Reporte Complejidades

Constructora a partir de una string

La complejidad de esta operación es O(n), donde n es el tamaño de la cadena, pues hace un recorrido por toda la cadena y los añade a un vector usando push_back, y esta operación tiene complejidad O(1), por lo que no afecta la complejidad total.

Constructora a partir de otro BigInteger

Tiene complejidad O(n), donde n es la cantidad de dígitos del BigInteger, pues recorre una vez todo el vector de dígitos.

Función getDigit

Tiene complejidad O(1), porque acceder a un elemento en una posición específica de un vector, se hace en tiempos constantes.

Función getSize

Tiene complejidad O(1), porque el tamaño de un vector se obtiene en tiempo constante.

Sobrecarga < y <=

En el mejor de los casos estas operaciones tienen complejidad O(1) y se da cuando los tamaños de los BigInteger son diferentes.

En el peor de los casos tiene complejidad O(n), donde n es la cantidad dígitos del número y se da cuando el primer número es menor o menor o igual, dependiendo del operador.

Sobrecarga ==

En el mejor de los casos tiene complejidad O(1), y se da cuando los tamaños son diferentes o sus signos son diferentes.

En el peor de los casos tiene complejidad O(n) y se da cuando los dos números son iguales, donde n es el número de dígitos.

Función toString

Tiene complejidad O(n), donde n es el número de dígitos, pues se hace un recorrido por todo el número.

Operación sumIguales y sumDiff

Tienen complejidad O(n), donde n es el número de dígitos del num1. Esto debido a que se hace un recorrido por todo el vector y todas las operaciones usadas, tienen complejidad O(1).

Operación quitar ceros.

En el mejor de los casos tiene complejidad O(1) y se da cuando no hay ningún cero a la izquierda.

En el peor de los casos tiene complejidad O(n) donde n es el tamaño del vector de dígitos y se da cuando todos los elementos del vector son iguales a 0.

Operación add

Tiene complejidad O(n), ya que no hay ciclos anidados, pero los operaciones que usa y llama, tienen como complejidad O(n).

Operación cambiarSigno

Tiene complejidad $O(1)$, puesto que accede a la última posición del vector, la cual da el signo y lo multiplica por -1.

Operación substract

Tiene complejidad $O(n)$, puesto que las operaciones que usa, a excepción de add, tienen complejidad $O(1)$, entonces sumarle la complejidad de add, es decir $O(n)$, queda como complejidad $O(n)$.

Operación product

Tiene complejidad $O(n \cdot m)$ donde el tamaño del num y m el tamaño del entero al cual se le aplica la operación. Es decir, que tiene complejidad cuadrática.

Operación addDigit

Tiene complejidad $O(n)$, donde n es el tamaño del vector, pues hay que mover todos los elementos una posición a la derecha

Operación divMod

Es una operación bastante ineficiente, ya que hace muchos recorridos sobre el número. Además, hace un llamado a operaciones muy ineficientes. Con todo esto, tiene complejidad $O(n \cdot m)$ donde n es el tamaño del número sobre el cual se aplica la función y m la cantidad de dígitos final de div.

Operación quotient y remainder

Tiene la misma complejidad que divMod, en el peor de los casos, es decir que corre en tiempo cuadrático y se da cuando el valor absoluto del del segundo número es menor que el del primer número. En el mejor de los casos se da en complejidad $O(n)$ y se da cuando el segundo número es mayor o igual que el primer número, pues

a pesar de que no se usa la operación divMod, sí se usa el clear

Sobrecargas

Todas las sobrecargas tiene la mismo complejidad que las operaciones previamente mencionadas. Eso sí, se les agrega un recorrido, al copiar en todas un BigInteger.

Función sumarListaValores

Tiene complejidad $O(n^2)$, debido a que recorre la lista y en cada momento, hace una suma, operación que tiene complejidad $O(n)$, por lo que la función ejecuta esto $n \cdot n$ veces, donde n es el tamaño de la lista.

Función multiplicarListaValores

Tiene complejidad $O(n^3)$, debido a que llama a la función product que tiene complejidad cuadrática, por lo que al recorrer toda la lista llama a esa operación n veces, donde n es el tamaño de la lista. Entonces se ejecuta $n \cdot n^2$ veces, lo que da la complejidad dada.

Operación pow

Tiene complejidad $O(n^3)$, donde n es el número de la potencia, puesto que ejecuta n veces, la operación product, la cual tiene complejidad cuadrática, lo que forma la complejidad cúbica.