**D - Thor**

**1. Reducción del Problema**

* **Descripción en palabras:**

el problema pide que se rastree el número de notificaciones sin leer en un smartphone, que tiene n aplicaciones. Se tiene q eventos. Estos eventos pueden ser: una aplicación que genera una nueva notificación, Thor leyendo todas las notificaciones de una aplicación específica, o Thor leyendo las primeras t notificaciones generadas en el teléfono en general. Después de cada evento, debes indicar cuántas notificaciones sin leer quedan. Inicialmente no hay notificaciones

* **Descripción en lenguaje matemático:**

Sea N el número de aplicaciones y Q el número de eventos. Sea U el conjunto de notificaciones sin leer, inicialmente U=∅. Representamos cada notificación generada cronológicamente como nk​, donde k es el índice del evento de tipo 1 que la generó. Sea Ax​ el conjunto de todas las notificaciones generadas por la aplicación x.

Para cada evento i=1,…,Q:

* Si tipo i​=1 (App xi​ genera notificación): Se crea una nueva notificación nk​. Se añade nk​ a U.
* Si tipo i​=2 (Thor lee todas de App xi​): U de U ∖ Axi​​.
* Si tipo i​=3 (Thor lee las primeras ti​ notificaciones generadas): U de U ∖ {nj​∣1≤j≤ti​}.

El objetivo es imprimir ∣U∣ después de cada evento.

**2. Reducción de la Solución**

* **Descripción en palabras:**

Para resolver el problema, se necesita mantener un seguimiento del estado de cada notificación (leída o sin leer) y tener formas eficientes de actualizarlas y contarlas según los tres tipos de eventos. Esto implica gestionar notificaciones generadas por aplicación y también por su orden cronológico de aparición. El objetivo final es informar el total de notificaciones sin leer después de cada acción de Thor.

* **Descripción en lenguaje matemático:**

Descripción en lenguaje matemático:

Sea N el número total de eventos.  
 Mantenemos una estructura de datos que almacena el estado de cada notificación:

* Sea S⊆{1,2,...,t} conjunto de notificaciones sin leer en el tiempo t.
* Cada notificación se asocia con una aplicación ai∈{1,...,n}

Para cada evento:

1. Si se genera una nueva notificación de la app aa:  
      Agregar t a S, con marca de app a.
2. Si se marcan como leídas todas las de la app aa:  
      Eliminar de S todos los elementos cuya app sea a.
3. Si se marcan como leídas todas las notificaciones hasta el evento número x:  
      Eliminar de S todos los i≤xi

**Después de cada evento, se reporta ∣S∣|S|, es decir, el número total de notificaciones sin leer.**

**3. Código Realizado y Análisis**

* Enlace(s) Código: <https://vjudge.net/solution/61083760/lcc24yE4lOxRNbK7zxPg>

**4. Casos de Prueba**

* Casos de prueba:

**Caso 1**

**Entrada**

**3 4**

**1 3**

**1 1**

**1 2**

**2 3**

**Justificación:**

Se crean notificaciones de apps 3, 1, 2 y luego Thor lee todas las notificaciones de la app 3.

Esto prueba que se descuenta correctamente solo lo de la app 3.

**Salida**

1

2

3

2

**Caso 2**

**Entrada**

**4 6**

**1 2**

**1 4**

**1 2**

**3 3**

**1 3**

**1 3**

Justificación:

Se generan 3 notificaciones, luego se leen las primeras 3 globalmente. Después se generan 2 nuevas.

Este caso prueba que la lectura global lee en orden y puede anular notificaciones de varias apps.

Salida

1

2

3

0

1

2

**Caso 3**

2 6

1 1

1 1

2 1

2 1

1 1

2 1

**Justificación:**

App 1 genera 2 notificaciones, se leen, y luego se vuelve a leer sin nuevas. La cuenta no debe cambiar por la segunda lectura.

**Caso 4**

**Salida**

1

2

0

0

1

0

**Caso 5 el caso más extremo que se ocurrió**

**Entrada**

300000 300000

1 1

1 2

1 3

...

1 300000 300,000 notificaciones distintas, una por cada app (tipo 1)

3 300000 leer TODAS en un solo evento (tipo 3)

**Justificación:**

El primer bloque genera 300,000 notificaciones únicas, cada una de una app distinta (esto estresa la memoria y la lista por\_app).

El evento tipo 3 con t = 300000 obliga a revisar todas las notificaciones almacenadas. Esto estresa el recorrido lineal de notificaciones y la variable leidas\_global.

**Este caso fuerza:**

* La longitud de notificaciones al máximo.
* La longitud de cada sublista en por\_app a 1 (caso peor para recorrer muchas apps).
* El recorrido total de notificaciones en el tipo 3 a su límite.

**5. Iteración en Caso de Solución Incorrecta (o explicación Solución Correcta)**

**Código incorrecto:** [**https://vjudge.net/solution/61013517/5sFGvfeL76eMKo3WPRMn**](https://vjudge.net/solution/61013517/5sFGvfeL76eMKo3WPRMn)

* **Descripción del error:** tuves varios errores pero el error principal es la eficiencia ósea en algunos test me salia Time limit exceeded on test 7 a veces en el test 7 y otras veces en otros test y todo eso debido a un uso ineficiente de estructuras y recorridos. En particular, se hace un recorrido completo de la lista historial en cada iteración para recalcular el estado de las notificaciones. Esto implica una complejidad O(q²) en el peor caso, ya que se recorren hasta 300,000 elementos hasta 300,000 veces, lo cual excede fácilmente el límite de tiempo.
* **Proceso de depuración**: Primero mire el código a simple vista y me daba cuenta que el código su complejidad de tiempo era cuadrática, igual me pregunté si todo el tiempo era cuadrática o solo es algun caso aislado en donde lo fuera, así que muchos casos de pruebas (como 20 o 30) y lo que hice fue debugear linea por linea y esto fue lo que encontre.

Detecté que en cada evento, el código calculaba desde cero cuántas notificaciones estaban sin leer. específicamente esta línea:

**Uso de .count() en listas (O(n))** querecorre todo el historial en cada evento (O(q) veces que seria O(q²) total), reconstrucción completa del estado de lectura tras cada eventos

Ahí cree un caso de prueba artificial con 100,000 eventos tipo 1 seguidos de un evento tipo 3. Esto tardó más de 2 segundos, confirmando que la complejidad no era lineal.

* **Solución corregida:**

Es mismo que en el punto 3: <https://vjudge.net/solution/61083760/lcc24yE4lOxRNbK7zxPg>

esto fue lo que cambie:

* primero cambie de enfoque de recalcular todo por uno incremental
* mantuve algunas cosas como un contador global de no leídas (sin\_leer)
* use listas para almacenar el índice de notificaciones por app (por\_app)
* use un índice (leidas\_global) que evita volver a recorrer notificaciones ya leídas por tipo 3

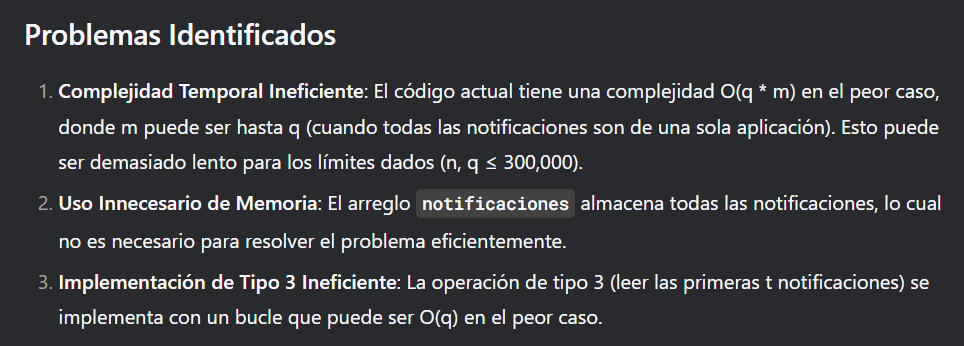
ya con todas esa soluciones paso a complejidad O(q \* m) lo cual es casi igual que cuadrática pero para este caso pasé todos los test quizás haya forma de hacerlo más eficiente pero en mi caso no la encontre

**6. Preguntas de Aprendizaje**

* Temas aplicados: Fuera de for, arrays, variables ninguno
* Dificultad de la implementación: Medio, fue el primer ejercicio que ya se puso una dificultad y lo más difícil fue como fuera
* Recursos utilizados: Ninguno

**7. Feedback LLM**

* Envío código LLM: <https://vjudge.net/solution/61083837/uk3e0hL6XL4AoMCRjzPO>
* **Comparación de su códigos**: el código LLM curiosamente no pasó el todos los test dando error de memoria en el test 9(un error que no había visto antes) así que solo por eso el mejor seria el mio(su versión final) si logro pasar todos los test
* Feedback de LLM: Pida al LLM que evalúe y corrija su código:

Me dijo que era ineficiente y que usa memoria innecesaria 

Y la verdad si estoy de acuerdo en cuanto que es aún ineficiente, aunque no pense que diria algo con respecto a la memoria aunque la verdad nunca me enfoque que fuera más eficiente con respecto a eso