ADN mutante

Grace es una bióloga trabajando en una firma de bioinformática en Singapur. Como parte de su trabajo, ella analiza la secuencia de ADN de varios organismos. Una secuencia de ADN está definida como una cadena consistente de caracteres "A", "T", y "C". Nota que en esta tarea las secuencias de ADN **no contienen el carácter "G"**.

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

La distancia de mutación entre dos secuencias es la mínima cantidad de mutaciones requeridas para transformar de una secuencia a otra, o -1 si no es posible transformar de una secuencia a otra usando mutaciones.

Grace está analizando 2 secuencias de ADN, a y b, ambas consistiendo en n elementos con índices desde 0 hasta $n-1$. 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 esta definida como una secuencia de caracteres consecutivos de s, cuyos índices van desde x hasta y inclusive. En otras palabras, $s[x..y]$ es la secuencia $s[x]s[x+1]\ldots s[y]$.

Detalles de implementación

Debes implementar el siguiente procedimiento:

```
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 exactamante una vez, antes de cualquier llamada a `get_distance`.

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

- x, y: índices indicando el inicio y el final de la subcadena a ser analizada.
- El procedimiento debe regresar la distancia de mutación entre las subcadenas $a[x..y]$ y $b[x..y]$.
- Este procedimiento es llamado exactamente q veces.

Ejemplo

Considera la siguiente llamada:

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

Digamos que el evaluador llama `get_distance(1, 3)`. Esta llamada debe regresar la distancia de mutación entre $a[1..3]$ y $b[1..3]$, estas son las secuencias "TAC" y "CTA". "TAC" puede ser transformada a "CTA" via 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.

Así que, esta llamada debe regresar 2.

Digamos que el evaluador llama `get_distance(4, 5)`. Esta llamada debe regresar la distancia de mutación entre "AT" y "TA". "AT" puede ser transformada a "TA" a través de una sola mutación, y claramente al menos una mutación es necesaria.

Así que, esta llamada debe regresar 1.

Finalmente, digamos que el evaluador llama `get_distance(3, 5)`. Como **no hay manera** para que la secuencia "CAT" se transforme en "ATA" via cualquier secuencia de mutaciones, esta llamada debe regresar -1.

Restricciones

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

Subtareas

1. (21 puntos) $y - x \leq 2$
2. (22 puntos) $q \leq 500,\ y - x \leq 1000$, cada cáracter de a y b es "A" o "T".
3. (13 puntos) Cada cáracter 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-esima llamada de `get_distance`.

La salida del evaluador de ejemplo tiene el siguiente formato:

- línea $1 + i$ ($0 \leq i \leq q - 1$) : el valor regresado por la i-esima llamada de `get_distance`.