ADN Mutante

Grace es una bióloga trabajando en una firma biométrica en Singapur. Como parte de su trabajo, ella analiza secuencias de ADN de varios organismos. Una secuencia de ADN está definida como una cadena que consiste de los caracteres "A", "T" and "C". Note que en esta tarea, las secuencias de ADN **no contienen el caracter "G".**

Definimos una mutación como una operación en una secuencia de ADN donde dos elementos de la secuencia son intercambiados. Por ejemplo, una mutación simple puede transformar "A**C**T**A**" en "A**A**T**C**" intercambiando los caracteres resaltados "A" y "C".

La distancia de la mutación entre dos secuencias es el mínimo número de mutaciones requeridas para transformar una secuencia en otra, o -1 si no es posible transformar una secuencia en otra usando mutaciones.

Grace está analizando dos secuencias de ADN a y b, ambas constan de n elementos con índices del 0 al $n-1$. Tu tarea es ayudar a Grace a responder q preguntas de la forma: cuál es la distancia de mutación entre la subcadena $a[x..y]$ y la subcadena $b[x..y]$? Aquí, una subcadena $s[x..y]$ de una secuencia de ADN s está definida como una secuencia de caracteres consecutivos de s, cuyos índices son de x a y inclusive. En otras palabras, $s[x..y]$ es una secuencia $s[x]s[x+1]\ldots s[y]$.

Detalles de Implementación

Debes implementar los siguientes procedimientos:

```
void init(string a, string b)
```

- a, b: cadenas de tamaño n, describiendo las dos secuencias de ADN a ser analizadas.
- Este procedimiento es llamado exactamente una vez, antes de cualquier llamada a `get_distance`.

```
int get_distance(int x, int y)
```

- x, y: los índices de inicio y fin de las subcadenas a ser analizadas.
- El procedimiento debe retornar la distancia de la mutación entre las subcadenas $a[x..y]$ y $b[x..y]$.
- Este procedimiento es llamado exactamente q veces.

Ejemplo

Considere la siguiente llamada:

```
init("ATACAT", "ACTATA")
```

Digamos que el evaluador llama a `get_distance(1, 3)`. Esta llamada debe retornar la distancia de mutación entre $a[1..3]$ y $b[1..3]$, esto es, las secuencias "TAC" y "CTA". "TAC" puede ser transformado en "CTA" con 2 mutaciones: **TA**C $\rightarrow$ **C**A**T**, seguido de C**AT** $\rightarrow$ C**TA**, y la transformación es imposible con menos de 2 mutaciones.

Por lo tanto, esta llamada debe retornar 2.

Digamos que el evaluador llama a `get_distance(4, 5)`. Esta llamada debe retornar la distancia de mutación entre las secuencias "AT" y "TA". "AT" puede ser transformado en "TA" a travez de una simple mutación, y claramente al menos una mutación es requerida.

Por lo tanto, esta llamada debe retornar 1.

Finalmente, digamos que el evaluador llama a `get_distance(3, 5)`. Ya que **no hay forma** para que la secuencia "CAT" sea transformada en "ATA" por ninguna secuenca de mutaciones, esta llamda debe retornar -1.

Restricciones

- $1 \leq n, q \leq 100\ 000$
- $0 \leq x \leq y \leq n - 1$
- Cada caracter de a y b es "A", "T", o "C".

Subtareas

1. (21 puntos) $y - x \leq 2$
2. (22 puntos) $q \leq 500,\ y - x \leq 1000$, cada caracter de a y b es "A" o "T".
3. (13 puntos) cada caracter de a y b es "A" o "T".
4. (28 puntos) $q \leq 500,\ y - x \leq 1000$
5. (16 puntos) Sin restricciones adicionales.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada en el siguiente formato:

- línea 1: n q
- línea 2: a
- línea 3: b
- línea $4 + i$ ($0 \leq i \leq q - 1$): x y para la i-ésima llamada a `get_distance`.

El evaluador de ejemplo imprime tu respuesta en el siguiente formato:

- línea $1 + i$ ($0 \leq i \leq q - 1$) : el valor de retorno de la i-ésima llamada a `get_distance`.