DNA mutante

La biologa Grace analizza le sequenze di DNA di vari organismi, rappresentate come stringhe di caratteri "A", "T", e "C". Nota che queste sequenze **non contengono il carattere "G".**

Definiamo una *mutazione* come uno scambio tra due qualsiasi elementi di una sequenza di DNA: per esempio, una singola mutazione può trasformare "A**C**T**A**" in "A**A**T**C**" scambiando i caratteri "A" e "C" evidenziati.

La *distanza di mutazione* tra due sequenze è quindi il minimo numero di mutazioni necessarie a trasformare una sequenza nell'altra (oppure -1 se non è possibile farlo).

Grace deve analizzare due sequenze a e b, entrambe di n elementi numerati da 0 a $n-1$, e deve rispondere a q domande del tipo: *qual è la distanza di mutazione tra le sottostringhe* $a[x..y]$ *e* $b[x..y]$*?* (con $s[x..y]$ denotiamo la sequenza $s[x]s[x+1]\ldots s[y]$).

Note di implementazione

Devi implementare le seguenti funzioni:

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

- a, b: le sequenze di DNA di lunghezza n da analizzare.
- Questa funzione è chiamata esattamente una volta, prima di tutte le chiamate a `get_distance`.

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

- x, y: indici di inizio e fine delle sottostringhe da analizzare.
- Questa funzione deve restituire la distanza di mutazione tra le sottostringhe $a[x..y]$ e $b[x..y]$.
- Questa funzione viene chiamata esattamente q volte.

Esempio

Considera la seguente chiamata:

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

Supponiamo il grader chiami `get_distance(1, 3)`. Questa chiamata deve restituire la distanza di mutazione tra $a[1..3]$ e $b[1..3]$, e cioè "TAC" e "CTA". "TAC" può essere trasformata in "CTA" tramite 2 mutazioni: **T**A**C** $\rightarrow$ **C**A**T**, seguita da C**AT** $\rightarrow$ C**TA**, e la trasformazione è impossibile con meno di 2 mutazioni. Quindi, questa chiamata deve restituire 2.

Assumiamo che poi il grader chiami `get_distance(4, 5)`. Questa chiamata deve restituire la distanza di mutazione tra "AT" e "TA": per farlo è necessaria e sufficiente una singola mutazione, e quindi la chiamata deve restituire 1.

Infine, supponiamo il grader chiami `get_distance(3, 5)`. Dato che **non esiste** un modo per trasformare la sequenza "CAT" in "ATA" tramite mutazioni, questa chiamata deve restituire -1.

Assunzioni

- $1 \leq n, q \leq 100\ 000$.
- $0 \leq x \leq y \leq n - 1$.
- I caratteri di a e b sono tutti "A", "T", o "C".

Subtask

1. (21 punti) $y - x \leq 2$.
2. (22 punti) $q \leq 500$, $y - x \leq 1000$, e a e b consistono dei soli caratteri "A" e "T".
3. (13 punti) a e b consistono dei soli caratteri "A" e "T".
4. (28 punti) $q \leq 500$, $y - x \leq 1000$.
5. (16 punti) Nessuna limitazione aggiuntiva.

Grader di esempio

Il grader di esempio legge l'input nel seguente formato:

- riga 1: n q
- riga 2: a
- riga 3: b
- righe $4 + i$ ($0 \leq i \leq q - 1$): x y per la i-esima chiamata a `get_distance`.

Il grader di esempio stampa l'output nel seguente formato:

- righe $1 + i$ ($0 \leq i \leq q - 1$) : il valore restituito dalla i-esima chiamata a `get_distance`.