ANALISIS CLASE BINARIA

Funcion	Complejidad espacial	Complejidad temporal
Cosntructor binario(long valor)	O(1)	O(log n)
Operador de suma binario operator+(const binario &op2) const	O(1)	O(n)
Operador de salida ostream& operator<<(ostream &salida, const binario &num)	O(1)	O(n)

Constructor binario

análisis espacial:

- La variable bin ocupa 40 elementos de tipo short, lo que equivale a 40 * 2 bytes = 80 bytes de memoria.
- La variable valor ocupa 4 bytes de memoria (asumiendo un long de 32 bits).
- En total, el constructor utiliza 80 + 4 = 84 bytes de memoria.

Análisis temporal:

- El bucle for que inicializa bin se ejecuta 40 veces, con una complejidad temporal de O(n).
- El bucle for que convierte el valor decimal a binario se ejecuta en promedio log2(valor) veces, con una complejidad temporal de O(log n).
- La asignación bin[j] = valor % 2; se realiza log2(valor) veces.
- La división valor /= 2; se realiza log2(valor) veces.
- En total, la complejidad temporal del constructor es O(log n).

instrucción	Complejidad espacial	Complejidad temporal
binario(long valor) (Declaración de constructor)	O(1)	O(1)
for (int i = 0; i <= 39; i++)	O(1)	O(n)
bin[i] = 0;	O(1)	O(n)
for (int j = 39; valor != 0 && j >= 0; j--)	O(1)	O(log n)
bin[j] = valor % 2;	O(1)	O(log n)
valor /= 2;	O(1)	O(log n)

Operador de suma

Análisis espacial:

- La variable temp ocupa 40 elementos de tipo short, lo que equivale a 80 bytes de memoria.
- La variable acarreo ocupa 2 bytes de memoria (asumiendo un short).
- En total, el operador de suma utiliza 80 + 2 = 82 bytes de memoria.

Análisis temporal:

- El bucle for que realiza la suma binaria se ejecuta 40 veces, con una complejidad temporal de O(n).
- La operación temp.bin[i] = bin[i] + op2.bin[i] + acarreo; se realiza 40 veces.
- La operación temp.bin[i] %= 2; se realiza en promedio 1 vez por cada 2 iteraciones del bucle, con una complejidad temporal de O(n/2).
- La operación acarreo = 1; se realiza en promedio 1 vez por cada 2 iteraciones del bucle, con una complejidad temporal de O(n/2).
- En total, la complejidad temporal del operador de suma es O(n).

Instrucción	Complejidad espacial	Complejidad temporal
binario operator+(const binario &op2) const (Declaración de operador)	O(1)	O(1)
binario temp;	O(1)	O(1)
int acarreo = 0;	O(1)	O(1)
for (int i = 39; i >= 0; i--)	O(1)	O(n)
temp.bin[i] = bin[i] + op2.bin[i] + acarreo;	O(1)	O(n)
if (temp.bin[i] > 1)	O(1)	O(n)
temp.bin[i] %= 2;	O(1)	O(n/2)
acarreo = 1;	O(1)	O(n/2)
return temp;	O(1)	O(1)

Operador de salida

Análisis espacial:

- La variable i ocupa 4 bytes de memoria (asumiendo un int).
- En total, el operador de salida utiliza 4 bytes de memoria.

Análisis temporal:

- El bucle for que busca el primer dígito no nulo se ejecuta en promedio 39 veces, con una complejidad temporal de O(n).
- La operación salida << num.bin[i]; se realiza en promedio 39 veces, con una complejidad temporal de O(n).
- En total, la complejidad temporal del operador de salida es O(n).

instrucción	Complejidad espacial	Complejidad temporal
ostream& operator<<(ostream &salida, const binario &num) (Declaración de operador)	O(1)	O(1)
int i;	O(1)	O(1)
for (i = 0; (num.bin[i] == 0) && (i <= 39); i++);	O(1)	O(n)
if (i == 40)	O(1)	O(1)
salida << 0;	O(1)	O(1)
else	O(1)	O(1)
for (; i <= 39; i++)	O(1)	O(n)
salida << num.bin[i];	O(1)	O(n)
return salida;	O(1)	O(1)