1. SequenceDetectorMarks: 30

Problem Description

Nitya is a digital circuit designer. She decides to design a sequence detector which may be either an overlapping model or a non-overlapping model.

Sequence detector is a digital circuit which is used to detect the sequences from a series of numbers. These detectors may be overlapping or non-overlapping.

Example of how non-overlapping sequence detector detects sequence 101 is depicted below.

com.tcs.cv.automata.ei.middleware.DocxToHtmlConverter@31c7c281:image1.jpeg

So, from the series given above, say Series 1, non-overlapping sequence detector detects four 101 sequences

Example of how overlapping sequence detector detects sequence 101 is depicted below

com.tcs.cv.automata.ei.middleware.DocxToHtmlConverter@31c7c281:image2.png

From the series given above, say Series 2, overlapping sequence detector detects three 101 sequences. The last 5 digits in the given series i.e., 10101 has two 101 sequences where the middle 1 is used by both left and right trailing digits.

The process of any circuit design starts with requirements and state transition diagram to represent that requirement. So, Nitya asked her assistant to create different state transition diagrams for the hardware needed to detect different sequences. Hardware will need to be customized per sequence. The assistant designed different hardware for different sequences, but forgot to mention the sequence to be detected on respective state transition diagrams. So now, Nitya must go through every sequence and determine, what is the sequence to be detected and whether the sequence detector is overlapping or non-overlapping.

So, she decides to code the state transition diagram and find which sequence is to be detected and what is the type of detector.

state transition diagram example and explanation:

com.tcs.cv.automata.ei.middleware.DocxToHtmlConverter@31c7c281:image3.png

Here a and b represent different states in the state transition diagram. State of circuit changes from one state to another depending on inputs provided by arrow representations. Here, initial state starts with 'a' and then goes to 'b' and then back to 'a'. A sequence can be formed only when state changes.

For example,

com.tcs.cv.automata.ei.middleware.DocxToHtmlConverter@31c7c281:image4.png

Here state changes from a to b when input is 1 and giving output 0. So, the sequence starts with 1. State changes from b to c when input is 0. So, the second digit in sequence is 0. State changes from c to a when input is 1 and giving output as 1, and also when input is 0 and giving output as 0. But the output is 1 only when input is 1 from the above diagram. So, the third digit in sequence is 1. Thus, the sequence that this state transition diagram can detect is 101.

A sequence detector is non-overlapping when state changes from final state to initial state with output being 1. In this case, initial state is a and final state is c. In any other case, it is an overlapping sequence detector. A sequence is said to detected by observing the output when the output goes to 1. So, the above state model is non-overlapping sequence detector.

Constraints

2 <= Length of sequence <= 10

Each input line contains 4 space separated inputs viz. < present_state (char), next_state (char), input (0/1), output (0/1)>

4 < = Number of lines of input < = 20

For a sequence to be detected, every state must be visited at least once and the number of transitions should be equal to Number of states

A given state transition diagram always corresponds to one sequence. As explained in Input section, the number of chars will depict the number of states in a state transition diagram

Input

Input consists of variable number of lines of input where each line contains 4 space separated inputs which represents present_state (char), next_state (char), input (0/1), output (0/1) of the state transition diagram.

Output

Print the sequence to be detected in first line.

Print the type of sequence detector in the second line. If the detector is non-overlapping print "Non Overlapping Sequence Detector" else print "Overlapping Sequence Detector".

Time Limit (secs)

1

Examples

Input

```
a b 1 0

b c 0 0

b b 1 0

a a 0 0

c a 0 0

c a 1 1
```

Output

101

Non Overlapping Sequence Detector

Explanation

Here a and b represent different states in state transition diagram. state of circuit changes from one state to another depending on inputs provided by arrow representation. Initial state starts with 'a' and then goes to 'b' and then to 'c' and so on. A sequence can be formed only when state changes.

com.tcs.cv.automata.ei.middleware.DocxToHtmlConverter@31c7c281:image5.png

Here state changes from a to b when input is 1 and output is 0. So, the sequence starts with 1. State changes from b to c when input is 0. So, the second digit in sequence is 0. State changes from c to a when input is 1 and output is 1, and also when input is 0 and output is 0. But the output is 1 only when input is 1 from the above diagram. So, the third digit in sequence is 1. Thus, the sequence that this state model diagram can detect is 101

Since a sequence detector is non-overlapping when state changes from final state to initial state with output being 1, the state transition diagram above is a non-overlapping sequence detector, where, the initial state is a and final state is c.

Thus, the output is

101

Non Overlapping Sequence Detector

Example 2

Input

a b 1 0

a a 0 0

b a 0 0

b c 1 0

c c 1 0

d a 0 0

d b 1 1

c d 0 0

Output

1101

Overlapping Sequence Detector

Explanation

The diagram for given inputs is:

com.tcs.cv.automata.ei.middleware.DocxToHtmlConverter@31c7c281:image6.png

Using above state transition diagram, the sequence is 1101 and detector is overlapping detector. Thus, the output is

1101

Overlapping Sequence Detector

2. HammingDistanceMarks: 30

Problem Description

Vyom just learned about binary numbers. One day his tutor gave him T similar tasks and asked him to find the answer for them. As the number of tasks is more and also the size of input in each task is large, he concluded that manual calculation will be tough, so he decided to write a program for that.

Given T binary strings of varying lengths which consists of only 0s and 1s. He will be given two values A and B which indicates cost of one occurrence of sub strings "01" and "10". The total cost of the given string will be the sum of the costs of all "01" and "10". His task is to minimize the cost of given strings in each case, by rearranging it in any order. After he rearrange the string, he has to find the hamming distance between the original string and the rearranged string and print it in each case. In case of invalid input, print "INVALID". If there are more than one rearrangement which gives least cost, then consider the string which gives minimum hamming distance.

As Vyom is new to binary strings, he is a bit confused. Can you help Vyom to implement!

Note: The sub strings are considered in an overlapping manner i.e., in the string 010, there is one "01" and one "10".

Constraints

1 <= length of string <= 10^5

0 <= A,B <= 10^4

Input

First line consists of T the number of test cases.

For each test case there will be two lines, first line consists of the binary string and the second line consists of A and B separated by space.

Output

For each string, print the hamming distance in a new line.

Refer E*xamples* section for more clarity.

Time Limit (secs)

1

Examples

Example 1

Input

2

0100

3 2

000

4 5

Output

2

0

Explanation

Here, cost of original string viz. 0100 is 5, because there is one occurrence of both "01" and "10". Now this string can be transformed into a new string viz. 1000 which is having one occurrence of "10". The cost of transformed string= (number of occurrences of "01")*3 + (number of occurrences of "10")*2 = 0*3 + 1*2 = 2 which is the minimum possible and the hamming distance of original and transformed string 2.

The string 000 has the cost of 0 which is minimum, and hence no need to do any transformation. So the hamming distance will be 0.

Example 2

Input

1

01001a10

1 2

Output

INVALID

Explanation

The given string is not binary string.

3. PlaceFinderMarks: 100

Problem Description

The Place Finder is a special tool made for finding locations in areas without internet. To know where you are, you need a group of these devices because they work together. These devices can easily talk to each other.

Each device can find where other devices are and how far they are. This information is shared among devices, creating a network. This network helps with communication and figuring out where devices are in relation to each other. To make the system more accurate, all devices in the network sync up and point in the same direction.

In a real time scenario , each devices is able to communicate with each other, but it is not able to find the direction and distances of all devices due to natural obstacles. However for few devices it may be possible to know their direction and distances. Now Your job is to find how far apart two specific devices are, in this connected network. Each device has a unique ID for easy communication and identification.

The calculations required to find distance between devices can be understood from the through the *Examples section*

Constraints

1 < N < 16

Angle is measured in degree.

Distance is in meters

Input

First line consists of an Integer 'N' denoting total number of place finder devices

Second line consist of 'N' pairs of space separated integers. Each pair is separated by a colon. First value of the pair represents the device ID, second value represents the number of devices that can be found by this device.

Next lines consist of "N" logical sections. Each logical section consists of

- Device ID as its First line,
- Followed as many number of lines as devices that can be found by this device. Each such lines has three space separated Integers denoting "Device ID", " Distance" and "Angle".

Last line consists of two space separated integers representing Id of devices between whom we have to find the distance .

Output

Print the Distance rounded off to two decimal points denoting the distance between the given two devices

Time Limit (secs)

1

Examples

Input

5

1:2 2:3 3:2 4:3 5:2

1

2 9 0

3 8 270

2

5 8 330

4 8 270

1 9 180

3

1 8 90

4 9 0

4

2 8 90

3 9 180

5 8 30

5

2 8 150

4 8 210

1 5

Output

16.42

Explanation

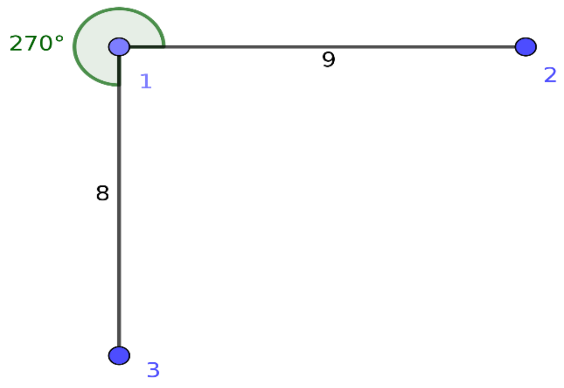
The first line describes there are totally five devices

The second line describes that the device "1" is able to find the direction and distance of 2 devices and device "2" is able to find the direction and distance of 3 devices and so on.

From the third line Individual logical section starts
From the first line of this logical section, we say it is the data from device "1"

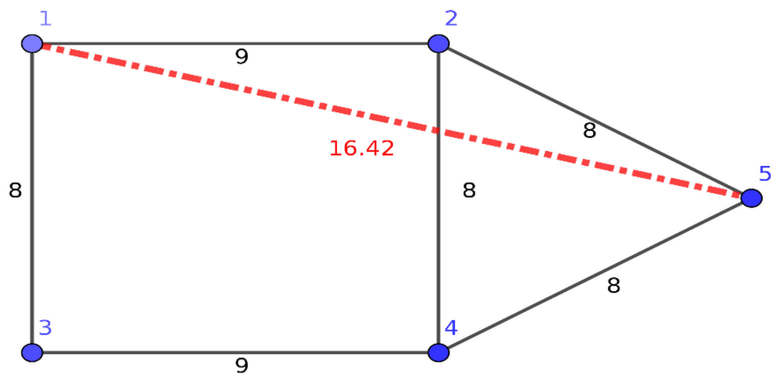
From second line of this logical section, device "2" is located at distance of 9 m from the device "1" at the angle of 0 degree.

From third line of the logical section, device "3" is located at distance of 8 m from the device "1" at an angle of 270 degree.



The above diagram describes the first logical section.

By continuing similarly for all the logical sections, we can visualize the network. Below figure depicts the network



Now when we measure the distance between the device "1" and device "5" we get the output as 16.42.

Example 2

Input

5

1:2 2:3 3:4 4:3 5:2

1

2 5 0

3 5 60

2

1 5 180

3 5 120

4 5 60

3

2 5 300

4 5 0

5 5 60

1 5 240

4

2 5 240

3 5 180

5 5 120

5

3 5 240

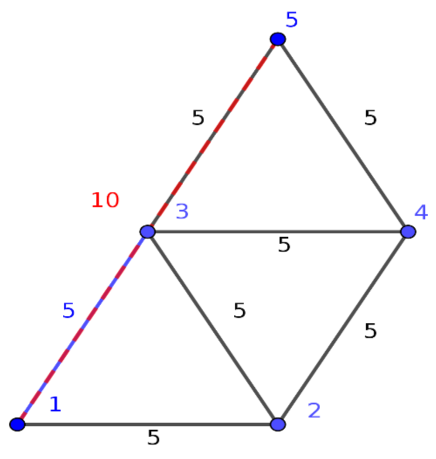
4 5 300

1 5

Output

10.00

Explanation



Above diagram depicts the network. The distance between the device "1" and device "5" are 10.

Hence the output 10.00.

4. ScoreOfCellsMarks: 100

Problem Description

James is a school going kid, who just learned about rows, columns and tables. On one fine day, he drew a table consisting of n rows and m columns. In each cell, he wrote a number randomly.

James defined a concept "score" for each cell. The score of a cell A is the number of unique ways possible from all the cells to reach A. For this, he has to follow a rule which states that, he can only move from current cell to either down/right cell only if the value those cells hold is not lesser than the value in the current cell.

Given an integer *k*, find out the cells with score k. If there are more than one cell, then print the indices of the cells in the order they occur in the table from left to right and top to bottom. Print "NO" if there are no cells with such score.

Note: Two ways are said to be unique if they have at least one different cell in its path.

Constraints

1 <= n,m <= 100

0 <= table[i] <= 10^4

Input

First line contains two integers n,m separated by space.

Next n lines contain the values in the cells of the table.

Next line contains an integer k.

Output

Print the indices of cell(s) with given score in separate lines.

Refer to *Examples* section for better understanding.

Time Limit (secs)

1

Examples

Example 1

Input

4 3

6 4 5

8 5 3

9 7 2

1 9 10

1

Output

0 2

1 0

1 1

Explanation-

The score of a cell A is the number of unique ways possible from all the cells to reach A.

For the cell [0,2], [0,1] -> [0,2] is the only possible way.

For the cell [1,0], [0,0] -> [1,0] is the only possible way.

For the cell [1,1], [0,1] -> [1,1] is the only possible way.

So, the score of all these cells is 1 and we print them in the order of their occurrence.

Example 2

Input

3 3

6 16 19

14 20 17

21 12 11

4

Output

1 1

Explanation-

The score of a cell A is the number of unique ways possible from all the cells to reach A.

For the cell [1,1], four ways are possible.

[0,0] -> [1,0] -> [1,1], [0,0] -> [0,1] -> [1,1], [1,0] -> [1,1], [0,1] -> [1,1]

So the score of [1,1] is 4 and we print it as output.

5. GreedyVirusMarks: 200

Problem Description

Phew, thank goodness you're back! During your absence, we observed some unauthorized data communication occurring from our storage warehouse. We promptly reported this to the chief since some data had already been stolen. The chief instructed us to trace the destination of the data theft. Our storage warehouse comprises individual data containers, arranged in an M*N matrix, with each container holding varying quantities of data represented as units. The containers are organized in M rows and N columns (with indexing starting from 1).

Our observation revealed that the virus consistently targets the container with the maximum data units. However, when two containers have equal data units, the virus's behavior becomes unclear, and we must avoid such scenarios. To determine the data's destination, continuous communication between the virus and the thief must occur. However, our priority is to prevent further data theft.

As the virus targets containers with more data, it always seeks neighboring containers. It can travel in all 8 directions in its quest for infecting container with maximum data. If any of the 8 neighboring container has more data units, the virus moves there. Consequently, our strategy is to entice the virus into a storage unit containing dummy data. Also since data can be generated on the fly, assume that infinite amount of data can be fudged (made available) to lead and trap the virus into desired location.

Information about the virus's current location and the desired location, which is the dummy container, is known. Given our awareness of the data warehouse's size (M*N) and the data stored in the containers, we aim to determine the minimum data required to be produced to entice the virus to the dummy container.

Adding data consumes time, so identifying where to fudge how much data, is crucial. The goal is to trap the virus with minimum amount of fudged data.

Constraints

2 <= N, M <= 7

0 < Units of data in containers < 30

Input

First line consist of two integers separated by space representing M and N.

Next M lines, each contain N space separated integers, representing data units in the corresponding containers.

Next line consist of two space separated integers, indicating current location of the virus

Last line consists of two space separated integers, specifying the location of the dummy container where we want to entice the virus to.

Output

The smallest amount of overall data needed to be fudged to attract the virus to the dummy container.

Time Limit (secs)

1

Examples

input

4 4

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

2 2

4 1

output

5

Explanation

The virus is on container [2,2] and we have to bring it to container [4, 1]. The neighbouring containers of [2, 2] are [(1,1),(1,2),(1,3),(2,1),(2,3),(3,1),(3,2),(3,3)] having data of [1,2,3,5,7,9,10,11] units, respectively. Now we know that virus will move to container [3, 3] because it contains the maximum amount of data in its neighbourhood. However, we also know that from [3, 3] reaching [4, 1] is impossible. Hence we now need to fudge data.

We can fudge and add 3 units of data to [3,1] where already 9 units of data is present. After fudging, it will became 12. 12 units of data will make it the maximum in neighbourhood. The virus will now thus move to [3,1] .

Now the virus is on container [3,1], The neighbouring container of the virus are [(2,1),(2,2),(3,2),(4,1),(4,2)] having data of [5,6,10,13,14] units, respectively. we will add 2 units of data to [4,1] where already 13 units of data is present. Adding 2 unit data will make it 15. 15 units of data is the maximum in the neighbourhood. The virus will then, move to [4,1], our dummy container

A total of 3+2 = 5 units of data was required to be fudged to entice the virus to desired location.

Similarly, one more possiblities is [2,2]->[3,2]->[4,1] which also requires 2 + 3 = 5 units of data to be fudged, Hence output is 5.

Example 2

input

5 6

1 17 18 20 11 10

3 17 15 18 16 15

10 11 20 6 8 3

18 18 5 11 4 16

3 4 8 17 18 20

1 1

3 4

output

16

Explanation

We will add 1 unit of data to [2,2] so they virus will move there, Next we want to move it to [3,3] but it is already the maximum in its neighbourhood. So, no new data needs to be added. Next, 15 units of data is required to make [3,4] maximum. So 15 + 1 = 16.

6. HarmonicHomologyMarks: 200

Problem Description

There are several tunes in the music world. When these tunes are combined, we get a soothing melody. The combination of these melodies will result in a complete song.

Ananya is a musician who always tries to create unique songs. She mixes up these tunes together and forms melodies. There is a standard hierarchy for these tunes which resembles a tree. All the tunes that belong to a certain level sound alike, and the parent of these tunes defines their category.

The hierarchy will be given in the input. Ananya is currently composing a song, to make the song more musical, she will have to form melodies which sound similar.

You will be given the hierarchy of the tunes, two melodies, and three integers A, B and C. A melody is the chain of tunes separated by "-". Two tunes are said to be similar if they are equal or belong to the same level in the hierarchy. You are allowed to perform two operations on tunes i.e., compare or remove. A term "concordance score" is introduced, which is initially zero. When you compare, if tunes are similar, then concordance score will be increased by A or else decreased by B. If you prefer to remove a tune from one of the string, concordance score will be decreased by C. You cannot remove tune at ith index from both the strings simultaneously. You will have to move to the next tune only when the current tune is either compared or removed.

Ananya is trying to increase the similarity between the given melodies by increasing their concordance score by performing the above-mentioned operations. Thus, the more the concordance score, more the similarity!

Help Ananya in finding the maximum concordance score possible.

Note: These terms used here do not resemble original music terms.

Constraints

1 <= number of tunes in each melody <= 500

1 <= A, B, C <= 10^4

Tunes contains only lower-case alphabets.

Input

The first line consists of a single integer N denoting the number of parent nodes.

Next N lines contains music categories in the form of <parent tune> : <list of child tunes separated by space>. The first parent tune is always the root node of the tree.

N+2 line consists of melody m1.

N+3 line consists of melody m2.

The last line consists of three integers A, B and C.

Output

Print the maximum concordance score of the given melodies.

Time Limit (secs)

1

Examples

Example 1

Input

5

pop : rock country disco

rock : jazz blues

country : swing

disco : salsa new ambient

swing : gospel techno

rock-swing-new-salsa-swing-blues

swing-swing-blues-jazz

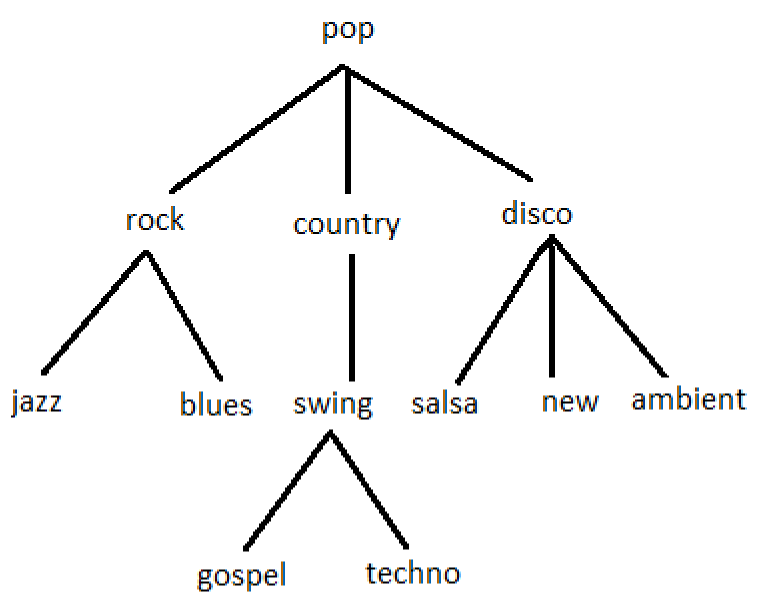
4 2 1

Output

14

Explanation-

Given hierarchy is.



Given melody1 = *rock-swing-new-salsa-swing-blues* and melody2 = *swing-swing-blues-jazz*

Steps followed to get maximum concordance score:

Remove rock and swing tunes from melody1. Melody1 becomes *new-salsa-swing-blues.* Concordance Score = -2

Compare *new* tune from melody1 with *swing* tune from melody2. Both are similar. Concordance Score = -2+4 = 2

Compare *salsa* tune from melody1 with *swing* tune from melody2. Both are similar because both are at the same level. Concordance Score = 2 + 4 = 6

Compare *swing* tune from melody1 with *blues* tune from melody2. Both are similar because both are at the same level. Concordance Score = 6 + 4 = 10

Compare *blues* tune from melody1 with *jazz* tune from melody2. Both are similar because both are at the same level. Concordance Score = 10 + 4 = 14

No other set of operations will give a concordance score greater than 14.

Example 2

Input

4

carol : brio briet heavy

brio : funk

briet : punk

funk : disco

punk-brio-funk-briet

disco-carol-heavy-brio

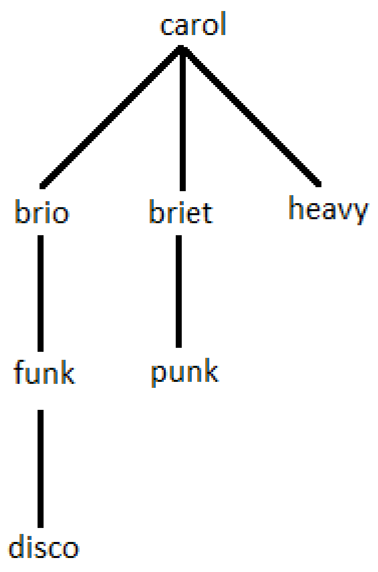
10 8 2

Output

12

Explanation-

Given hierarchy is.



Given melody1 = *punk-brio-funk-briet* and melody2 = *disco-carol-heavy-brio*

Steps followed to get maximum concordance score:

Remove *punk* tune from melody1. Melody1 becomes *brio-funk-briet .* Concordance score = 0 - 2 = -2

Remove *disco, carol* tunes from melody2. Melody2 becomes *heavy-brio*. Concordance score= -2 - 4 = -6

Compare *brio* from melody1 and *heavy* from melody2. Both are similar because both are at the same level. Concordance Score = -6 + 10 = 4

Remove *funk* from melody1. It becomes *brio-briet*. Concordance score = 4 - 2 = 2

Compare *briet* from melody1 and *brio* from melody2. Both are similar because both are at the same level. Concordance Score = 2 + 10 =12

No other set of operations will give concordance score greater than 12.