ДНК

 За повечето от вас биологията не влиза в топ класацията по интересни предмети, но при Иванчо не е така - той е маниак на тема генетика. Сега иска да сътвори армия от мутанти, но за това ще трябва да си поиграе с дезоксирибонуклеиновата киселина или по-просто казано – ДНК. Иванчо вече знае как иска да изглеждат мутантите, т.е. крайния вид на ДНК-то им, но сега ще трябва да го сътвори на практика като прилага поредица от определен брой позволени операции върху вече съществуващо ДНК.

 Може да считаме, че началното (съществуващото) ДНК и желаното от Иванчо са последователности от малки латински букви (низове), а позволените операции се извършват само върху началното (текущото), като те са:

1. Преместване на подниз в текущата последователност от едно място на друго.
2. Обръщане на подниз в текущата последователност.
3. Изтриване на единичен символ от текущата последователност.
4. Вмъкване на единичен символ (малка латинска буква) в текущата последователност.
5. Също така Иванчо разполага с речник, съставен от няколко по-малки части ДНК, които може да налага върху текущата последователност

За да може да живее всеки един от мутантите, задължително условие е

полученото след извършване на операциите (крайното) и желаното от Иванчо ДНК да са с еднаква дължина. Овен това, Иванчо разполага с таблица, която представя генетичната разлика между всеки два символа. Чрез нея, той знае колко близо се е доближил до желаното ДНК.

Както предполагате всичко това ще отнеме много ценно време на Иванчо, а той иска да притежава мутантите възможно най-скоро, все пак кой не иска да владее вселената :).

 Сега Иванчо моли вас, добри програмисти, да му помогнете да сътвори своята армия, като напишете програма **fixdna**, която по зададена начална и желаната от Иванчо ДНК да използва определен брой от позволените операции, за да получи последователност, максимално близка до желаната.

**Вход:** На първия ред на входния файл **fixdna.in** ще са записани три числа **N**, **K** и **T**, като **N** е дължината на началната и крайната последователност, **K** е големина на азбуката; съставена е от първите **K** малки латински букви и **T** е максималният брой операции, които могат да се извършват върху текущата последователност.

На следващите два реда са записани два низа – **startSequence** и **targetSequence**, съответно началната последователност (текущата последователност в началото) и последователността, до която искаме максимално да “доближим” началната.

Изпълнено е че:

Дължината на **startSequence** = дължината на **targetSequence** = **N**;

**startSequence** и **targetSequence** са съставени от първите **K** малки латински букви.

На следващите **K** реда са записани **K** брой числа – матрицата **charDifference**, чрез която дефинираме “разлика между буквите”.

За нея са изпълнени:

**charDifference**[i][i] = 0, **charDifference**[i][j] = **charDifference**[j][i] за i ≠ j

“Разликата” между буквите c1 и c2 е равна на **charDifference**[c1 - ‘a’][c2 - ‘a’], ако c1 и c2 са от тип char; индексирането започва от 0.

На следващия ред е записано числото **V** - големина на речника (брой думи в него)

На всеки от следващите **V** реда се съдържа един стринг - **wordi** - i-тата дума от речника, съставена от букви от азбуката, която дефинирахме по-рано.

**Изход:** Изходният файл **fixdna.out** трябва да съдържа число **C** - броят на операциите, които програмата ще изпълни върху първоначалната последователност.

0 <= **C** <= **T**

На следващите **C** реда е описанието на поредната операция, зададена чрез съответните параметри. Първият параметър е кодът на операцията (от 1 до 5). Операциите се извеждат в реда, в който трябва да бъдат изпълнени.

**Забележка:**  При всички операции индексирането започва от 0

**Операциите** са както следва:

1. **Преместване на подниз**.

 Параметри: **код = 1**, **left**, **right**, **lettersOnTheLeft**

Действие: Взема се подниза **currentSequence**[**left** ... **right**] и се изтрива от **currentSequence**. След това се добавя между позиции **lettersOnTheLeft** - 1 и **lettersOnTheLeft** (с други думи се поставя така че да има **lettersOnTheLeft** символа преди него).

 Условие: 0 <= **left** <= **right** < |currentSequence|,

0 <= **lettersOnTheLeft** <= |currentSequence| (дължината се отчита след премахването на подниза, който сме избрали)

1. **Обръщане на подниз**

 Параметри: **код = 2, left, right**

 Действие: Взима се подниза **currentSequence**[**left** ... **right**] и елементите му се обръщат огледално (извършва се операция reverse). Поднизът запазва позицията си.

 Условие: 0 <= **left** <= **right** < |currentSequence|

1. **Изтриване на символ**

 Параметри: **код = 3, index**

 Действие: Изтрива символът, който се намира на позиция **index**.

 Условие: 0 <= **index** < |currentSequence|

Операцията е позволена само ако стрингът не е празен.

1. **Вмикване на символ**

 Параметри: **код = 4, index, letter**

 Действие: Всички символи от интервала [**index**, |currentSequence| - 1] се изместват с една позиция надясно, а символът **letter** се поставя на позиция **index**.

Условие: 0 <= **index** <= |currentSequence|.

**letter** трябва да е част от азбуката, дефинирана от **K**.

 Позволено е текущият стринг да стане с дължина по-голяма от **N**.

1. **Налагане на дума от речника**

 Параметри: **код = 5, wordIndex, positionIndex**

 Действие: Взема се думата dictionary[**wordIndex**] и се “налага” върху последователността, започвайки от позиция **positionIndex**. Елементите на подниза currentSequence[**positionIndex** ... **positionIndex** + |word|] се заместват с думата.

 Условие: 0 <= **wordIndex** < **V**;

 0 <= **positionIndex** <= **N** – |word|.

 Думата не трябва да излиза извън границите на последователността.

**Оценяване:**

Ако някоя от операциите е невалидна, програмата получава 0 точки на съответния тест.

Нека означим крайната последователност с finalSequence.

Ако |finalSequence| ≠ N, програмата получава 0 точки на съответния тест.

Иначе дефинираме *yourScore* = ∑ charDifference[finalSequence[i]][targetSequence[i]]**,** i = 0..N-1

*minScore* е минималният резултат, получен от всички коректни програми за този тест.

Програмата получава (*minScore / yourScore*) ^ 2 процента от точките, предвидени за съотвения тест.

**Ограничения:**

За всички тестове:

K <= 26

В 10% от тестовете:

N <= 100

T <= 20

V <= 5

|wordi| <= 10

0 <= charDifference[i][j] <= 20

В 20% от тестовете:

N <= 1000

K <= 5

T <= 100

V = 0

0 <= charDifference[i][j] <= 20

В 20% от тестовете:

N <= 5000

T <= 500

V <= 10

|wordi| <= 25

0 <= charDifference[i][j] <= 30

В 25% от тестовете:

N <= 20 000

K <= 15

T <= 1000

V <= 25

|wordi| <= 50

0 <= charDifference[i][j] <= 40

В 25% от тестовете:

N <= 200 000

T <= 15 000

V <= 50

|wordi| <= 100

0 <= charDifference[i][j] <= 50

**Ограничение за време**: 8 сек

**Ограничение за памет**: 256 MB

Предварителни тестове: 20

Финални тестове: 100

**Примерен тест:**

|  |  |
| --- | --- |
| **fixdna.in** | **fixdna.out** |
| 10 3 5abcbaaccbbbbaaccacba0 2 32 0 13 1 03abcaababcc | 51 3 6 23 05 0 62 6 84 6 a |
| Score: 0 |

**Обяснение на изхода**:

Текущата последователност след всяка операция е:

“abc**baac**cbb” -> “ab**baac**ccbb”

“**a**bbaacccbb” -> “bbaacccbb”

“bbaacccbb” -> “bbaacc**abc**”

“bbaacc**abc**” **-> “**bbaacc**cba”**

“bbaacccba” -> “bbaacc**a**cba”

Score = 0, защото за i={0..N-1} charDifference[startSequence[i]][currentSequence[i]] == 0.

С други думи startSequence == currentSequence.