Mutating DNA

Grace is een bioloog die voor een bioinformatica-bedrijf in Singapore werkt. Als onderdeel van haar werk analyseert ze DNA-reeksen van verschillende organismen. Een DNA-reeks is gedefinieerd als een string met de karakters "A", "T" en "C". Merk op dat in deze taak DNA-reeksen het **karakter "G" niet bevatten.**

We definiëren een mutatie als een bewerking op een DNA-reeks waar twee elementen worden omgewisseld. Een enkele mutatie kan bijvoorbeeld "A**C**T**A**" omzetten in "A**A**T**C**" door de gemarkeerde karakters "A" en "C" om te wisselen.

De mutatie-afstand tussen twee reeksen is het minimum aantal mutaties dat nodig is om de ene reeks om te zetten in de andere, of -1 als het onmogelijk is de ene reeks door middel van mutaties om te zetten in de ander.

Grace analyseert twee DNA-reeksen a en b die beide bestaan uit n elementen met indices van 0 tot en met $n-1$. Jouw taak is om Grace te helpen met het beantwoorden van q vragen van de volgende vorm: wat is de mutatie-afstand tussen deelreeks $a[x..y]$ en deelreeks $b[x..y]$? We definiëren hier de deelreeks $s[x..y]$ van een DNA-reeks s als de reeks aaneengesloten karakters van s waarvan de indices x tot en met y zijn. Met andere woorden, $s[x..y]$ is de reeks $s[x]s[x+1]\ldots s[y]$.

Implementatiedetails

Implementeer de volgende functies:

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

- a, b: strings van lengte n, die de twee DNA-reeksen die moeten worden geanalyseerd beschrijven.
- Deze functie wordt precies één keer aangeroepen, voordat `get_distance` aan wordt geroepen.

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

- x, y: de begin- en eindindex van de deelreeksen die moeten worden geanalyseerd.
- De functie moet de mutatie-afstand tussen de deelreeksen $a[x..y]$ en $b[x..y]$ teruggeven.
- De functie wordt precies q keer aangeroepen.

Voorbeeld

Neem de volgende aanroep:

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

Stel dat de grader nu `get_distance(1, 3)` aanroept. Deze aanroep moet de mutatie-afstand tussen $a[1..3]$ en $b[1..3]$ teruggeven, dus tussen de reeksen "TAC" en "CTA". "TAC" can omgezet worden naar "CTA" via 2 mutaties: **TA**C $\rightarrow$ **CA**T, gevolgd door C**AT** $\rightarrow$ C**TA**. Het is onmogelijk dit in minder dan 2 mutaties te doen.

Daarom moet voor deze aanroep 2 worden teruggegeven.

Stel dat de grader `get_distance(4, 5)` aanroept. Deze aanroep moet de mutatie-afstand tussen "AT" en "TA" teruggeven. "AT" kan met één transformatie worden omgezet naar "TA" en het is duidelijk dat minstens één mutatie nodig is.

Daarom moet voor deze aanroep 1 worden teruggegeven.

Stel dat tenslotte `get_distance(3, 5)` wordt aangeroepen. Aangezien er **geen manier** is om de reeks "CAT" om te zetten naar "ATA" via wat voor reeks mutaties dan ook, zal er -1 moeten worden teruggegeven.

Randvoorwaarden

- $1 \leq n, q \leq 100\,000$
- $0 \leq x \leq y \leq n - 1$
- Elk karakter van a en b is een van "A", "T" en "C".

Subtasks

1. (21 punten) $y - x \leq 2$
2. (22 punten) $q \leq 500,\ y - x \leq 1000$, elk karakter van a en b is ofwel "A" of "T".
3. (13 punten) Elk karakter van a en b is ofwel "A" of "T".
4. (28 punten) $q \leq 500,\ y - x \leq 1000$
5. (16 punten) Geen aanvullende randvoorwaarden.

Voorbeeldgrader

De voorbeeldgrader leest de invoer in het volgende formaat:

- regel 1: n q
- regel 2: a
- regel 3: b
- regel $4 + i$ ($0 \leq i \leq q - 1$): x y voor de i-de aanroep aan `get_distance`.

De voorbeeldgrader schrijft je antwoorden naar de uitvoer in het volgende formaat:

- regel $1 + i$ ($0 \leq i \leq q - 1$): de teruggegeven waarde van de i-de aanroep aan `get_distance`.