Historial de Usuario- Semana 2

H01 – Ingresar nombre y notas de estudiantes

- **Descripción:** El sistema solicita al usuario nombre y nota de cada estudiante que desee ingresar en una sola cadena de texto.
- **Caso exitoso:** Se ingresa nombres y notas en orden [Nombre Nota], sin puntos ni comas para separarlos, separados por espacios.
- **Caso no exitoso:** El usuario ingresa los datos en un orden diferente o usa comas/puntos para separar los datos.
- **Entrada:** Nombres y notas.
- **Salida:** No hay, se almacenan internamente los datos en una lista.

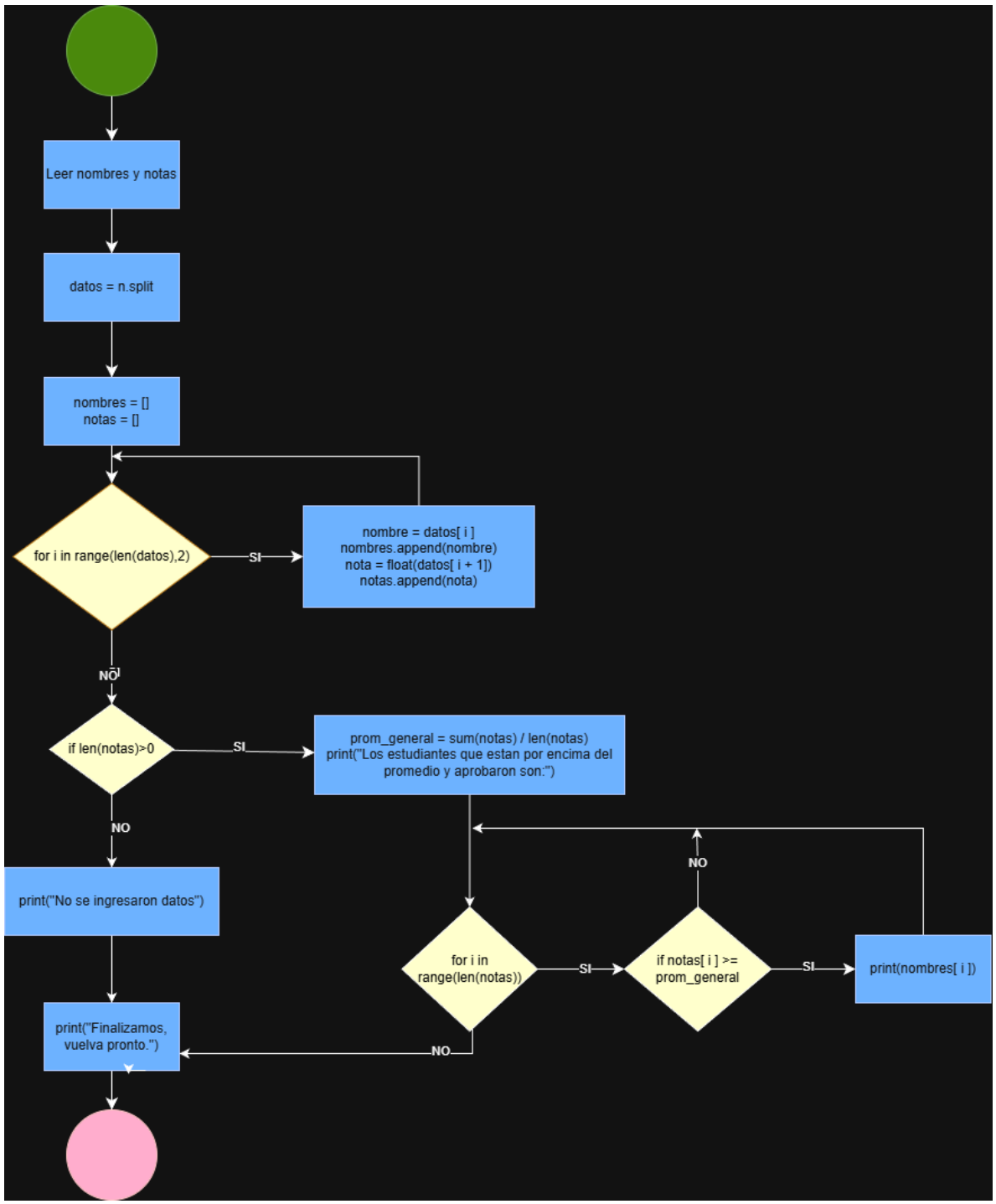
H02 – Evaluar estudiantes por encima del promedio.

- **Descripción**: El sistema evalúa que estudiantes cumplen con la condición de tener una nota igual o superior al promedio general de los datos ingresados.
- **Caso exitoso:** Se muestra el nombre de los estudiantes que igualaron o superaron el promedio acumulado.
- **Caso no exitoso**: El usuario no ingreso datos.
- **Entrada**: Lista de nombres y lista de notas procesadas.
- **Salida:** Nombres de los estudiantes que cumplen con estar por encima del promedio y aprobaron.

Requerimientos

- **R01:** Debe solicitar nombres y notas de los estudiantes en una misma entrada.
- **R02:** Debe separar y almacenar los nombres y las notas de los estudiantes en listas independientes.
- **R03:** Debe calcular el promedio aritmético de todas las notas ingresadas.
- **R04:** Debe comparar la nota de cada estudiante individualmente con el promedio general que se obtuvo.
- **R05:** Debe mostrar los nombres de los estudiantes que aprobaron por cumplir con ser mayor o igual al promedio general que se obtuvo.
- **R06:** Debe validar si se ingresaron datos para evitar errores de cálculo, no dividir por cero.

Diagrama de Flujo



Complejidad

Línea de Código	Costo	Tiempo
`n = input("Ingrese los datos de los estudiantes: ")`	C_1	1
`datos = n.split()`	C_2	1
`nombres = []`	C_3	1
`notas = []`	C_4	1
`for i in range(len(datos),2):`	C_5	$\sum_{i=0}^{n} 1 = n+1$
`    nombre = datos[i]`	C_6	$\sum_{i=1}^{n} 1 = n$
`    nombres.append(nombre)`	C_7	$\sum_{i=1}^{n} 1 = n$
`    nota = float(datos[i + 1])`	C_8	$\sum_{i=1}^{n} 1 = n$
`    notas.append(nota)`	C_9	$\sum_{i=1}^{n} 1 = n$
`if len(notas)>0:`	C_{10}	$1\ (len(notas) > 0)$
`    prom_general = sum(notas)/len(notas)`	C_{11}	$1\ (len(notas) > 0)$
`    print("Los estudiantes que estan por encima del promedio y aprobaron son:")`	C_{12}	$1\ (len(notas) > 0)$
`    for i in range(len(notas)):`	C_{13}	$\sum_{i=0}^{n} 1 = n+1$
`        if notas[i] >= prom_general:`	C_{14}	$\sum_{i=1}^{n} 1 = n$
`            print(nombres[i])`	C_{15}	$\sum_{i=1}^{n} 1 = n$
`else:`	C_{16}	$1\ (n < 0)$
`    print("No se ingresaron datos.")`	C_{17}	$1\ (n < 0)$
`print("Finalizamos, vuelva pronto.")`	C_{18}	1

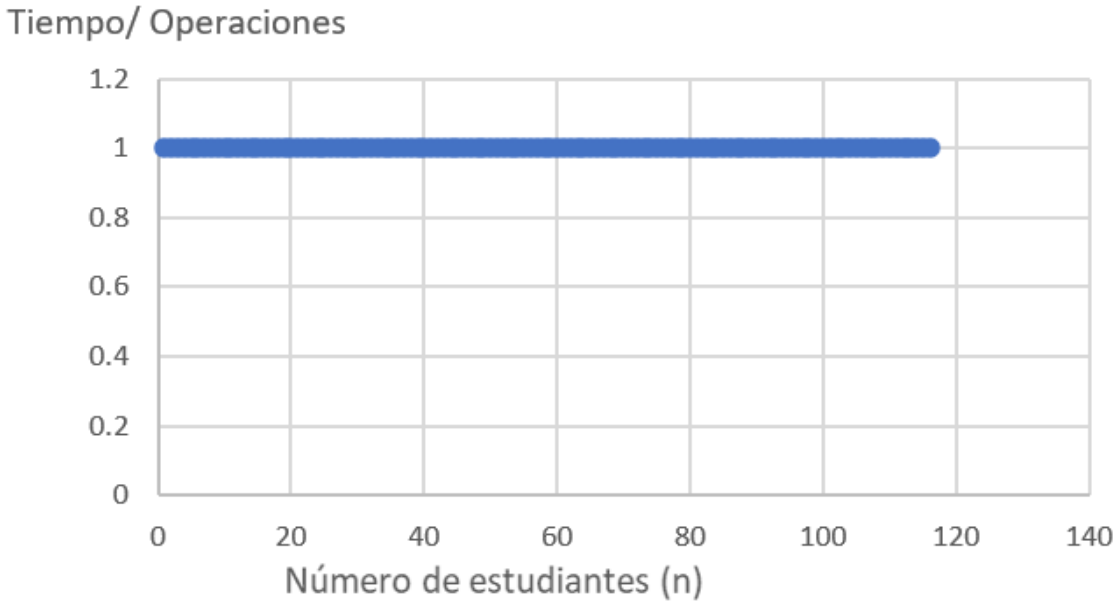
$$T(n) = C_1 + C_2 + C_3 + C_4 + C_5(n+1) + (C_6 + C_7 + C_8 + C_9)n + C_{10} + C_{11} + C_{12} + C_{13}(n+1) + (C_{14} + C_{15})n + C_{16} + C_{17} + C_{18}$$

$$T(n) = (C_5 + C_6 + C_7 + C_8 + C_9 + C_{13} + C_{14} + C_{15})n + C_1 + C_2 + C_3 + C_4 C_{10} + C_{11} + C_{12} + C_{16} + C_{17} + C_{18}$$

$$T(n) = an + b$$

Análisis de Casos y Graficas

Mejor caso: Constante



Peor caso: $O(n) = n$

Análisis General

El algoritmo es eficiente para procesar listas de estudiantes, ya que solo requiere recorrer del numero total de datos (nombres + notas) a la mitad, haciendo que su complejidad sea lineal. Podemos observar que, al hallar su complejidad en caso de duplicar la cantidad de alumnos, el tiempo de procesamiento también lo hará, siendo ideal para analizar este caso escolar. El uso de n.split y la división de los datos en dos listas, no afecta en nada, refiriéndonos al hecho de que almacene mas datos; son los mismos solo que al dividirlos volvemos el proceso mas eficiente.