15-2-2026

Semana 3

Diseño del problema de los estudiantes con divide y conquistar

Laila Zareth Romano Guerrero
DDYA-7

Descripción del problema

El sistema recibe una cadena de texto con nombre y notas de estudiantes separados por un espacio, los procesa de forma secuencial para calcular el promedio y luego identifica quienes superan dicho promedio. Si tenemos pocos datos ingresados el tiempo de respuesta es casi inmediato, pero a medida que mas estudiantes son ingresados, el procesamiento que antes era lineal puede llegar a ser mejor/optimizado mediante el uso de la herramienta Dividir y Conquistar. El objetivo es descomponer la lista de estudiantes en subproblemas mas pequeños, resolver la condición de que aprueben de forma independiente y llegar a combinar los resultados para obtener la lista final de estudiantes que superan el promedio general.

Requerimientos (Historias de usuario)

H01- Procesamiento de datos

- **Descripción:** El sistema solicita al usuario los nombres y las notas de los estudiantes en una sola cadena de texto para ser luego organizada.
- **Caos exitoso:** Se ingresan los datos en el orden “Nombre Nota”, separados por espacios y el sistema los separa en listas independientes de nombres y notas.
- **Caso no exitoso:** El usuario ingresa una cadena vacía, datos que no permiten una separación par o ingresa sin orden los datos de los estudiantes.
- **Entrada**: Cadena de texto con nombres y notas.
- **Salida:** Ninguna, el sistema almacena internamente en listas.

H02-Filtrado recursivo

- **Descripción:** El sistema aplica una estrategia recursiva de dividir y conquistar para evaluar que estudiantes tienen una nota mayor o igual al promedio general calculado.
- **Caso exitoso:** El sistema descompone la lista (divide) hasta el nivel individual, nuestro caso base, valida la condición y combina (conquistar) los resultados para mostrar los nombres de los estudiantes que aprobaron.
- **Caso no exitoso:** No se ingresaron datos suficientes para calcular un promedio o realizar la división recursiva.
- **Entrada:** Lista de nombres, lista de notas, el promedio general y los limites (low, high).
- **Salida:** Lista de nombres de los estudiantes que lograron cumplir con la condición, generada por la unión de los sub- resultados (divisiones que hicimos anteriormente).

Requerimientos

- **R01:** Debe recibir la entrada y separar los datos en listas de nombres y notas.
- **R02:** Debe calcular el promedio de nota total del grupo antes de iniciar el proceso de filtrado.
- **R03**: Debe dividir el problema encontrando un punto medio de forma recursiva,
- **R04**: Debe validar en el caso base si la nota del estudiante cumple con la condición de ser mayor o igual al promedio.
- **R05:** Debe unir las listas de resultados de las divisiones (izquierda y derecha) para reconstruir la solución.

Diagrama de flujo

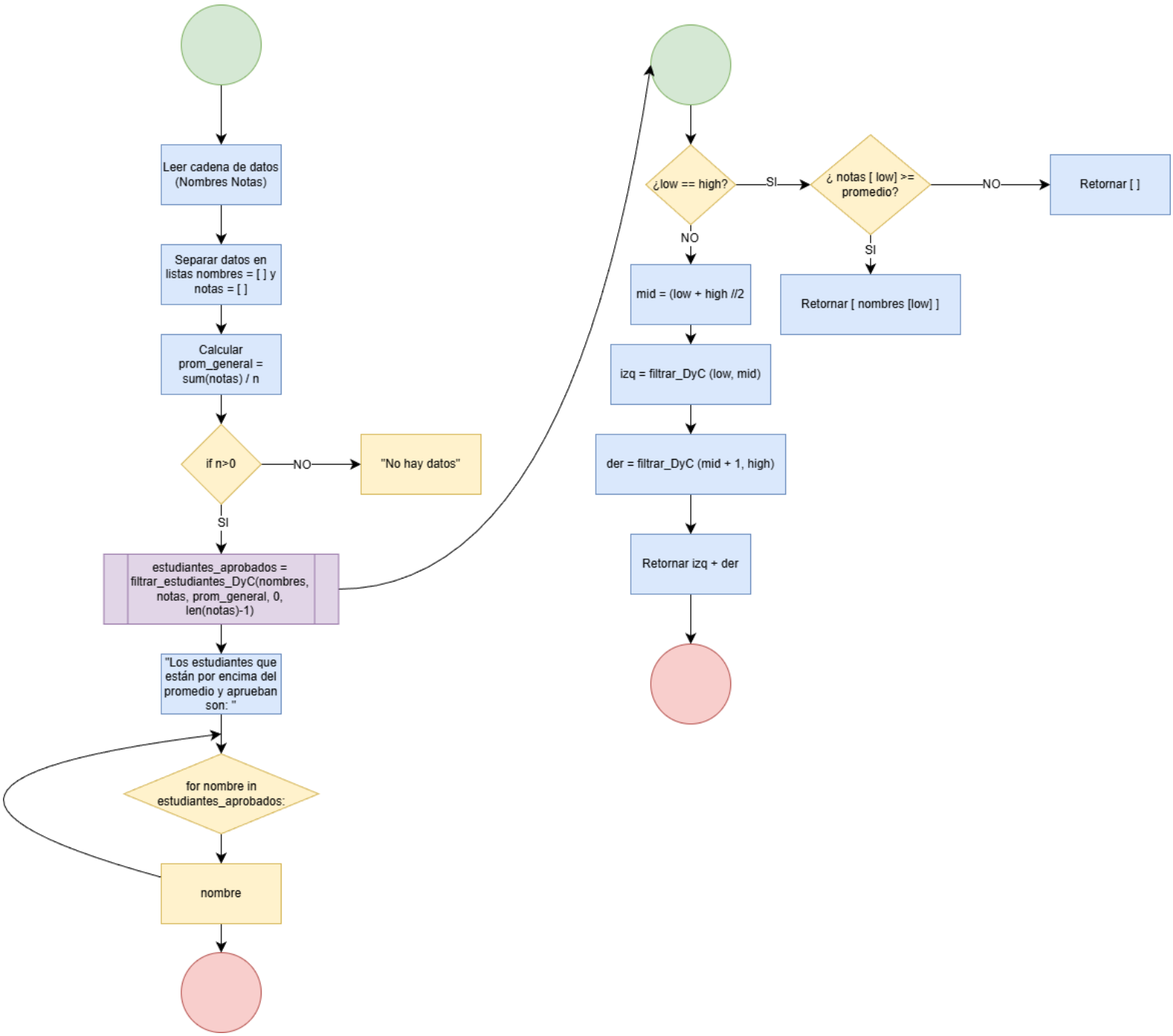


Diagrama de secuencia

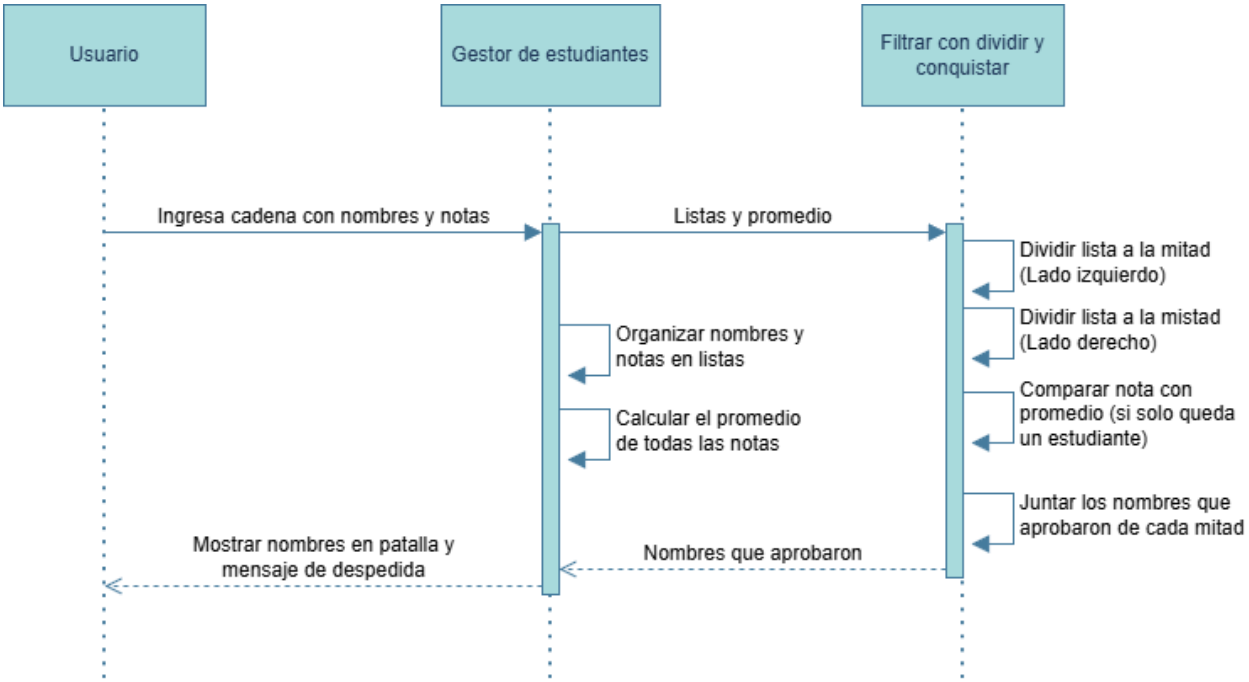


Diagrama de caso de uso

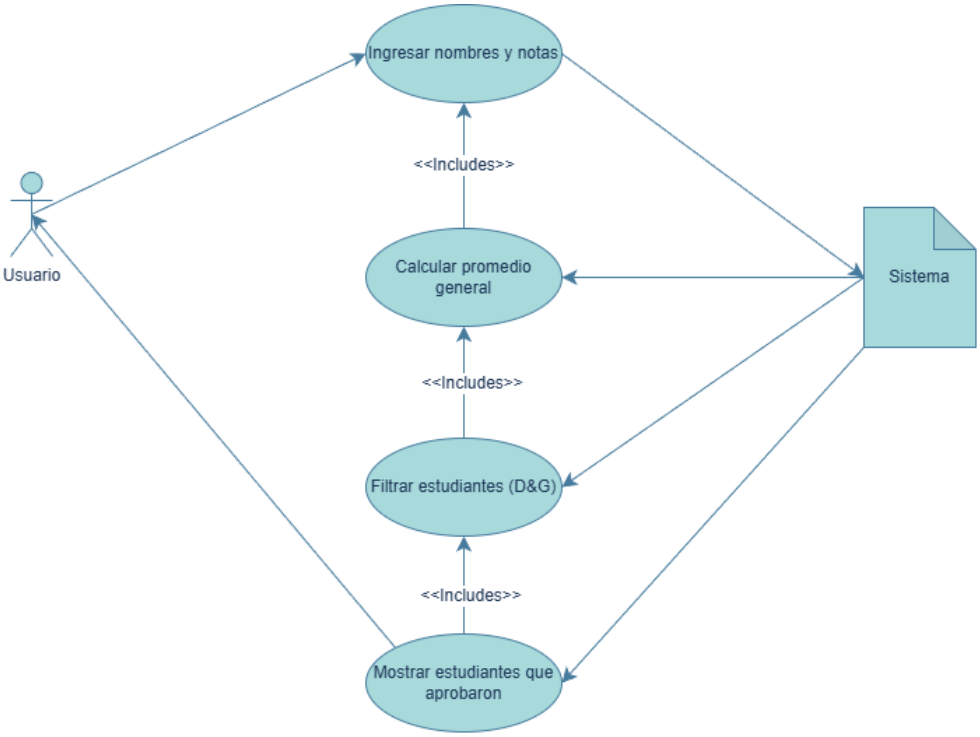


Tabla de complejidad

Línea	Cost	Times
`    if low == high:`	C_4	$2n - 1$
`        if notas[low] >= promedio:`	C_5	$n\ (notas[low] >= promedio)$
`            return [nombres[low]]`	C_6	$1\ (notas[low] >= promedio)$
`        else:`	C_7	
`            return []`	C_8	1
`    mid = (low + high) // 2`	C_{10}	$n - 1$
`    izq = filtrar_estudiantes_DyC(nombres, notas, promedio, low, mid)`	$C_{12} + T(\frac{n}{2})$	1
`    der = filtrar_estudiantes_DyC(nombres, notas, promedio, mid + 1, high)`	$C_{13} + T(\frac{n}{2})$	1
`    return izq + der`	C_{15}	1
`    n = input("Ingrese los datos de los estudiantes: ")`	C_{18}	1
`    datos = n.split()`	C_{19}	1
`    nombres = []`	C_{20}	1
`    notas = []`	C_{21}	1
`    for i in range(0, len(datos), 2):`	C_{22}	$\frac{n}{2} + 1$
`        nombres.append(datos[i])`	C_{23}	$\frac{n}{2}$
`        notas.append(float(datos[i + 1]))`	C_{24}	$\frac{n}{2}$
`    if len(notas) > 0:`	C_{26}	$1(len(notas) > 0)$
`        prom_general = sum(notas) / len(notas)`	C_{27}	$n(len(notas) > 0)$
`        estudiantes_aprobados = filtrar_estudiantes_DyC(nombres, notas, prom_general, 0, len(notas) - 1)`	$C_{29} + T(n)$	$1(len(notas) > 0)$
`        print("Los estudiantes que están por encima del promedio y aprueban son:")`	C_{30}	1
`        for nombre in estudiantes_aprobados:`	C_{31}	$n + 1$
`            print(nombre)`	C_{32}	n
`    else:`	C_{33}	
`        print("No se ingresaron datos.")`	C_{34}	1
`    print("Finalizamos, vuelva pronto")`	C_{35}	1

Complejidad total

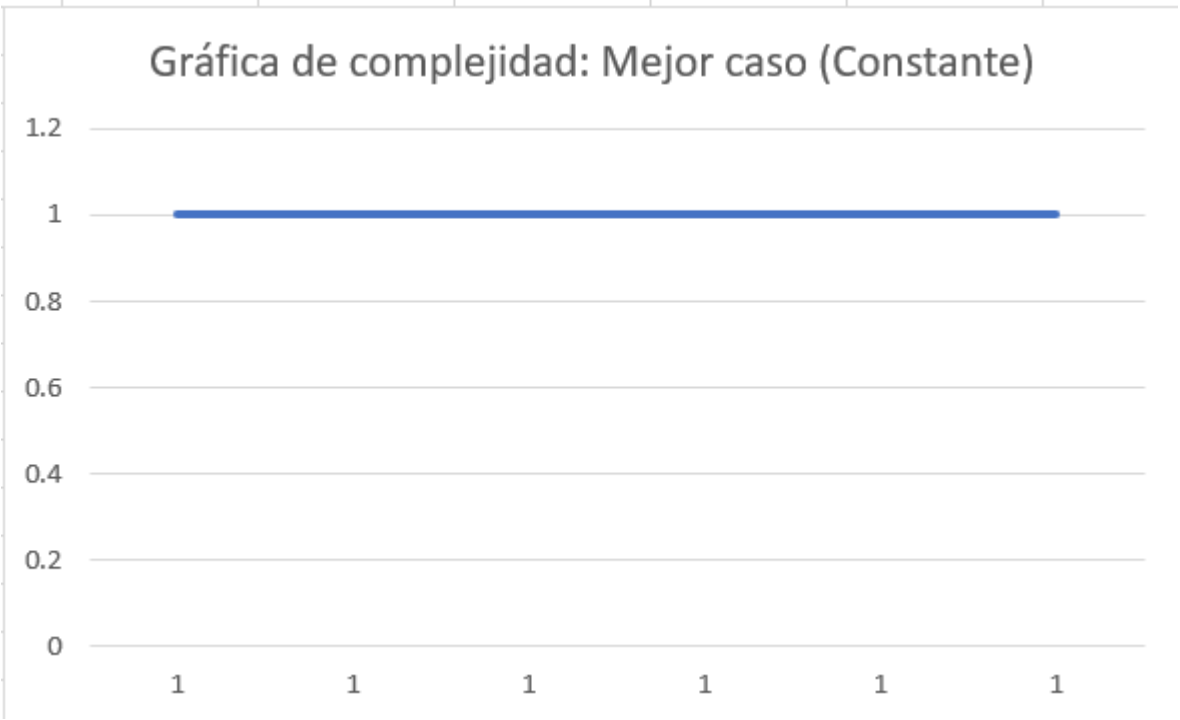
$$\boldsymbol{T(n) = Cn + 2T(n/2) + C}$$

Notación Big Oh

$$\boldsymbol{T(n) = 0(n)}$$

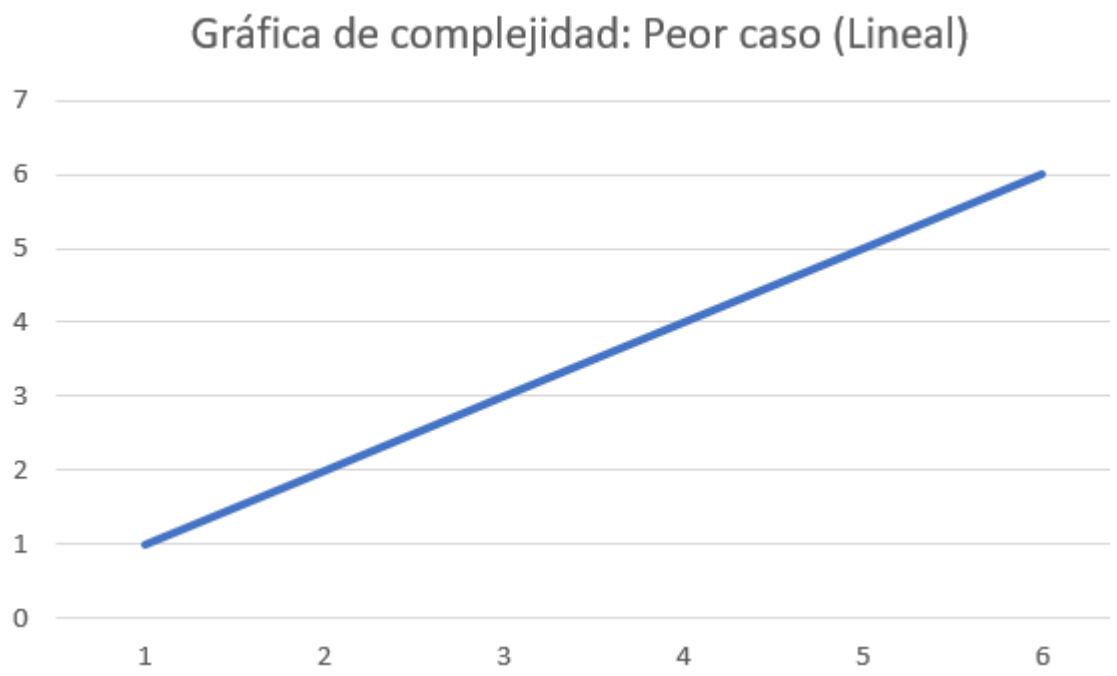
Mejor caso $\boldsymbol{O(1)}$

El sistema no tiene datos que procesar. Puede que el usuario ingresara una cadena vacía o datos no válidos. El sistema llega a validar si el contenido de la lista de notas es mayor a cero, llega a la conclusión que es falso y enseguida le muestra al usuario que no se ingresaron datos.



Peor caso $\boldsymbol{O(n)}$

El sistema procesa una cantidad n grande de estudiantes. El sistema debe realizar el ciclo para separar los datos, luego calcular el promedio de acuerdo a la lista de datos (recorre la lista) y finalmente ejecuta la función recursiva de dividir y conquistar, generando subproblema (n/2). El tiempo de ejecución aumenta de manera lineal a medida que se agregan mas datos de estudiantes.



Análisis

Al rediseñar el sistema bajo la herramienta de dividir y conquistar, se observa que la eficiencia temporal se mantiene en una escala lineal, similar al método iterativo que se aplico anteriormente. La implementación de filtrar a los estudiantes con D&C permite que el problema de filtrado se descomponga en unidades independientes. Esto transforma un proceso que solo era una fila en una estructura que se va descomponiendo como una estructura de árbol, permitiendo que la mitad izquierda y la mitad derecha se procesan al mismo tiempo, reduciendo el tiempo de respuesta real. En conclusión, este cambio es una forma de adaptar nuestro problema para que mas adelante si agregamos muchos datos, sean procesados un poco más rápido que hacerlo con una sola fila de datos.