

Мутации ДНК

Грейс – биолог, она работает в компании, занимающейся биоинформатикой в Сингапуре. На работе она занимается анализом ДНК различных организмов. В этой задаче последовательность ДНК определена как строка, состоящая из символов "A", "T" и "C". Обратите внимание, что в этой задаче последовательности ДНК **не могут содержать символ "G"**.

Определим мутацию как операцию на последовательности ДНК, в результате которой два произвольных элемента этой последовательности меняются местами. Например, в результате мутации можно преобразовать последовательность "АСТА" в последовательность "ААТС", поменяв местами выделенные жирным символы "А" и "С".

Мутационным расстоянием между двумя последовательностями назовем минимальное число мутаций, необходимое для превращения одной последовательности ДНК в другую, либо число -1 , если превратить одну последовательность ДНК в другую с помощью мутаций нельзя.

Грейс анализирует две последовательности ДНК a и b , каждая из которых состоит из n элементов, проиндексированных от 0 до $n - 1$. Ваша задача — помочь Грейс ответить на q запросов следующего формата: чему равно мутационное расстояние между подстрокой $a[x..y]$ и подстрокой $b[x..y]$? Здесь подстрока $s[x..y]$ для последовательности ДНК s определяется как последовательность подряд идущих символов s с индексами от x до y , включительно. Другими словами, $s[x..y]$ представляет собой последовательность $s[x]s[x + 1] \dots s[y]$.

Детали реализации

Вам следует реализовать следующие функции:

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

- a , b : строки длины n , задающие две последовательности ДНК, которые необходимо проанализировать.
- Эта функция будет вызвана ровно один раз, до вызовов функции `get_distance`.

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

- x , y : начальный и конечный индекс подстрок, которые необходимо проанализировать.
- Функция должна вернуть мутационное расстояние между $a[x..y]$ и $b[x..y]$.
- Эта функция будет вызвана q раз.

Пример

Рассмотрим следующий вызов функции:

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

Пусть затем грейдер делает вызов функции `get_distance(1, 3)`. Этот вызов должен вернуть мутационное расстояние между $a[1..3]$ и $b[1..3]$, то есть между последовательностями ДНК "ТАС" и "СТА". "ТАС" может быть преобразована в "СТА" с помощью 2 мутаций: **ТАС** → **САТ**, затем **САТ** → **СТА**, а выполнить преобразование менее, чем за 2 мутации, невозможно.

Следовательно, данный вызов должен вернуть 2.

Пусть затем грейдер делает вызов функции `get_distance(4, 5)`. Этот вызов должен вернуть мутационное расстояние между "АТ" и "ТА". "АТ" может быть преобразована в "ТА" за одну мутацию, и ясно, что требуется хотя бы одна мутация.

Следовательно, данный вызов должен вернуть 1.

Наконец, пусть затем грейдер делает вызов функции `get_distance(3, 5)`. Поскольку **не существует** способа преобразовать последовательность "САТ" в "АТА" с помощью последовательности мутаций, этот вызов должен вернуть -1 .

Ограничения

- $1 \leq n, q \leq 100\,000$
- $0 \leq x \leq y \leq n - 1$
- Каждый символ строк a и b это "А", "Т" или "С".

Подзадачи

1. (21 балл) $y - x \leq 2$
2. (22 балла) $q \leq 500$, $y - x \leq 1000$, каждый символ строк a и b – это "А" или "Т".
3. (13 баллов) каждый символ строк a и b – это "А" или "Т".
4. (28 баллов) $q \leq 500$, $y - x \leq 1000$
5. (16 баллов) Нет дополнительных ограничений.

Пример грейдера

Пример грейдера считывает входные данные в следующем формате:

- строка 1: $n\ q$
- строка 2: a
- строка 3: b
- строка $4 + i$ ($0 \leq i \leq q - 1$): $x\ y$ для i -го вызова `get_distance`.

Пример грейдера выводит результат в следующем формате:

- строка $1 + i$ ($0 \leq i \leq q - 1$): значение, которое вернул i -й вызов `get_distance`.