Actividad 2: Análisis de algoritmos.

Diego Javier Mora Cortés

26 de Octubre 2022

CUCEI - INGENIERÍA INFORMÁTICA

Algoritmia

Profesor: David Alejandro Gómez Anaya

Índice

1. Análisis de algoritmos.

1.1. Algoritmo 1

Escribir un algoritmo que reciba como entrada números enteros positivos, hora (0 a n), minutos (0 a 59) y segundos (0 a 59), e imprima esa hora minutos y segundos después de añadir un segundo.
Entrada: $h, m, s \in \mathbb{N}_0$
Salida: $\emptyset$

Algorithm 1 Hora después de un segundo

```
1: s ← s + 1 {2, el uno se puede reemplazar}
2: if s > 59 then
3:    aux ← s/60 {2, truncates}
4:    s ← s mod 60 {2}
5:    m ← m + aux {2}
6:    if m > 59 then
7:       aux = m/60 {2, truncates}
8:       m ← m mod 60 {2}
9:       h ← h + aux {2}
10:   end if
11: end if
12: IMPRIME(h,m,s, separador = ' ') {1}
```

Complejidad

$$T(n) = 17$$

$$T(n) \in O(1)$$

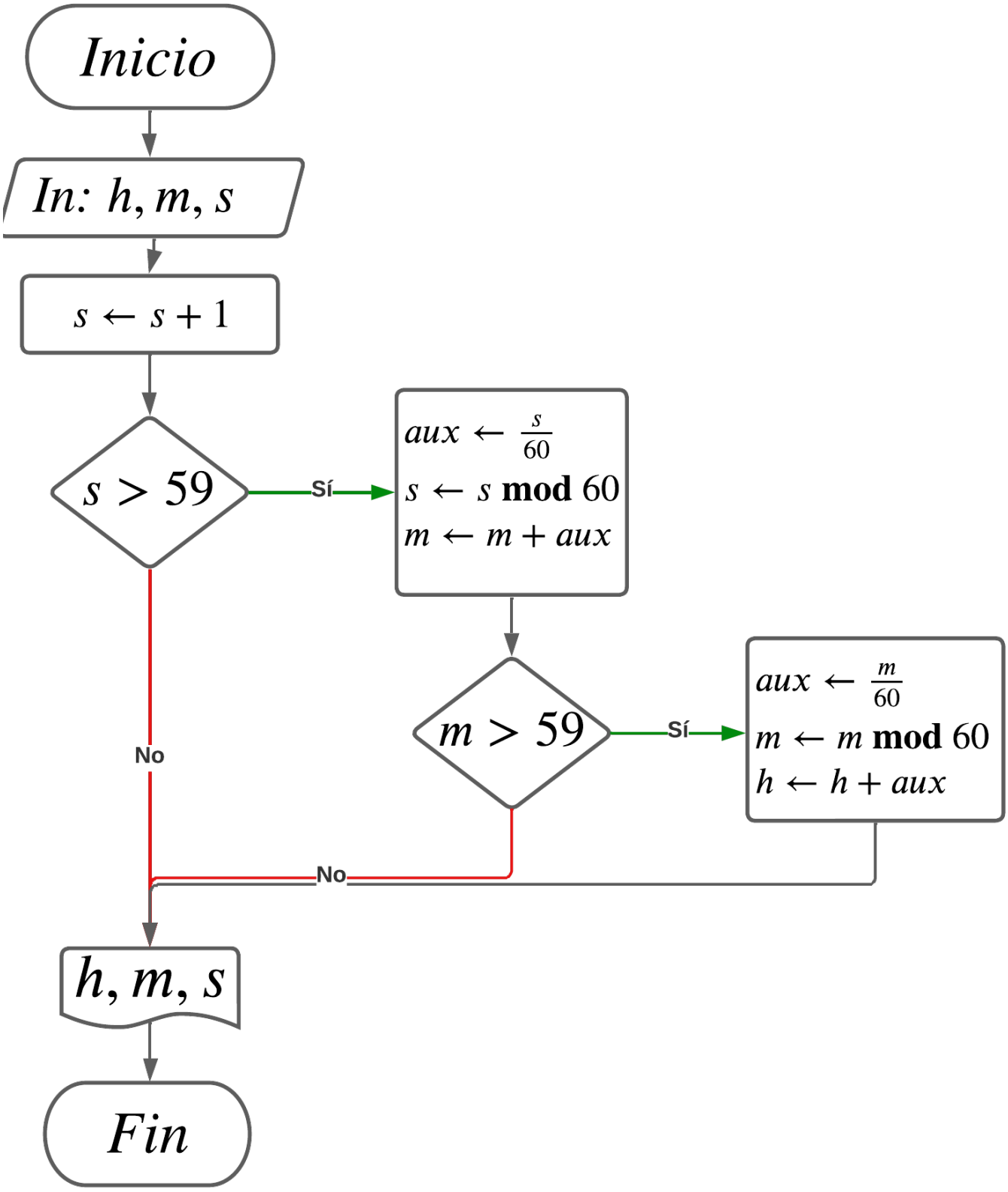

Figura 1: Diagrama de flujo de algoritmo 1

1.2. Algoritmo 2

Escribir un algoritmo que imprima los cubos de los primeros números naturales tales que el cubo sea menor que n. n es la entrada del algoritmo.
Entrada: $n \in \mathbb{N}$
Salida: imprime: $i^3 \mid i^3 < n \wedge i \in \mathbb{N}$

Algorithm 2 Primeros cubos naturales antes que n

1: $i \leftarrow 1$
2: $p \leftarrow 1$
3: **while** $p < n$ **do**
4: **IMPRIME** (p)
5: $i++$
6: $p \leftarrow i^3$ $\{3\}$
7: **end while**

Complejidad

$$T(n) = 2 + 1 + \sum_{j=1}^{\lceil \sqrt[3]{n} \rceil - 1} (6)$$

$$T(n) = 3 + (\lceil \sqrt[3]{n} \rceil - 1 - 1 + 1) \cdot 6$$
$$T(n) = 3 + (\lceil \sqrt[3]{n} \rceil - 1) \cdot 6$$
$$T(n) = 6\lceil \sqrt[3]{n} \rceil - 6 + 3$$
$$\boldsymbol{T(n) = 6\lceil \sqrt[3]{n} \rceil - 3}$$
$$\boldsymbol{T(n) \in O(n^{\frac{1}{3}})}$$

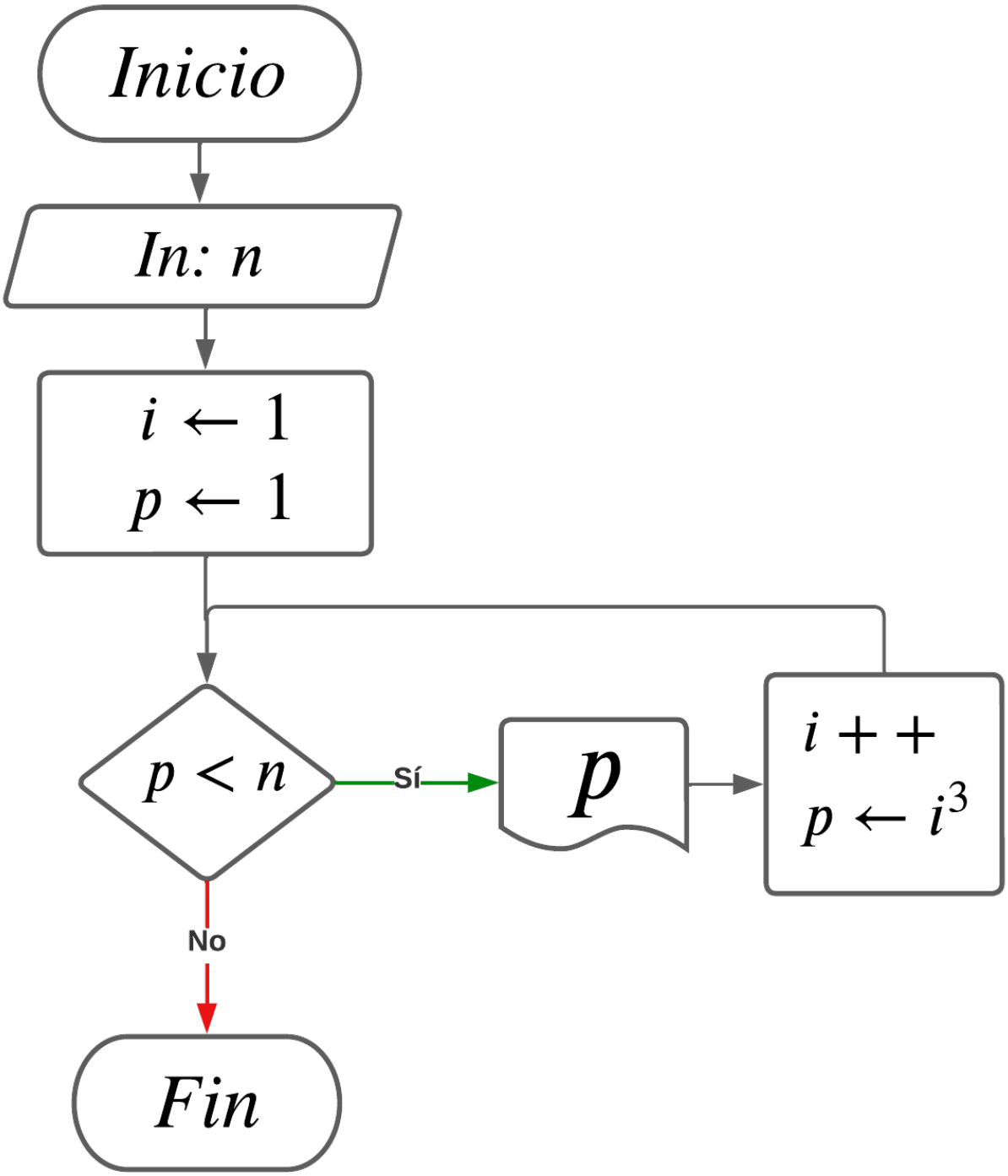

Figura 2: Diagrama de flujo de algoritmo 2

1.3. Algoritmo 3

Escriba un algoritmo que calcula el producto de dos matrices de NxN y devuelva el resultado.
Entrada: $a[n][n], b[n][n] \in \mathbb{Q}$
Salida: $c[n][n] \in \mathbb{Q}$

Algorithm 3 Producto de dos matrices $n \times n$

```
1: c[n][n]
2: for i = 0, i < n , i++ do
3:    for j = 0, j < n , j++ do
4:       c[i][j] = 0 {3}
5:       for k = 0, k < n , k++ do
6:          c[i][j]+ = a[i][k] · b[k][j] {6(accesos) + 3(aritmética y asignación) = 9}
7:       end for
8:    end for
9: end for
10: return c
```

Complejidad

$T_1(n)$: líneas 5-7

$$T_1(n) = 2 + \sum_{k=0}^{n-1}(2+9)$$

$$T_1(n) = 2 + \sum_{k=0}^{n-1}(11)$$

$$T_1(n) = 2 + (n-1-0+1)\cdot 11$$

$$T_1(n) = 11n + 2$$

$T_2(n)$: líneas 3-8

$$T_2(n) = 2 + \sum_{j=0}^{n-1}\Big(2 + 3 + T_1(n)\Big)$$

$$T_2(n) = 2 + \sum_{j=0}^{n-1}\Big(T_1(n) + 5\Big)$$

$$T_2(n) = 2 + \sum_{j=0}^{n-1}\Big(11n + 2 + 5\Big)$$

$$T_2(n) = 2 + \sum_{j=0}^{n-1}\Big(11n + 7\Big)$$

$$T_2(n) = 2 + (n-1-0+1)\cdot\Big(11n + 7\Big)$$

$$T_2(n) = 2 + (n)\cdot\Big(11n + 7\Big)$$

$$\boldsymbol{T_2(n) = 11n^2 + 7n + 2}$$

$$\boldsymbol{T(n) = 2 + 2 + \sum_{j=0}^{n-1} \Big(2 + T_2(n) \Big)}$$

$$T(n) = 4 + \sum_{i=0}^{n-1} \Big(2 + T_2(n) \Big)$$

$$T(n) = 4 + \sum_{i=0}^{n-1} \Big(2 + (11n^2 + 7n + 2) \Big)$$

$$T(n) = 4 + \sum_{i=0}^{n-1} \Big(11n^2 + 7n + 4 \Big)$$

$$T(n) = 4 + (n - 1 - 0 + 1) \cdot (11n^2 + 7n + 4)$$

$$T(n) = 4 + (n) \cdot (11n^2 + 7n + 4)$$

$$T(n) = 4 + (11n^3 + 7n^2 + 4n)$$

$$\boldsymbol{T(n) = 11n^3 + 7n^2 + 4n + 4}$$

$$\boldsymbol{T(n) \in \mathrm{O}(n^3)}$$

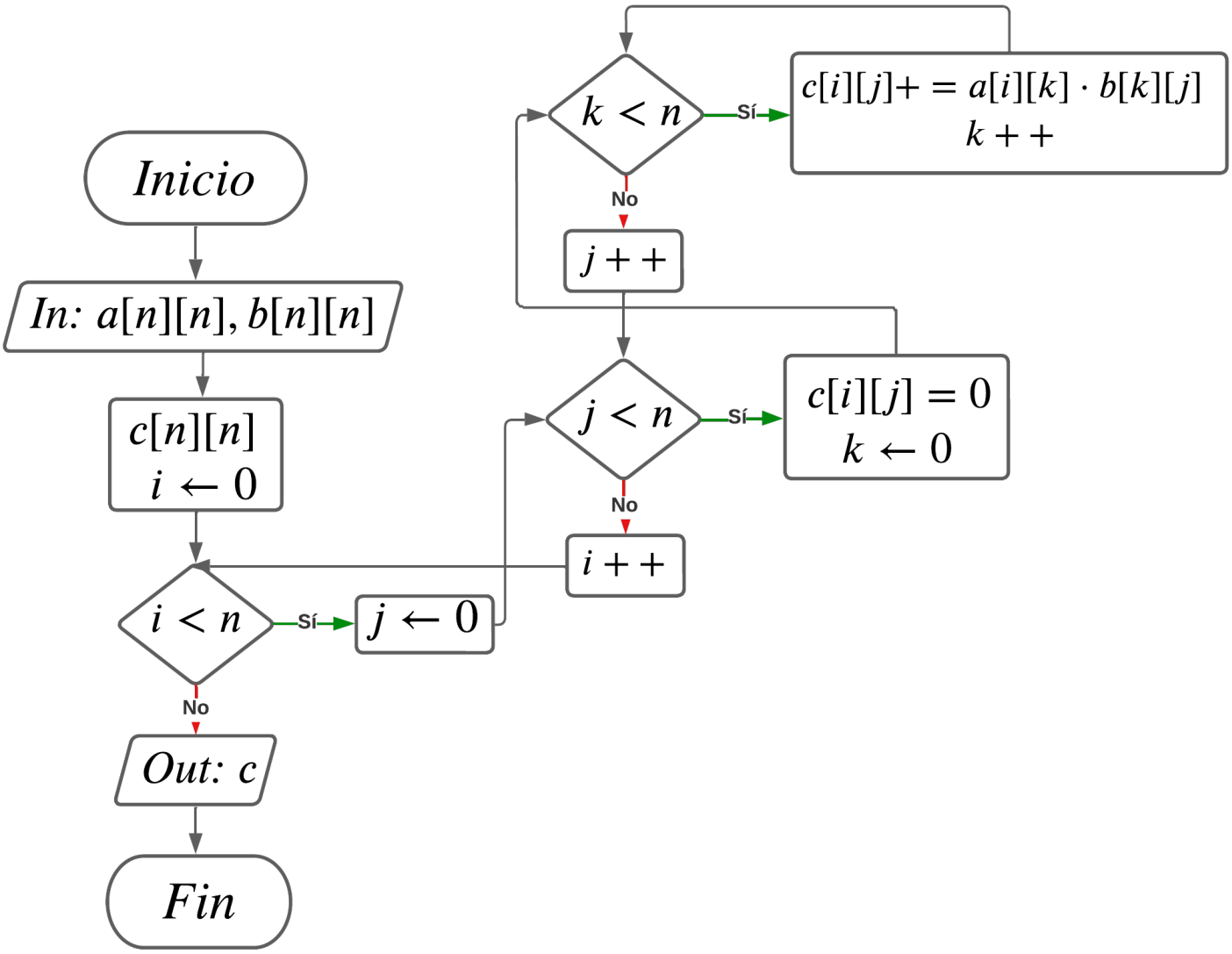

Figura 3: Diagrama de flujo de algoritmo 3

1.4. Algoritmo 4

Escriba un algoritmo que recibe 2 cadenas de caracteres (cad[n] y subCad[m], | n>=m), y analiza si la subcadena de caracteres se encuentra en la cadena de caracteres. La función devuelve verdadero si la subcadena se encuentra en la cadena.
Entrada: cad[n], subCad[m] **| $m \leq n$**
Salida: true/false

Algorithm 4 Subcadena en cadena

```
1: last ← n − m
2: for i ← 0, i ≤ last , i + + do
3:    j ← 0
4:    while cad[i + j] == subCad[j] do
5:       if j == m-1 then
6:          return true
7:       end if
8:       j + +
9:    end while
10: end for
11: return false
```

Complejidad

$T_1(n, m)$: líneas 4-9

vemos que cuando $\boldsymbol{j \equiv m - 1}$ el algoritmo retorna verdadero,
entonces el límite superior será cuando $\boldsymbol{j \equiv m - 2}$
entonces:

$$T_1(n,m) = 4 + \sum_{j=0}^{m-2} \Big(4 + 2 + 1\Big)$$

$$T_1(n,m) = 4 + \sum_{j=0}^{m-2} (7)$$

$$T_1(n,m) = 4 + (m - 2 - 0 + 1) \cdot (7)$$

$$T_1(n,m) = 4 + (m - 1) \cdot (7)$$

$$T_1(n,m) = 7m - 3$$

$$\boldsymbol{T(n,m) = 3 + 2 + \sum_{i=0}^{n-m} \Big(2 + 1 + T_1(n,m)\Big)}$$

$$T(n,m) = 5 + \sum_{i=0}^{n-m} \Big(T_1(n,m) + 3\Big)$$

$$T(n,m) = 5 + \sum_{i=0}^{n-m} \Big(7m - 3 + 3\Big)$$

$$T(n,m) = 5 + \sum_{i=0}^{n-m} (7m)$$

$$T(n,m) = 5 + (n - m - 0 + 1) \cdot (7m)$$

$$T(n,m) = 5 + (n - m + 1) \cdot (7m)$$

$$\boldsymbol{T(n,m) = 7mn - 7m^2 + 7m + 5}$$

$$\boldsymbol{T(n,m) \in O(mn)}$$

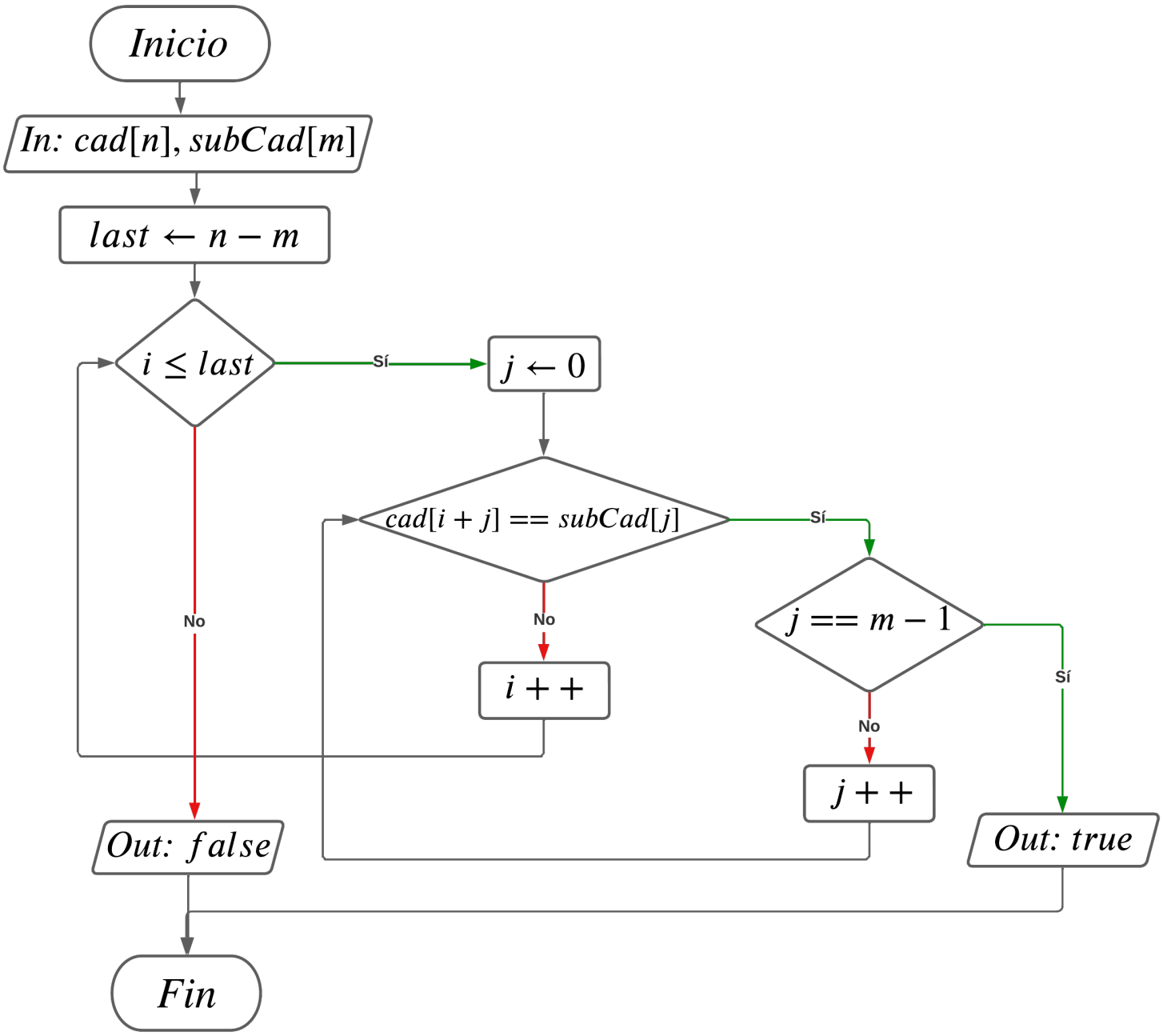

Figura 4: Diagrama de flujo de algoritmo 4

2. Solución (22 guante)

En un cajón hay 22 guantes: 5 pares de guantes son rojos, 4 pares son amarillos y 2 pares son verdes. Puedes tomar los guantes de uno en uno en la oscuridad y solo puedes saber el color una vez que estén fuera de la caja.

¿Cuál es la menor cantidad de guantes que debes elegir para tener al menos un par del mismo color en el mejor caso?
Dos guantes, ya que con la mejor de la suertes los primeros dos completarán el par del mismo color.
Y ¿En el peor caso?
Se tendrían que sacar cuatro guantes, con la peor de las suertes los primeros tres serían de colores diferentes, y el cuarto sería de un color el cual ya hemos sacado antes.