Mutierte DNA

Grace ist eine Biologin und arbeitet für ein Bioinformatikunternehmen in Singapur. Eine ihrer Aufgaben besteht darin, DNA-Sequenzen verschiedener Organismen zu untersuchen. Eine DNA-Sequenz ist eine Zeichenkette, welche aus den Buchstaben "A", "T" und "C" besteht. Insbesondere enthalten DNA-Sequenzen in dieser Aufgabe **den Buchstaben "G" nicht.**

Eine Mutation ist eine Operation auf DNA-Sequenzen, wobei zwei Buchstaben vertauscht werden. Zum Beispiel kann eine Mutation die DNA-Sequenz "A**C**T**A**" in "A**A**T**C**" umwandeln, indem die fettgedruckten Buchstaben "A" und "C" vertauscht werden.

Die Mutationsdistanz zwischen zwei Sequenzen ist die kleinste Anzahl an Mutationen, welche nötig ist um eine Sequenz in die andere umzuwandeln, oder -1, falls es nicht möglich ist, eine Sequenz nur mittels Mutationen in die andere umzuwandeln.

Grace analysiert zwei DNA-Sequenzen a und b der Länge n, welche jeweils von 0 bis $n-1$ indiziert sind. Deine Aufgabe ist es, Grace bei q Fragen der Form "was ist die Mutationsdistanz zwischen den Teilsequenzen $a[x..y]$ und $b[x..y]$?" zu helfen. Mit Teilsequenz $s[x..y]$ ist hier eine Sequenz aufeinanderfolgender Buchstaben gemeint, deren Indizes alle Zahlen von x bis y inklusive enthalten. Anders ausgedrückt ist $s[x..y]$ die Sequenz $s[x]s[x+1]\ \ldots\ s[y]$.

Implementierungsdetails

Implementiere folgende Funktionen:

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

- a, b: Zeichenketten der Länge n, welche die zu analysierenden DNA-Sequenzen darstellen.
- Diese Funktion wird genau einmal aufgerufen, nämlich vor dem ersten Auruf von `get_distance`.

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

- x, y: Start- und Endindex der zu untersuchenden Teilsequenz.
- Diese Funktion soll die Mutationsdistanz zwischen den Teilsequenzen $a[x..y]$ und $b[x..y]$ zurückgeben.
- Diese Funktion wird genau q-mal aufgerufen.

Beispiel

Betrachte den folgenden Funktionsaufruf:

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

Nimm an, dass der Grader `get_distance(1, 3)` aufruft. Dieser Aufruf soll die Mutationsdistanz zwischen $a[1..3]$ und $b[1..3]$, also den Sequenzen "TAC" und "CTA" zurückgeben. "TAC" kann mittels 2 Mutationen in "CTA" umgewandelt werden: **TA**C $\rightarrow$ **C**A**T**, gefolgt von C**AT** $\rightarrow$ C**TA**, und es ist unmöglich, diese Umwandlung mit weniger als 2 Mutationen durchzuführen.

Aus diesem Grund sollte der Aufruf 2 zurückgeben.

Nimm nun an, dass der Grader `get_distance(4, 5)` aufruft. Dieser Aufruf soll die Mutationsdistanz zwischen "AT" und "TA" zurückgeben. "AT" kann mit einer Mutation in "TA" umgewandelt werden und eine Mutation ist auch sicher notwending.

Aus diesem Grund sollte der Aufruf 1 zurückgeben.

Nimm schließlich an, dass der Grader `get_distance(3, 5)` aufruft. Da es **keine** Möglichkeit gibt, die Sequenz "CAT" nur mittels Mutationen in "ATA" umzuwandeln, sollte der Aufruf -1 zurückgeben.

Beschränkungen

- $1 \leq n, q \leq 100\ 000$
- $0 \leq x \leq y \leq n - 1$
- Jedes Zeichen von a und b ist entweder "A", "T", oder "C".

Subtasks

1. (21 Punkte) $y - x \leq 2$
2. (22 Punkte) $q \leq 500,\ y - x \leq 1000$, jeder Buchstabe von a und b ist entweder "A" oder "T".
3. (13 Punkte) Jeder Buchstabe von a und b ist entweder "A" oder "T".
4. (28 Punkte) $q \leq 500,\ y - x \leq 1000$
5. (16 Punkte) Keine weiteren Beschränkungen.

Beispiel-Grader

Der Beispiel-Grader liest die Eingabe im folgenden Format:

- Zeile 1: n q
- Zeile 2: a
- Zeile 3: b
- Zeile $4 + i$ ($0 \leq i \leq q - 1$): x y für den i-ten Aufruf von `get_distance`.

Der Beispiel-Grader gibt deine Antworten im folgenden Format aus:

- Zeile $1 + i$ ($0 \leq i \leq q - 1$) : der Rückgabewert des i-ten Aufrufs von `get_distance`.