

INTEGRANTES

Jhon Edwin Escudero Arias

Jhon David Copete Viatela

William Javier Jimenez Moran

Ejercicio #1

Instrucción	Costo	Cuántas veces se repite
boolean first = true;	C1	O(1)
String msg = "R = [";	C2	O(1)
for (int i = 1; i < L.length; i++)	C3	O(n)
if (L[i] == L[i - 1])	C4	O(n-1)
if (i == 1 L[i] != L[i - 2])	C5	O(n-1)
if (!first)	C6	O(n-1)
msg += ", ";	C7	O(n-1)
msg += L[i];	C8	O(n-1)
first = false;	C9	O(n-1)
msg += "];	C10	O(1)
return msg;	C11	O(1)

$$O(n) = C1 + C2 + C3n + C4(n-1) + C5(n-1) + C6(n-1) + C7(n-1) + C8(n-1) + C9(n-1) + C10 + C11$$

$$O(n) = C1 + C2 + C3n + C4n - C4 + C5n - C5 + C6n - C6 + C7n - C7 + C8n - C8 + C9n - C9 + C10 + C11$$

$$O(n) = (C3 + C4 + C5 + C6 + C7 + C8 + C9)n + (C1 + C2 + C10 + C11 - C4 - C5 - C6 - C7 - C8 - C9)$$

El término n es de mayor grado, por lo que el Big O de este algoritmo es:

$$O(n)$$

```

public static String se3_2(String palabra){

    int longitud = palabra.length();

    for (int i = 0; i < longitud / 2; i++) {

        if (palabra.charAt(i) != palabra.charAt(longitud - 1 - i))
        {

            return "No es un palindromo";

        }

    }

    return "Es un palindromo";

}

```

Ejercicio #2

Instrucción	Costo	Cuántas veces se repite
<code>int longitud = palabra.length();</code>	C1	O(1)
<code>for (int i = 0; i < longitud / 2; i++) {</code>	C2	O(n)
<code>if (palabra.charAt(i) != palabra.charAt(longitud - 1 - i))</code>	C3	O(n-1)
<code>return "No es un palindromo";</code>	C4	O(n-1)
<code>return "Es un palindromo";</code>	C5	O(1)

$$O(n) = C1 + C2 + C3(n-1) + C4(n-1) + C5$$

$$O(n) = C1 + C2 + C3n - C3 + C4n - C4 + C5$$

$$O(n) = (C2 + C3 + C4)n + (C1 + C5 - C3 - C4)$$

El término n es de mayor grado, por lo que el Big O de este algoritmo es:

$$O(n)$$

```

public static int se3_3 (String bits){
    int bit = 0;
    for (int i = 0; i < bits.length(); i++) {

```

```

        if (bits.charAt(i) == '1') {
            bit++;
        }
    }
    return bit;
}

```

Ejercicio #3

Instrucción	Costo	Cuántas veces se repite
int bit = 0;	C1	O(1)
for (int i = 0; i < bits.length(); i++) {	C2	O(n)
if (bits.charAt(i) == '1') {	C3	O(n-1)
bit++;	C4	O(n-1)
return bit;	C5	O(1)

$$O(n) = C1 + C2 + C3(n-1) + C4(n-1) + C5(1)$$

$$O(n) = (C1 + C5 - C3 - C4) + n(C2 + C3 + C4)$$

El término n es de mayor grado, por lo que el Big O de este algoritmo es:

$$O(n)$$