necesaria para calcular $(a \cdot b) \bmod n$.

Primero tenemos la función que va a calcular la multiplicación modular, lo mejor era encapsular esta operación en una función llamada multiplicacion_modular, para que fuera fácil de reutilizar. Esta función toma tres valores: los dos números a multiplicar y el módulo que define el rango. Luego, dentro de la función, simplemente se multiplicarán los dos números y se utilizará la operación módulo (%) para obtener el residuo, que es el resultado de la operación modular, por último devolvimos este valor como salida de la función.

En conclusión, esta parte del código no solo facilita la ejecución de operaciones modulares básicas, sino que también puede adaptarse fácilmente a aplicaciones más complejas, gracias a su diseño eficiente, reutilizable y robusto.

EXPONENCIACIÓN MÓDULO N

Esta es la función principal que realiza la operación. Toma tres parámetros:

- **base**: El número que será elevado a una potencia.
- **exponente**: La potencia a la que se elevará la base.
- **módulo**: Define el rango en el que debe encontrarse el resultado.

El algoritmo implementado utiliza la técnica de *exponenciación rápida modular* (o *exponentiation by squaring*), que es altamente eficiente, ya que reduce significativamente el número de multiplicaciones necesarias. El proceso es el siguiente:

1. Inicializa el resultado en 1.
2. Mientras el exponente sea mayor que 0, realiza las siguientes operaciones:
    - Si el exponente es impar, multiplica el resultado actual por la base y calcula el módulo.
    - Divide el exponente entre 2 (división entera) y eleva la base al cuadrado, calculando nuevamente el módulo.
3. Cuando el exponente llega a 0, el resultado contiene el residuo de la operación modular.

Finalmente, la función devuelve este resultado, garantizando que siempre esté dentro del rango definido por el módulo.

Visualización de Resultados

1) Visualización de resultados por entrada manual:

El programa solicita valores (x,y,n) y un operador (+,-,*,^). Según la operación elegida. Al hacerlo de forma manual solo obtienes una salida por consola con el resultado.

2) Visualización Gráfica: Residuos Módulo n:

   Después de realizar cualquier operación modular, el programa genera una representación gráfica de los residuos módulo n:

   Círculo de Residuos, los residuos (0,1,2,...,n−1) se distribuyen uniformemente sobre un círculo.

   Cada residuo se representa como un punto azul, con etiquetas que indican su valor.

   Por ejemplo, n=5, los residuos serán 0, 1, 2, 3 y 4, y el gráfico mostrará estos valores alrededor de un círculo.

3) Procesamiento de Archivos CSV:

   Si el usuario opta por usar un archivo CSV:

   Lectura: El programa lee las filas del archivo, donde cada fila representa una operación.

   Cálculo: Realiza las operaciones especificadas (basadas en el operador) y agrega los resultados en una nueva columna.

   En archivo: Sobrescribe el archivo original o crea uno nuevo con los resultados en una columna adicional llamada "Resultados".

Archivo CSV

Para que el programa procese el archivo CSV correctamente, este debe cumplir con las siguientes condiciones:

Encabezados

El archivo debe incluir un encabezado en la primera fila que especifique las columnas necesarias. Los encabezados requeridos son:

x: Primer operando (un número entero).

operador: El operador a utilizar, puede ser uno de los siguientes:

+ para la suma modular.

- para la resta modular.

* para multiplicación modular.

^ para exponenciación modular.

y: Segundo operando (un número entero).

n: El módulo bajo el cual se realiza la operación (un número entero).

Filas de Datos

Cada fila después del encabezado debe contener exactamente cuatro valores:

Columna x: Un número entero, que es el primer operando.

Columna operador: Uno de los operadores válidos (+, -, *, ^).

Columna y: Un número entero, que es el segundo operando.

Columna n: Un número entero mayor que 0, que representa el módulo.

Restricciones

Datos completos: No debe haber valores en blanco o nulos en las columnas x, operador, y o n.

Valores válidos: Los valores deben ser numéricos enteros, y el operador debe ser uno de los permitidos.

Formato CSV: El archivo debe estar delimitado por comas (,) y guardar con la extensión .csv

Entrada de Datos

- Entrada desde Archivo CSV

El programa puede leer un archivo CSV que contiene datos de operaciones modulares. Cada fila en el archivo representa una operación, y las columnas incluyen los siguientes valores:

1- x: Primer operando.

2- *operador*: Operación a realizar (+, -, *, ^).

3- y: Segundo operando.

4- n: Módulo bajo el cual se realizará la operación.

Funcionamiento en el código:

El programa solicita al usuario la ruta y nombre del archivo CSV. Luego se utiliza la librería csv para leer el contenido.

Para esta parte del código, dentro de las líneas 141-145, lo que se le pide al usuario es mostrar la dirección en la cual se encuentra la dirección del documento, con su respectivo nombre, después genera estos procesos en base a la línea del código:

1- open(ruta, mode="r"): Abre el archivo CSV en modo de lectura.

2- csv.reader(archivo): Permite leer el archivo fila por fila.

3- next(lectura): Lee y almacena la primera fila como encabezado, ya que representa los nombres de las columnas.

Salidas de Datos

-Salida de información desde un archivo CSV

El programa escribe un nuevo archivo CSV que contiene los mismos datos de entrada junto con una columna adicional llamada Resultados, donde se almacenan los resultados de las operaciones modulares realizadas.

Funcionamiento en el código:

El programa utiliza la librería csv para escribir los resultados en el archivo, añadiendo la nueva columna de resultados.

Esto lo hace en las líneas del código 147-172, haciéndolo de esta forma:

1- open(ruta, mode="w"): Abre el archivo CSV en modo de escritura.

2- escritor.writerow(campos): Escribe el encabezado con la columna extra de Resultados.

3- row.append(resul): Añade el resultado de la operación modular al final de cada fila.

4- escritor.writerow(row): Escribe la fila con el resultado incluido en el archivo.

Finalmente imprime los resultados utilizando las líneas de código 173 y 174, para las que únicamente se imprime un comentario y el resultado de los residuos modulares.

Consideraciones Técnicas

La lectura y la validación del archivo CSV

Agregar funciones para la correcta lectura de los archivos CSV y ver de manera correcta su contenido

Desarrollar algoritmos para realizar operaciones aritméticas modulares y resolver

Exportación de Resultados en Formato CSV

Conclusiones

Este proyecto demuestra la importancia de la aritmética modular como una herramienta esencial en matemáticas y programación, al mismo tiempo que facilita su comprensión. La función de multiplicación modular básica no solo permite realizar cálculos eficientes y precisos, sino que también sienta las bases para aplicaciones más avanzadas, como la criptografía, los algoritmos de hashing y la manipulación de grandes volúmenes de datos numéricos.

Además, al integrar elementos como el procesamiento de datos y la visualización interactiva, el proyecto se posiciona como un recurso educativo y práctico para entender los fundamentos de la modularidad en diversos contextos. Esto refuerza la utilidad de las matemáticas computacionales para resolver problemas complejos, fomentar la lógica algorítmica y explorar nuevas aplicaciones en tecnología y ciencia de datos.