El misterio de la cadena secreta (misterio)

Descripción

En el reino de Numeralia, los guardianes custodian una **cadena secreta** S de longitud L, compuesta únicamente de letras minúsculas (`a`–`z`). La cadena está almacenada en una cripta que solo permite consultar un carácter a la vez: los sabios pueden pedir el carácter que está en la posición i (1-indexado), y la cripta revelará esa letra.

El Consejo de Sabios te encomienda la misión de **descubrir toda la cadena secreta** haciendo un número eficiente de consultas. Cuando creas que conoces la cadena completa, debes enviar tu propuesta al Consejo para que sea evaluada.

Implementación

El evaluador iniciará el proceso indicándote la longitud de la cadena y esperará a que le indiques el primer índice que deseas consultar.

Después, repetidamente:

- El evaluador te proporcionará el carácter en el índice que pediste.
- Tu programa debe decidir el siguiente índice a consultar.
- Este ciclo se repite hasta que devuelvas -1, indicando que terminaste de consultar.

Finalmente, el evaluador te pedirá la cadena completa, la evaluará y te dirá si acertaste.

Debes implementar las siguientes funciones en C++:

```
int iniciar_descifrado(int longitud, int subtask);

int procesar_caracter(char caracter);

string obtener_cadena_descifrada();
```

O las siguientes funciones en Python:

```
def iniciar_descifrado(longitud: int, subtarea: int) -> int:

def procesar_caracter(caracter: str) -> int:

def obtener_cadena_descifrada() -> str:
```

La función `iniciar_descifrado` se encargará de iniciar el proceso de descifrado de la cadena secreta. Esta función será llamada exactamente una vez al inicio de la interacción, recibiendo como 2 parámetros la longitud L de la cadena oculta y la subtarea ($1 \leq \boldsymbol{subtarea} \leq 6$). Debes utilizar esta llamada para inicializar tus estructuras de datos, preparar tu estrategia de consulta y devolver el índice de la primera posición que quieres consultar. Si no deseas realizar ninguna consulta, puedes devolver `-1` directamente.

La función `procesar_siguiente_caracter` será llamada repetidamente por el evaluador cada vez que consultes un índice. En cada llamada, recibirás el carácter correspondiente a la posición que pediste anteriormente. Debes procesar ese carácter, actualizar tu estado interno y devolver el siguiente índice que quieres consultar. Si consideras que has reunido suficiente información, puedes devolver `-1` para finalizar el ciclo de consultas. Esta función puede ser llamada hasta un máximo de $2 \times L$ veces, dependiendo de tu estrategia.

La función `obtener_cadena_descifrada` será llamada exactamente una vez al final del proceso de consultas. En esta función debes devolver la cadena completa que propones como solución, construida en base a la información que reuniste durante las llamadas anteriores. Tu respuesta será evaluada para verificar si coincide con la cadena secreta.

Ejemplos

Ejemplo 1

Supongamos que la cadena secreta es "cab" (longitud 3).

1. Se llama a `iniciar_descifrado(3)` → tu programa devuelve 1.
2. Evaluador: te da `c`.
3. Se llama a `procesar_caracter('c')` → devuelves 2.
4. Evaluador: te da `a`.
5. Se llama a `procesar_caracter('a')` → devuelves 3.
6. Evaluador: te da `b`.
7. Se llama a `procesar_caracter('b')` ' → devuelves -1.
8. Se llama a `obtener_cadena_descifrada()` → devuelves "cab".
9. Evaluador verifica la propuesta.

Restricciones

- $1 \leq L \leq 10^5$, donde L es la longitud de la cadena secreta.
- La cadena está compuesta únicamente de letras minúsculas (`a`–`z`).
- El evaluador garantiza que la cadena cumple una de las condiciones descritas en las subtareas.

Subtareas

Subtarea 1 (22 puntos)

Las cadenas serán de la forma $S = cad1\,cad1\,cad1$, donde $cad1$ es una cadena con caracteres distintos. Es decir, la cadena secreta consiste en repetir tres veces o más una misma secuencia única de letras distintas.

- Ejemplo: `abcabcabc`, `zxywzxywzxywzxywzxyw`.

Subtarea 2 (22 puntos)

La cadena tendrá caracteres consecutivos en orden creciente o decreciente.

- Ejemplos:
  - Creciente: `abcdefg`, `mnopqr`.
  - Decreciente: `zyxwv`, `kjih`.

Subtarea 3 (7 puntos)

La cadena estará formada únicamente por un solo tipo de letra repetida.

- Ejemplos: `aaaaa`, `zzzzzz`.

Subtarea 4 (22 puntos)

La cadena tendrá todos los caracteres ordenados (de menor a mayor en valor alfabético), y cada tipo de carácter aparecerá la misma cantidad de veces.

- Ejemplo: `aabbcc`, `eeefff` (cada letra aparece dos y tres veces respectivamente, y las letras están en orden).

Subtarea 5 (22 puntos)

La cadena será un palíndromo, es decir, se lee igual de izquierda a derecha que de derecha a izquierda.

- Ejemplos: `abba`, `racecar`, `abcba`.

Subtarea 6 (5 puntos)

Las cadenas no serán de ninguno de los tipos descritos en las subtareas anteriores; es decir, se trata de cadenas completamente generales sin restricciones

especiales.

- Ejemplo: `abacabad"`, `qwertyui`.

Evaluación

Cada subtarea será evaluada con X casos de prueba, donde X es igual a la cantidad de puntos asignados a la subtarea (por ejemplo, la Subtarea 1 con 22 puntos tendrá 22 casos, la Subtarea 6 con 5 puntos tendrá 5 casos, etc.). En otras palabras, habrá un caso de prueba por cada punto de la subtarea.

Para cada caso de prueba:

- Si la cadena propuesta coincide exactamente con la cadena secreta **y** el número de llamadas realizadas a la función `procesar_caracter` es el mínimo posible para el tipo de cadena evaluado, se otorgará 1 punto completo para ese caso.
- Si la cadena propuesta coincide exactamente con la cadena secreta pero se realizaron más llamadas que el número mínimo óptimo, se otorgarán 0.1 puntos para ese caso.
- Si la cadena propuesta es incorrecta (no coincide con la cadena secreta), se otorgarán 0 puntos para ese caso.

El evaluador garantiza que el número máximo de llamadas a `procesar_caracter` en cada caso de prueba es máximo $2 \times L$. Si este límite es alcanzado y aún no se ha devuelto `-1`, el ciclo de consultas se terminará automáticamente y el evaluador llamará a la función `obtener_cadena_descifrada` para recibir la propuesta final del participante.

Al finalizar, los puntos de todos los casos de prueba se sumarán para determinar tu puntuación total.

Ejemplo:

- Subtarea 1 (22 casos): si aciertas 18 cadenas con el número óptimo de consultas y 4 cadenas con más consultas de las necesarias, recibirías $18 \times 1 + 4 \times 0.1 = 18.4$ puntos sobre 22.