**INTEGRANTES**

Jhon Edwin Escudero Arias

Jhon David Copete Viatela

William Javier Jimenez Moran

**Ejercicio #1**

| **Instrucción** | **Costo** | **Cuantas veces se repite** |
| --- | --- | --- |
| for (int i = 0; i < L.length; i++) | C1 | O(n) |
| int minimum = i; | C2 | O(n-1) |
| for (int j = i + 1; j < L.length; j++) | C3 | O(n-1) |
| if (L[j] < L[minimum]) | C4 | O(n^2) |
| minimum = j; | C5 | O(n^2) |
| int temp = L[minimum]; | C6 | O(n-1) |
| L[minimum] = L[i]; | C7 | O(n-1) |
| L[i] = temp; | C8 | O(n-1) |
| boolean first = true; | C9 | O(1) |
| String msg = "R = ["; | C10 | O(1) |
| for (int i = 1; i < L.length; i++) | C11 | O(n) |
| if (L[i] == L[i - 1]) | C12 | O(n-1) |
| if (i == 1 || L[i] != L[i - 2]) | C13 | O(n-1) |
| if (!first) | C14 | O(n-1) |
| msg += ", "; | C15 | O(n-1) |
| msg += L[i]; | C16 | O(n-1) |
| first = false; | C17 | O(n-1) |
| msg += "]"; | C18 | O(1) |
| return msg; | C19 | O(1) |

O(n) = C1(n) + C2(n-1) + C3(n-1) + C4(n^2) + C5(n^2) + C6(n-1) + C7(n-1) + C8(n-1) + C9(1) + C10(1) + C11(n) C12(n-1) + C13(n-1) + C14(n-1) + C15(n-1) + C16(n-1) + C17(n-1) + C18(1) + C19(1)

O(n) = (C1 + C2 + C3 + C6 + C7+ C8 + C11 + C12 + C13 + C14 +C15 + C16 + C17)n + n^2(C4 + C5) + (C9 + C10 + C18 + C19 - C2 - C3 - C6 - C7 - C8 - C12 - C13 - C14 - C15 - C16 - C17)

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

O(n^2)

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

**O(n) = C1 + C2 + C3(n/2 + 1) +C4(n/2 + 1) +C5**

**O(n) = (C1 + C5 + C3 + C4) + n(C2 + C3 +C4) + ½(C2 + 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** | **Cuantas 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)