Mutating DNA

Paula je programerka koja je igrom slučaja zalutala u laboratorij vrckastog doktora Malnara. Doktor Malnar odmah joj je ponudio osvježavajuće piće, ljutu papričicu i posao u laboratoriju. Vidjevši njegovu papričicu, Paula je odmah na sve pristala i sada se bavi analizom DNA sekvenci raznih organizama (u svrhe stvaranja još jačih papričica). DNA sekvencu definiramo kao string znakova "A", "T" i "C". Naime, za potrebe dobivanja žestokih papričica, DNA sekvence **neće sadržavati znak "G".**

Nad DNA sekvencom definiramo operaciju mutacije u kojoj dva elementa sekvence mijenjaju mjesta. Primjerice, jedna mutacija može transformirati sekvencu "A**C**T**A**" u sekvencu "A**A**T**C**" zamjenom mjesta masno otisnutih znakova "A" i "C".

Mutacijska udaljenost između dvije sekvence jest najmanji broj mutacija potreban da se jedna sekvenca transformira u drugu, ili -1 ako takvu transformaciju nije moguće postići.

Paula analizira dvije DNA sekvence a i b koje se (obje) sastoje od n znakova indeksiranih od 0 do $n-1$. U sklopu svoje analize, Paula treba odgovoriti na q pitanja oblika: koja je mutacijska udaljenost između podstringa $a[x..y]$ i podstringa $b[x..y]$?

Pritom, podstring $s[x..y]$ DNA sekvence s definiran je kao slijed uzastopnih znakova sekvence s, čiji su indeksi između x i y (uključivo). Drugim riječima, $s[x..y]$ je string $s[x]s[x+1]\ \ldots\ s[y]$.

Implementacijski detalji

Potrebno je implementirati sljedeće procedure:

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

- a, b: stringovi duljine n, predstavljaju dvije DNA sekvence koje je potrebno analizirati.
- Ova će se procedura pozvati točno jednom, prije prvog poziva procedure `get_distance`.

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

- x, y: početni i završni indeks podstringova koje je potrebno analizirati.
- Procedura treba vratiti mutacijsku udaljenost između podstringova $a[x..y]$ i $b[x..y]$.
- Procedura će biti pozvana točno q puta.

Primjer

Promotrimo sljedeći poziv:

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

Recimo da će ocjenjivač pozvati `get_distance(1, 3)`. Ovaj poziv treba vratiti mutacijsku udaljenost između $a[1..3]$ i $b[1..3]$, odnosno sekvenci "TAC" i "CTA". Sekvencu "TAC" možemo transformirati u "CTA" koristeći 2 mutacije: **TA**C $\rightarrow$ **CA**T, i nakon toga C**AT** $\rightarrow$ C**TA**. Nemoguće je napraviti transformaciju u manje od dvije mutacije.

Stoga, procedura treba vratiti 2.

Recimo da će ocjenjivač pozvati `get_distance(4, 5)`. Ovaj poziv treba vratiti mutacijsku udaljenost između sekvenci "AT" i "TA". Sekvencu "AT" možemo transformirati u "TA" jednom mutacijom, a očito je barem jedna mutacija potrebna.

Stoga, procedura treba vratiti 1.

Konačno, pretpostavimo da će ocjenjivač pozvati `get_distance(3, 5)`. Budući da **ne postoji** način da se sekvenca "CAT" transformira u sekvencu "ATA" nekim nizom mutacija, ovaj poziv treba vratiti -1.

Ograničenja

- $1 \leq n, q \leq 100\ 000$
- $0 \leq x \leq y \leq n - 1$
- Svaki znak stringova a i b je "A", "T" ili "C".

Podzadaci

1. (21 bod) $y - x \leq 2$
2. (22 boda) $q \leq 500,\ y - x \leq 1000,$ a i b se sastoje isključivo od znakova "A" i "T".
3. (13 bodova) a i b se sastoje isključivo od znakova "A" i "T".
4. (28 bodova) $q \leq 500,\ y - x \leq 1000$
5. (16 bodova) Nema dodatnih ograničenja.

Ogledni ocjenjivač

Ogledni ocjenjivač čita ulaz u sljedećem obliku:

- redak 1: n q
- redak 2: a
- redak 3: b
- redak $4 + i$ ($0 \leq i \leq q - 1$): x y za i-ti poziv procedure `get_distance`.

Ogledni ocjenjivač ispisuje izlaz u sljedećem obliku:

- redak $1 + i$ ($0 \leq i \leq q - 1$): izlazna vrijednost i-tog poziva procedure `get_distance`.