Mutating DNA

Grace es una biologa trabajando en una empresa de Bioinformática en Singapur. Como parte de su trabajo analiza secuencias de DNA de varios organismos. Una secuencia de DNA se define como una cadena de carácteres "A", "T" y "C". Nótese que para esta tarea la secuencia de DNA **no contiene el carácter "G"**.

Definimos un mutación como una operación en la secuencia de DNA donde dos elementos de la secuencia han sido intercambiados. Por ejemplo una única mutación podría transformar "A**C**T**A**" en "A**A**T**C**" intercambiando los carácteres "A" y "C" marcados.

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

Grace está analizando dos secuencias de DNA a y b, las dos consisten en n carácteres con índices de 0 a $n-1$. 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 subcadena $s[x..y]$ de la secuencia de DNA s se define como una secuencia de carácteres consecutivos de s, cuyos índices van desde x hasta y incluidos. En otras palabras, $s[x..y]$ es la secuencia $s[x]s[x+1]\ \ldots\ s[y]$.

Detalles de implementación

Tienes que implementar los siguientes procedimientos.

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

- a, b: strings de longitud n, describiendo dos secuencias de DNA a analizar.
- Este procedimiento se llama una vez exactamente, antes de hacer las llamadas a `get_distance`.

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

- x, y: valores de inicio y final de la subcadena a analizar.
- El procedimiento debe devolver la distáncia de mutación entre las subcadenas $a[x..y]$ y $b[x..y]$.
- Este procedimiento es llamado exactamente q veces.

Ejemplos

Considera la siguiente llamada:

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

Digamos que el grader llama `get_distance(1, 3)`. Esta llamada debería devolver la distancia de mutación entre $a[1..3]$ y $b[1..3]$, que es, la secuencia "TAC" y "CTA". "TAC" se puede transformar en "CTA" vía 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 pués, esta llamada debería devolver 2.

Digamos que el grader llama `get_distance(4, 5)`. Esta llamada debería devolver la distancia de mutación entre las secuencias "AT" y "TA". "AT" se puede transformar en "TA" usando una única mutación, y claramente se necesita como mínimo una.

Así pués, esta llamada debería devolver 1.

Finalmente, digamos que el grader llama `get_distance(3, 5)`. Como es **imposible** para la secuencia "CAT", de transformarse en "ATA" con una secuencia de mutaciones, esta llamada debería devolver -1.

Restricciones

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

Subtareas

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

Sample grader

El sample grader lee la entrada en el siguiente formato:

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

El sample grader escribe tu respuesta en el siguiente formato:

- line $1 + i$ ($0 \leq i \leq q - 1$) : el valor devuelto por la i-ésima llamada de `get_distance`.