Mutando ADN

Graciela es una bióloga que trabaja para una empresa de bioinformática en Singapur. Parte de su trabajo es analizar las secuencias de ADN de varios organismos. Se define una secuencia de ADN como una cadena de caracteres, que consiste de caracteres "A", "T", y "C". Notar que en este problema las secuencias de ADN **no contienen al caracter "G".**

Definimos una mutación como una operación sobre una secuencia de ADN, en la cual se intercambian dos elementos de la secuencia. Por ejemplo, mediante una mutación se puede transformar "A**C**T**A**" en "A**A**T**C**" intercambiando los caracteres remarcados "A" y "C".

La distancia de mutación entre dos secuencias es la cantidad mínima de mutaciones que se necesitan para transformar una secuencia en la otra, o bien -1 si no es posible transformar una secuencia en la otra utilizando mutaciones.

Graciela se encuentra analizando dos secuencias de ADN a y b. Ambas consisten de n elementos con índices desde 0 hasta $n-1$. Tu tarea es ayudar a Graciela 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 se define como una secuencia de caracteres consecutivos de s, cuyos índices son desde x hasta y inclusive. En otras palabras, $s[x..y]$ es la secuencia $s[x]s[x+1]\ ...\ s[y]$.

Detalles de implementación

Debes implementar las siguientes funciones:

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

- a, b: cadenas de longitud n, que describen las dos secuencias de ADN que se deben analizar.
- Esta función se llama exactamente una vez, antes de cualquier llamada a `get_distance`.

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

- x, y: índices de comienzo y fin de las subcadenas que se deben analizar.
- La función debe retornar la distancia de mutación entre las subcadenas $a[x..y]$ y $b[x..y]$.
- Esta función se llama exactamente q veces.

Ejemplo

Considera la siguiente llamada:

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

Digamos que el evaluador realiza la llamada `get_distance(1, 3)`. Esta llamada debe retornar la distancia de mutación entre $a[1..3]$ y $b[1..3]$, o sea, las secuencias "TAC" y "CTA". "TAC" se puede transformar en "CTA" por medio de 2 mutaciones: **TA**C $\rightarrow$ **CA**T, seguida de C**AT** $\rightarrow$ C**TA**, y la transformación es imposible de realizar con menos de 2 mutaciones.

Por lo tanto, esta llamada debe retornar 2.

Digamos que el evaluador realiza la llamada `get_distance(4, 5)`. Esta llamada debe retornar la distancia de mutación entre las secuencias "AT" y "TA". "AT" se puede transformar en "TA" con una sola mutación, y claramente se necesita por lo menos una mutación.

Por lo tanto, esta llamada debe retornar 1.

Finalmente, digamos que el evaluador realiza la llamada `get_distance(3, 5)`. Como no hay **ninguna forma** de transformar la secuencia "CAT" en la secuencia "ATA" por medio de una serie de mutaciones, esta llamada 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 points) $y - x \leq 2$
2. (22 points) $q \leq 500,\ y - x \leq 1000$, cada caracter de a y b es o bien "A" o bien "T".
3. (13 points) Cada caracter de a y b es o bien "A" o bien "T".
4. (28 points) $q \leq 500,\ y - x \leq 1000$
5. (16 points) Sin restricciones adicionales.

Evaluador local

El evaluador local lee de la entrada con 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 local escribe tus respuestas con el siguiente formato:

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