**Bronze** - **Perfect Squares for Not-So-Perfect Cases**

After writing so many problems and test cases, Jeffrey is having a horrible headache! However, he must finish writing the problems for the Seven Week Challenge! At this point, in this much pain, Jeffrey really doesn’t care about how much the headache will affect the quality of his problems. He is happy and will feel better about himself as long as he gets them done.

Jeffrey is currently writing up a problem on numbers. He would like the users to classify all numbers from ***A*** to ***B*** (inclusive) based on the properties associated with the numbers. As a child prodigy, Jeffrey knows how to classify even and odd numbers, as well as prime and composite numbers. But after quitting Sunday math, Jeffrey has forgotten how to identify if a number is a perfect square or not!

Of course, the users cannot know about this! Jeffrey must hide this fact and has come to you for help in identifying the perfect squares. Jeffrey will give you two integers, ***A*** and ***B***, and would like to know how many perfect squares there are within the range (inclusive)!

**Input Specifications**

The input consists of a line with two space-separated integers, ***A*** and ***B***.

* (-10000<=A,B<=10000)

**Output Specifications**

An integer representing the number of perfect squares between***A*** and ***B*** inclusive.

**Sample Input**

10 50

**Sample Output**

4

**Silver - Tests or Test Cases?**

Oh no! Jeffrey has a 2-day Mohan Physics test tomorrow on Melanie’s theorem and being the responsible student that he is, he hasn’t started studying yet! To nobody’s surprise, Jeffrey also has not started creating the test case input files for the weekly Seven Week Challenge which are due in 12 hours!

Not wanting to be rejected from University of Waterloo’s Computer Science program with his physics marks, Jeffrey has decided to go ham on studying for the night so he can pass the test tomorrow and the day after. What does this mean for the fate of the test cases? He’ll probably just make them up sometime before he goes to bed, or after he wakes up, or never. Oh well.

While solving a block-on-block centripetal force with a pulley problem using Melanie’s Chain Rule, Jeffrey suddenly came across a great idea! He can just generate all of the test cases by writing a program to do it. Jeffrey was very proud of his genius idea and put Mohan studying aside to work on the program.

However, two hours has passed and Jeffrey has yet to create a functioning program to generate the cases! He is now very frustrated, realizing that he has already lost two hours of precious studying time and made no progress. Moreover, he realized that the program that he wants to write will not earn him the acceptance from his dream school, but his physics mark will.

Worried about his university acceptance, mark in Mohan’s class and the chances of him making out of Massey successfully with enough credits, Jeffrey has assigned you with the task of writing the program for him.

Honoured to be the programming slave for the based-god Jeffrey, you are ready to create any program that he wants in order to make him happy. Knowing this, Jeffrey has a list of restrictions for generating his test cases. You must comply to the restrictions or Jeffrey’s cases will turn out to be horrible, and the participants of the contest will not be happy about it!

For each of Jeffrey’s test cases, he will need to generate all the words possible, line-by-line, that follow a specific set of rules. To generate them, Jeffrey will be able to use all of the letters in the alphabet in lowercase. However, there are rules on specific types of letters that can precede other specific types of letters. In additional to these rules, the length of the words generated must not exceed some given input, ***L***.

Jeffrey will tell you the number of different restrictions, ***N***, there are for generating the words. Each ‘restriction set’ contains a set of letters. For any given ‘restriction set’, ***S***, any of the letters contained in any of the preceding restriction sets can be followed by any of the letters in the set of ***S***. The words can only begin with the letters given in the first restriction set. (See the sample test cases for clarification)

**Input Specifications**

The first line will contain two space separated integers:

* ***N*** (1<=N<=26): The number of different restrictions sets that will follow
* ***L*** (1<=L<=N): The maximum length of the words generated

The following ***N*** lines will be structured like this:

* ***M*** (1<=M<=26): The number of letters in the current restriction set
* Followed by ***M*** space separated letters on the same input line that are in the current restriction set

**Output Specifications**

Output all possible possible words that comply with the restrictions given by Jeffrey, line-by-line in alphabetical order.

**Sample Input**

4 3

1 a

2 e i

1 j

1 m

**Sample Output**

a

ae

aej

aem

ai

aij

aim

aj

ajm

am

**Gold - Restoring Reputation**

Due to the (lack of) quality in the previous test cases that Jeffrey generated, his test cases are now frowned upon to use by the other problem setters. In an effort to restore his destroyed reputation in the Case Creation Community (CCC), Jeffrey is going through all of the input files he has, word by word, to improve the quality of his test data!

Jeffrey knows that if he doesn’t have good quality test cases this week, he will forever be shamed in the CCC and could never make it to the prestigious Case Creation Organization (CCO).

However, it is also a commonly known fact in the CCC that Jeffrey is a slow and lazy typer! For a given word in the input file, ***A***, Jeffrey would like to transform it to a new word, ***B***. In Jeffrey’s world (filled with angry typing, mechanical keyboards, VIM and Linux), he has three different operations to transform the word.

Jeffrey can…

1. Delete a character from ***A*** using ***D*** seconds
2. Insert a character into ***A*** using ***I*** seconds
3. Replace a character in ***A***with a new character using ***R*** seconds

(The difference in time for the different operations is obviously due to his loud mechanical keyboard.)

Before Jeffrey starts going through all the words, he would like to know how long it will take for him to go through each word so he can find the total time to allocate out of his day. Of course, Jeffrey only wants to know the minimum amount of time he needs to fix each word, he needs all of the remaining time in his day to post dank memes online as a publicity stunt for maximum likes and comments. With this, he could gain enough popularity to be nominated to join the Case Creation Organization!

**Input Specifications:**

On the first line will be four space separated integers:

* ***D*** (0<=D<=1000): The amount of time in seconds required to delete a character
* ***I*** (0<=I<=1000): The amount of time in seconds required to insert a character
* ***R*** (0<=R<=1000): The amount of time in seconds required to replace a character

The next line will contain two space separated strings, ***A*** and ***B***.

It is guaranteed that the length of ***A*** and ***B*** will be less than or equal to ***1000***.

**Output Specifications:**

Output the cheapest cost to transform word ***A*** to word ***B***.

**Sample Input**:

1 2 4

intention execution

**Sample Output:**

12