Mutace DNA

Grace pracuje jako biolog v jedné bioinformatické firmě v Singapuru. Součástí její práce je analýza sekvencí DNA různých organizmů. Sekvence DNA je definována jako znakový řetězec tvořený znaky "A", "T" a "C". V této úloze sekvence DNA **neobsahuje znaky "G".**

Mutací rozumíme takovou operaci se sekvencí DNA, kdy se dva znaky sekvence vzájemně prohodí. Například jednoduchá mutace může převést "A**C**T**A**" na "A**A**T**C**" prohozením zvýrazněných znaků "A" a "C".

Mutační vzdálenost dvou sekvencí definujeme jako minimální počet mutací potřebný k převedení jedné sekvence na druhou. Není-li možné převést jednu sekvenci na druhou pomocí mutací, jejich mutační vzdálenost je -1.

Grace právě analyzuje dvě sekvence DNA označené a a b. Každá je tvořena n znaky s indexy od 0 do $n-1$. Vaším úkolem je zodpovědět q otázek typu: Jaká je mutační vzdálenost mezi podřetězcem $a[x..y]$ a podřetězcem $b[x..y]$? Podřetězcem $s[x..y]$ v DNA sekvenci s rozumíme posloupnost po sobě jdoucích znaků řetězce s s indexy od x do y včetně. Jinými slovy řečeno, $s[x..y]$ je posloupnost znaků $s[x]s[x+1]\ \ldots\ s[y]$.

Implementační detaily

Implementujte následující funkce:

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

- a, b: znakové řetězce délky n, popisující dvě analyzované sekvence DNA.
- Tato funkce je volána právě jednou, před všemi voláními funkce `get_distance`.

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

- x, y: počáteční a koncový index analyzovaného podřetězce.
- Tato funkce vrací mutační vzdálenost mezi podřetězci $a[x..y]$ a $b[x..y]$.
- Tato funkce bude zavolána přesně q krát.

Příklad

Uvažujme následující volání:

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

Nechť vyhodnocovač zavolá `get_distance(1, 3)`. Toto volání vrátí mutační vzdálenost mezi $a[1..3]$ a $b[1..3]$, tedy vzdálenost sekvencí "TAC" a "CTA". "TAC" lze převést na "CTA" pomocí 2 mutací: nejprve **TA**C $\rightarrow$ **CA**T, a poté C**AT** $\rightarrow$ C**TA**. Méně než 2 mutace k převodu nestačí.

Volání proto vrátí výslednou hodnotu 2.

Nechť vyhodnocovač zavolá `get_distance(4, 5)`. Toto volání vrátí mutační vzdálenost mezi sekvencemi "AT" a "TA". "AT" můžeme převést na "TA" jednou mutací a je zřejmé, že alespoň jedna mutace je k převodu skutečně zapotřebí.

Volání proto vrátí výslednou hodnotu 1.

Nechť vyhodnocovač zavolá `get_distance(3, 5)`. Neexistuje **žádný** způsob, jak pomocí mutací převést sekvenci "CAT" na "ATA", proto toto volání vrátí -1.

Omezení

- $1 \leq n, q \leq 100\ 000$
- $0 \leq x \leq y \leq n - 1$
- Každý znak řetězců a a b je "A", "T" nebo "C".

Podúlohy

1. (21 bodů) $y - x \leq 2$
2. (22 bodů) $q \leq 500,\ y - x \leq 1000$, každý znak řetězců a a b je buď "A" nebo "T".
3. (13 bodů) každý znak řetězců a a b je buď "A" nebo "T".
4. (28 bodů) $q \leq 500,\ y - x \leq 1000$
5. (16 bodů) Žádná další omezení.

Ukázkový vyhodnocovač

Ukázkový vyhodnocovač čte vstup v následujícím tvaru:

- řádek 1: n q
- řádek 2: a
- řádek 3: b
- řádek $4 + i$ ($0 \leq i \leq q - 1$): x y pro i-té volání funkce `get_distance`.

Odpověď vypíše v následujícím tvaru:

- řádek $1 + i$ ($0 \leq i \leq q - 1$): výsledná hodnota i-tého volání funkce `get_distance`.