GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: REGLAS Y AUTÓMATA DE DIVISIBILIDAD

1. Introducción

Por muy trivial que parezca este asunto, ha sido bastante explotado en un buen número de problemas de concursos e inclusos en competencias. Y porque ?. Si para saber si un número a es divisible por otro número b solo basta con comprobar si el resto de la división es cero. Donde está entonces la dificultad del problema ?.

Resulta que el el 100 % de los problemas donde se necesita verificar si $a \ mod \ b = 0$ el número a tiene una cantidad de dígitos no puede ser almacenado por ningún de los tipos de datos númericos enteros conocidos. Entonces que hacer ?.

2. Conocimientos previos

2.1. Autómata finito

Un autómata finito (AF) o máquina de estado finito es un modelo computacional que realiza cómputos en forma automática sobre una entrada para producir una salida.

Este modelo está conformado por un alfabeto, un conjunto de estados finito, una función de transición, un estado inicial y un conjunto de estados finales. Su funcionamiento se basa en una función de transición, que recibe a partir de un estado inicial una cadena de caracteres pertenecientes al alfabeto (la entrada), y que va leyendo dicha cadena a medida que el autómata se desplaza de un estado a otro, para finalmente detenerse en un estado final o de aceptación, que representa la salida.

3. Desarrollo

Solucionar este tipo de problema tiene diferentes enfoques de solución los cuales analizaremos a continuación.

3.1. Usando BigInteger

El lenguaje de programación de Java nos ofrece la clase BigInteger para operar y manipular números grandes lo que haría que la solución a este tipo de problema con el uso de este lenguaje de programación sea bastante sencilla.

De igual forma con el lenguaje de Python sería muy trivial la solución de este ejercicio ya que el mismo soporta los números grandes de forma nativa. Pero como resolver el problema cuando las restricciones de tiempo y memoria del hace que las soluciones con el uso de estos lenguajes no es factibles. Veremos que en el próximo apartado daremos una otra variante más eficiente.

3.2. Reglas de divisibilidad

Una regla de divisibilidad es un conjunto de criterios o condiciones que te permiten determinar si un número es divisible por otro sin necesidad de realizar la división en sí. Estas reglas son útiles

para simplificar el proceso de división y para identificar rápidamente si un número es divisible por otro.

Estas reglas son útiles en matemáticas, especialmente en aritmética básica y en álgebra, ya que te permiten determinar rápidamente si un número es divisible por otro sin tener que realizar la división completa. Esto puede ahorrarte tiempo y facilitar el trabajo con números grandes.

Es importante recordar que las reglas de divisibilidad son herramientas útiles, pero no reemplazan la división real cuando se requiere una respuesta precisa.

A continuación algunas reglas de divisibilidad comunes:

- **Divisibilidad por 2:** Un número es divisible por 2 si su último dígito es par (0, 2, 4, 6, 8).
- **Divisibilidad por 3:** Un número es divisible por 3 si la suma de sus dígitos es divisible por 3.
- **Divisibilidad por 4:** Un número es divisible por 4 si los dos últimos dígitos forman un número divisible por 4.
- **Divisibilidad por 5:** Un número es divisible por 5 si su último dígito es 0 o 5.
- **Divisibilidad por 6:** Un número es divisible por 6 si es divisible por 2 y por 3.
- **Divisibilidad por 7:** Cuando la diferencia entre el número sin la cifra de las unidades y el doble de la cifra de las unidades es 0 ó múltiplo de 7.
- **Divisibilidad por 8:** Si sus tres últimas cifras son ceros o múltiplo de 8.
- **Divisibilidad por 9:** Un número es divisible por 9 si la suma de sus dígitos es divisible por 9.
- **Divisibilidad por 10:** Un número es divisible por 10 si su último dígito es 0.
- **Divisibilidad por 11:** Un número es divisible por 11 si la diferencia entre la suma de los dígitos en posiciones impares y la suma de los dígitos en posiciones pares es un múltiplo de 11. Por ejemplo, el número 4184 es divisible por 11 porque (4 + 8) - (1 + 4) = 7, que es un múltiplo de 11.
- **Divisibilidad por 13:** Para determinar si un número es divisible por 13, puedes restarle cuatro veces el último dígito al resto del número. Si el resultado es divisible por 13, entonces el número original también lo es. Por ejemplo, para comprobar si 156 es divisible por 13, calculamos 15 - (6 * 4) = 15 - 24 = -9, que es divisible por 13.
- **Divisibilidad por 17:** Una regla similar a la divisibilidad por 13 se aplica para determinar si un número es divisible por 17. Resta cinco veces el último dígito al resto del número y si el resultado es divisible por 17, entonces el número original también lo es.

Si fuera el primer caso el valor b es fijo siempre es el mismo realizamos el siguiente análisis. Analizamos si el valor b es un valor que tiene definida su regla de divisibilidad. De ser así implementamos un algoritmo que compruebe la regla de divisibilidad del valor b en la cadena donde esta almacenado el valor a.

En caso de que el valor b no tenga regla de divisibilidad conocida entonces lo descomponemos en factores que tengan definido sus reglas de divisibilidad y luego solo debemos comprobar que la cadena cumpla con cada una de las reglas de divisibilidad de cada factor. De cumplir con todas entonces podemos afirmar que el valor a es divisible por el valor b. Pero que hacer cuando esto no es posible ?

3.3. Autómata de divisibilidad

El autómata de divisibilidad es un tipo de autómata finito que se utiliza para determinar si un número es divisible por otro. Este tipo de autómata opera sobre secuencias de dígitos que representan números enteros y permite analizar propiedades de divisibilidad de manera eficiente.

La estructura básica de un autómata de divisibilidad consta de un conjunto finito de estados, una función de transición que determina cómo se mueve el autómata entre los estados, un estado inicial y uno o más estados finales que indican si el número analizado cumple con la propiedad de divisibilidad deseada.

El funcionamiento del autómata de divisibilidad se basa en la idea de simular la división entre el número que se desea comprobar si es divisible y el divisor, siguiendo un proceso determinístico. Cada estado del autómata representa un paso en este proceso de división simulada, y la función de transición determina cómo se mueve el autómata en función de los dígitos del número que está siendo analizado.

A medida que el autómata avanza a través de los dígitos del número, se evalúa si el número es divisible por el divisor en cada paso. Si el autómata alcanza un estado final al finalizar la secuencia de dígitos, entonces se concluye que el número es divisible por el divisor. En caso contrario, se determina que el número no es divisible.

Los autómatas de divisibilidad pueden ser diseñados para verificar diferentes propiedades de divisibilidad, como la divisibilidad por un número primo, por un número compuesto, por un conjunto específico de números, entre otros. Estos autómatas son útiles en diversas aplicaciones donde se requiere determinar la divisibilidad de números de forma eficiente y automática.

Esta variante es más efeciente aunque tiene un grado análisis mayor que el anterior. Es la variante que deberían utilizar aquellos amantes de C++.

El primer paso es almacenar el número a en una variable de tipo cadena *string* no hay otra variante. Lo siguiente es ver quien es b si es un valor fijo siempre o es una valor variable para cada caso.

Bueno en esta caso podemos aplicar un autómata de divisibilidad. Un autómata de divisibilidad permite saber si un número X con una gran cantidad de digitos expresado en una base Y es divisible por un número Z el cual esta en base decimal. Extrapolando lo anterior a nuestro problema a es X 10 es Y y b es Z.

4. Implementación

4.1. C++

```
int value(char _symbol){
   if('0'<= _symbol && _symbol<='9') return (_symbol-'0');
   if('A'<= _symbol && _symbol <='F') return (_symbol-'A')+10;
   if('a'<= _symbol && _symbol <='f') return (_symbol-'a')+10;
   return 0;
}

bool automatDivisible(string _number,int _base, int _decimal){
   int state=0;
   int size=_number.size();
   for(int i=0;i<size;i++)
      state=(state*_base+value(_number[i]))%_decimal;
   if(!state)return true;
   else return false;
}
```

4.2. Java

```
public static int value(char _symbol){
   if('0'<= _symbol && _symbol<='9') return (_symbol-'0');
   if('A'<= _symbol && _symbol <='F') return (_symbol-'A')+10;
   if('a'<= _symbol && _symbol <='f') return (_symbol-'a')+10;
   return 0;
}

public static boolean automatDivisible(String _number,int _base, int _decimal)
    {
   int state=0;
   int size=_number..length();
   for(int i=0;i<size;i++)
      state=(state*_base+value(_number.charAt(i)))%_decimal;
   if(state==false)return true; else return false;
}
```

5. Aplicaciones

Las reglas de divisibilidad tienen diversas aplicaciones en matemáticas y en la vida cotidiana. Algunas de las aplicaciones más comunes son:

1. **Simplificación de fracciones:** Las reglas de divisibilidad pueden ayudarte a simplificar fracciones de manera más rápida y sencilla. Por ejemplo, si identificas que tanto el numerador

como el denominador de una fracción son divisibles por un mismo número, puedes reducir la fracción dividiendo ambos términos por ese número.

2. **Identificación de números primos:** Las reglas de divisibilidad te permiten identificar rápidamente si un número es primo o no. Por ejemplo, si aplicas la regla de divisibilidad por 2 y el número no es divisible por 2, entonces sabes que no es un número primo.

3. **Verificación de cálculos:** Las reglas de divisibilidad pueden ser útiles para verificar la precisión de cálculos matemáticos. Por ejemplo, al multiplicar números grandes, puedes aplicar las reglas de divisibilidad para comprobar si el resultado es correcto sin tener que realizar la multiplicación completa nuevamente.

4. **Criptografía y seguridad informática:** En el campo de la criptografía y la seguridad informática, las reglas de divisibilidad se utilizan en algoritmos de encriptación y descifrado para garantizar la seguridad de la información.

5. **Organización de datos:** En ciertas áreas como la informática y la estadística, las reglas de divisibilidad se utilizan para organizar y clasificar datos de manera eficiente, especialmente cuando se trabaja con conjuntos numéricos grandes.

En resumen, las reglas de divisibilidad son herramientas matemáticas versátiles que tienen aplicaciones prácticas en diversos campos, desde simplificar cálculos hasta garantizar la seguridad de la información. Su uso puede facilitar el trabajo con números y mejorar la eficiencia en diferentes situaciones.

El autómata de divisibilidad tiene diversas aplicaciones en el campo de la informática y las matemáticas. Algunas de las aplicaciones del autómata de divisibilidad son:

1. **Verificación de divisibilidad:** El autómata de divisibilidad se puede utilizar para verificar si un número es divisible por otro sin necesidad de realizar la división de forma tradicional. Esto puede ser útil en situaciones donde se necesite comprobar la divisibilidad de números de manera eficiente.

2. **Generación de secuencias:** El autómata de divisibilidad también se puede utilizar para generar secuencias de números que cumplan ciertas propiedades de divisibilidad. Por ejemplo, se pueden generar secuencias de números primos o números compuestos utilizando autómatas de divisibilidad específicos.

3. **Criptografía:** En el campo de la criptografía, los autómatas de divisibilidad pueden utilizarse en la generación de claves criptográficas o en la implementación de algoritmos criptográficos. Por ejemplo, algunos algoritmos criptográficos utilizan propiedades de divisibilidad para garantizar la seguridad de las comunicaciones.

4. **Optimización de algoritmos:** En algunos casos, el uso de autómatas de divisibilidad puede ayudar a optimizar algoritmos que involucren operaciones aritméticas. Al utilizar propiedades de divisibilidad, es posible simplificar cálculos y mejorar la eficiencia de los algoritmos.

Estas son solo algunas de las aplicaciones del autómata de divisibilidad en diferentes campos. Su versatilidad y capacidad para analizar propiedades numéricas lo convierten en una herramien-

ta útil en diversas áreas de la informática y las matemáticas.

6. Complejidad

La implementación de funciones que comprueben las reglas de divisibilidad tienen una complejidad en el rango de O(1) a O(N) dependiendo de la regla que se chequee. En el caso del autómata de divisibilidad la complejidad O(N).

7. Ejercicios

A continuación una lista de ejercicios que se pueden resolver aplicando los contenidos abordados en la guía:

- DMOJ - Div 6