CLASE BINARIA CON LISTA DOBLEMENTE ENLAZADA: Complejidad Espacial y Temporal

Función	Complejidad temporal	Complejidad espacial
binario::binario(long valor)	O(n)	O(n)
binario::operator+(const binario &op2)	O(n + m)	O(n + m)
operator<<(ostream &salida, const binario &num)	O(n)	O(n)

Binario:binario(long valor)

instrucción	Complejidad temporal	Complejidad espacial
Bin=lista()	O(1)	O(1)
while (valor != 0)	O(log n)	O(log n)
bin.insertarAlInicio(valor % 2);	O(1)	O(n)
valor /= 2;	O(1)	O(n)
if (bin.getPrim() == nullptr)	O(1)	O(n)
bin.insertarAlInicio(0);	O(1)	O(n)

binario::operator+(const binario &op2)

instrucción	Complejidad temporal	Complejidad espacial
binario temp;	O(1)	O(1)
int acarreo = 0;	O(1)	O(1)
string sumaString;	O(1)	O(1)
pnodo nodo1 = bin.getUltm();	O(1)	O(1)
pnodo nodo2 = op2.bin.getUltm();	O(1)	O(1)
while (nodo1 != nullptr || nodo2 != nullptr || acarreo != 0)	O(max(m, n)) //m y n son las longitudes de las listas	
int bit1 = (nodo1 != nullptr) ? nodo1->num : 0;	O(1)	O(n+m)
int bit2 = (nodo2 != nullptr) ? nodo2->num : 0;	O(1)	O(n+m)
int suma = bit1 + bit2 + acarreo;	O(1)	O(n+m)
sumaString += to_string(suma % 2);	O(1)	O(n+m)
acarreo = suma / 2;	O(1)	O(n+m)
if (nodo1 != nullptr) nodo1 = nodo1->ant;	O(1)	O(n+m)
if (nodo2 != nullptr) nodo2 = nodo2->ant;	O(1)	O(n+m)
string resultado;	O(1)	O(1)

for (int i = sumaString.size() - 1; i >= 0; --i)	O(max(m, n))	O(max(m, n))
resultado += sumaString[i];	O(1)	O(n+m)
std::cout << "\n\n" << "n1 + n2 = " << resultado;	O(n+m)	O(1)
return temp;	O(1)	O(1)

operator<<(ostream &salida, const binario &num)

Instrucción	Complejidad temporal	Complejidad espacial
pnodo p = num.bin.getPrim();	O(1)	O(1)
if (p == nullptr)	O(1)	O(1)
salida << "0";	O(1)	O(1)
return salida;	O(1)	O(1)
while (p != nullptr)	O(n)	O(n)
salida << p->num;	O(1)	O(n)
p = p->sig;	O(1)	O(n)
return salida;	O(1)	O(1)